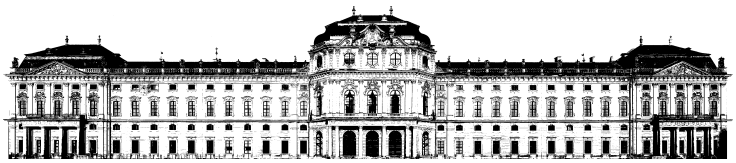
EuroCG 2020



Book of Abstracts

36th European Workshop on Computational Geometry

March 16–18, 2020 in Würzburg, Germany.

Preface

The 36th European Workshop on Computational Geometry (EuroCG 2020) was held at Universität Würzburg, Würzburg, Germany, on March 16–18, 2020. EuroCG is an annual workshop that combines a strong scientific tradition with a friendly and informal atmosphere. The workshop is a forum where researchers can meet, discuss their work, present their results, and establish scientific collaborations, in order to promote research in the field of Computational Geometry, within Europe and beyond.

We received 94 submissions, which underwent a limited refereeing process by the program committee in order to ensure some minimal standards and to check for plausibility. We selected 85 submissions for presentation at the workshop. Three submissions were later withdrawn (leading to gaps in the paper numbering at 2, 14, and 48). EuroCG does not have formally published proceedings; therefore, we expect most of the results outlined here to be also submitted to peer-reviewed conferences and/or journals. This book of abstracts, available through the EuroCG 2020 web site, should be regarded as a collection of preprints. In addition to the 82 contributed talks, this book also contains abstracts of the three invited lectures, given by Erin Wolf Chambers, Otfried Cheong, and Monique Teillaud.

Many thanks to all authors, speakers, and invited speakers for their participation, and to the members of the program committee and all external reviewers for their insightful comments. We gratefully thank the German Research Foundation (DFG grant KI 2477/1-1) for making this event possible and for helping us to keep the registration fees low. Special thanks to Bella Grigoryan, the members of the organizing committee, and the administration at Universität Würzburg, for their work that made EuroCG 2020 possible.

March 2020
Würzburg

Steven Chaplick
Philipp Kindermann
Alexander Wolff
(EuroCG 2020 co-chairs)

Organizing Committee (Universität Würzburg)

Steven Chaplick (co-chair)
Thomas van Dijk
Philipp Kindermann (co-chair)
Jonathan Klawitter
Myroslav Kryven
Andre Löffler
Alexander Wolff (co-chair)
Johannes Zink

Funded by
DFG Deutsche Forschungsgemeinschaft
German Research Foundation

Program Committee

Elena Arseneva	St. Petersburg State University
Michael Bekos	Universität Tübingen
Ahmad Biniaz	University of Windsor
Nicolas Bonichon	Université Bordeaux
Jean Cardinal	Université Libre de Bruxelles
Steven Chaplick (co-chair)	Universität Würzburg and Maastricht University
Olivier Devillers	INRIA Nancy
Emilio Di Giacomo	University of Perugia
Arthur van Goethem	Eindhoven University of Technology
William Evans	The University of British Columbia
Krzysztof Fleszar	University of Warsaw
Philipp Kindermann (co-chair)	Universität Würzburg
Linda Kleist	TU Braunschweig
Saeed Mehrabi	Carleton University
Arnaud de Mesmay	Institut d'électronique et d'informatique Gaspard-Monge (IGM)
Katarzyna Paluch	University of Wroclaw
Zuzana Patáková	IST Austria
Paweł Rzążewski	Warsaw University of Technology
Chan-Su Shin	Hankuk University of Foreign Studies
Bettina Speckmann	Eindhoven University of Technology
Joachim Spoerhase	Aalto University
Miloš Stojaković	University of Novi Sad
Sabine Storandt	Universität Konstanz
Martin Tancer	Charles University in Prague
Geza Tóth	Hungarian Academy of Sciences
Torsten Ueckerdt	Karlsruhe Institute of Technology
Ryuhei Uehara	Japan Advanced Institute of Science and Technology
Alexander Wolff (co-chair)	Universität Würzburg

Table of Contents

(Invited Talk) The Saga of the Skyline Points

Otfried Cheong[1]

1 **SCALGO, Denmark, and KAIST, South Korea**
otfried@kaist.airpost.net

Abstract

Skyline points or non-dominated points in a database are those points that are "best" in at least one of their attributes. In spatial databases, interesting implicit attributes are the distances to a given set of sites of interest. We present some history of the problem, and then show how computational geometry helps to transform it into a question about certain Voronoi diagrams with additive weights and a convex-distance function. Finally, we show how to solve the problem for n data points and m sites of interest in time $O((n+m)\log(n+m))$, improving on all previous results that require time proportial to nm.

Biography

Otfried Cheong received his Ph.D. at FU Berlin in 1992. After holding positions at Utrecht University, Postech, Hong Kong University of Science Technology, and TU Eindhoven, he has been at KAIST since 2005. He is on the editorial board of 'Discrete Computational Geometry' and 'Computational Geometry: Theory Applications', and was elected an ACM Distinguished Scientist in 2016. He is currently on leave from KAIST to work with Scalgo on water flow simulations.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.

(Invited Talk) Triangulations in CGAL: To Non-Euclidean Spaces... and Beyond!

Monique Teillaud[1]

1 **INRIA Nancy - Grand Est, LORIA, France**
monique.teillaud@inria.fr

Abstract

The talk will review some of the basic ideas underlying the design of the classic triangulation packages in CGAL. Then it will present more recent work on the computation of Delaunay triangulations of some flat tori and of the Bolza surface, and show how the CGAL basic ideas could be extended. Triangulations are known to have many applications. The talk will exhibit concrete uses of the various CGAL triangulation packages. Finally, future work and its motivation will be mentioned.

Biography

Former student of the École Normale Supérieure in Paris, holder of an Agrégation in Mathematics and a PhD in Computer Science ("Towards dynamic randomized algorithms in computational geometry"). Managing Editor of JoCG (the free and gratis Journal of Computational Geometry), PC Chair of SoCG'08, Chair of the Computational Geometry Steering Committee since 2016. Monique Teillaud has been involved in the CGAL project since the end of the 90's. She has co-authored several packages in the library. Her research has focused on computing triangulations in non-Euclidean spaces for more than ten years.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.

(Invited Talk) Quantifying Shape Using the Medial Axis

Erin Wolf Chambers[1]

1 **Saint Louis University, USA**
erin.chambers@slu.edu

Abstract

The medial axis plays a fundamental role in shape matching and analysis, but is widely known to be unstable to even small boundary perturbations. Methods for pruning the medial axis are usually guided by some measure of significance, with considerable work done for both 2- and 3-dimensional shapes. Such significance measures can be used for identifying salient features, and hence are useful for simplification, comparison, and alignment. In this talk, we will present theoretical insights and properties of commonly used significance measures, focusing on those in 2D and 3D that are both shape-revealing and topology-preserving, as well as being robust to noise on the boundary. We'll also discuss more recent work in progress on using such measures to de-noise a shape and identify topologically and geometrically prominent features. Finally, we will cover several applications of these measures and techniques to real-world data sets.

Biography

Dr. Erin Wolf Chambers is a Professor at Saint Louis University in the Department of Computer Science, with a secondary appointment in the Department of Mathematics. Her research focus is on computational topology and geometry, with a more general interest in combinatorics and combinatorial algorithms. Complementing this work, she is also active in research projects to support and improve the culture and climate in computer science and mathematics, as well as to try to improve broader STEM educational experiences at all levels. She serves on the Computational Geometry Steering Committee and the Women in Computational Topology Steering Committee, as well as being an editor for Journal of Computational Geometry and for the Journal of Applied and Computational Topology. She received her PhD in Computer Science from the University of Illinois at Urbana-Champaign in 2008, and was a Visiting Research Professor at Saarland University in summer 2011.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.

Expected Complexity of Routing in Θ_6 and Half-Θ_6 Graphs*

Prosenjit Bose[1], Jean-Lou De Carufel[2], and Olivier Devillers[3]

1 **Carleton University, Ottawa, Canada** jit@scs.carleton.ca
2 **University of Ottawa, Ottawa, Canada** jdecaruf@uottawa.ca
3 **Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France**
Olivier.Devillers@inria.fr

Abstract

We study online routing algorithms on the Θ_6-graph and the half-Θ_6-graph (which is equivalent to a variant of the Delaunay triangulation). Given a source vertex s and a target vertex t in the Θ_6-graph (resp. half-Θ_6-graph), there exists a deterministic online routing algorithm that finds a path from s to t whose length is at most $2\|st\|$ (resp. $2.89\|st\|$) which is optimal in the worst case [Bose *et al.*, SIAM J. on Computing, 44(6)]. We propose alternative, slightly simpler routing algorithms that are optimal in the worst case and for which we provide an analysis of the average routing ratio for the Θ_6-graph and half-Θ_6-graph defined on a Poisson point process.

1 Introduction

The half-Θ_6-graph or TD-Delaunay (Triangular-Distance Delaunay [1]) is the Delaunay triangulation for the convex metric whose disk has the shape of an equilateral triangle. The Θ_6-graph gathers the two half-Θ_6-graphs corresponding to two symmetric equilateral triangles. Given such a graph, one may be interested in its spanning ratio, that is the worse ratio between the length of a shortest path in the graph and the Euclidean length [6, 12], algorithms to compute paths [11, 10, 9, 8] knowing the whole graph, or routing algorithms that uses only local knowledge of the graph [5].

This paper has two main contributions. The first contribution consists of the design of two new algorithms for routing in the half-Θ_6-graph in the so called *negative-routing* case. Our new routing algorithms come in two flavors: one is memoryless and the other uses a constant amount of memory. These new negative-routing algorithms have a worst-case optimal routing ratio but are simpler and more amenable to probabilistic analysis than the known optimal routing algorithm [4]. We also provide a new point of view on routing [4] in the half-Θ_6-graph in the *positive-routing* case.

The second contribution is the analysis of the two new negative-routing algorithms and of the positive-routing algorithm in a random setting, namely when the vertex set of the Θ_6-graph and half-Θ_6-graph is a point set that comes from an infinite Poisson point process X of intensity λ. The analysis is asymptotic with λ going to infinity, and gives the expected length of the shortest path between two fixed points s and t at distance one. Our results depend on the position of t with respect to s. We express our results both by taking the worst position for t and by averaging over all possible positions for t.

* This work has been supported by INRIA Associated team TRIP, NSERC and ANR Aspag (ANR-17-CE40-0017).Full version: [3]

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

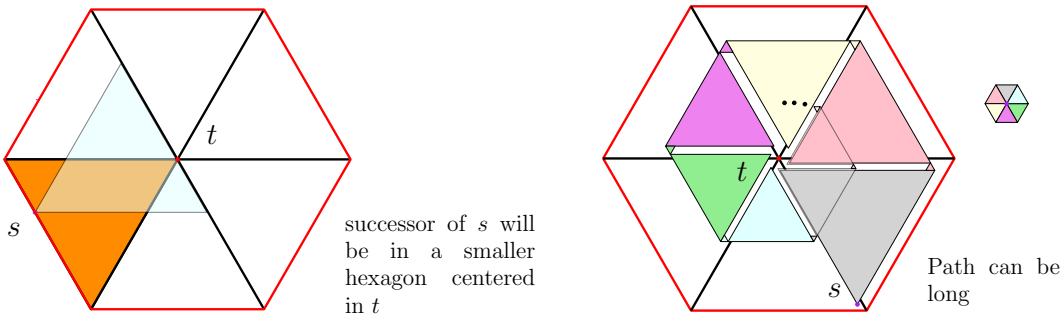


Figure 1 Naive Θ_6-routing terminates but its stretch is unbounded

2 Routing in Θ_6-graph and Half-Θ_6-graph

Let a *cone* be the region in the plane between two rays originating from the same point, referred to as the apex of the cone. The Θ_6-graph is formally defined as follows. For each point p, we split the plane around p into six cones defined by rays emanating from p making an angle of 0, $\frac{\pi}{3}$, $\frac{2\pi}{3}$, π, $\frac{4\pi}{3}$, and $\frac{5\pi}{3}$ with the horizontal axis. The cone whose bisector is an upward vertical ray emanating from p is labeled C_0^p, and $C_1^p \dots C_5^p$ are labeled in counterclockwise order. Given two vertices p and q with q in C_i^p, define the *canonical triangle* T_{pq} to be the equilateral triangle formed by the intersection of C_i^p and the half-plane that contains p, has q on its boundary, and whose boundary is a line perpendicular to the bisector of C_i^p. We call a canonical triangle T_{pq} *even* if q is in C_i^p for even i and *odd* otherwise. An edge exists in the Θ_6-graph between two vertices p and q if T_{pq} is empty. The half-Θ_6-graph uses the same rays to define the cone boundaries except only half the cones are used to define edges, namely only the even cones or only the odd cones. The even half-Θ_6-graph is defined using the neighbors of p in cones C_0^p, C_2^p, and C_4^p (Fig. 2 illustrates an even half-Θ_6-graph). The even (resp. odd) cones in the even half-Θ_6-graph are called positive (resp. negative) and symmetrically for the odd half-Θ_6-graph.

To route from a vertex s to a vertex t, a simple naive routing algorithm in Θ_6-graph consists of choosing as successor for s the one in the cone C_i^s containing t and then to iterate. This procedure always terminates but may have an unbounded routing-ratio (Fig. 1).

For the even half-Θ_6-graph Chew [7] provides a routing algorithm with a routing ratio of 2 when t is in a cone C_i^s with i even. As a consequence, the stretch ratio of the half-Θ_6-graph is 2 using the routing from t to s when i is odd. Bose *et al.* [4] address the routing problem with i odd and provide an algorithm with routing ratio $\frac{5}{\sqrt{3}} \simeq 2.89$ which is optimal for any constant-memory online routing algorithm [4]. Bonichon and Marckert [2] analyze the naive Θ_6-routing for Poisson distributed point sets.

3 Two Basic Routing Building Blocks on the Half-Θ_6-graph

We introduce *forward routing* and *side routing* two routing modes on the half-Θ_6-graph which serve as building blocks for our routing algorithms that have optimal worst-case behaviour. We consider the even half-Θ_6-graph and for ease of reference, we color the cones C_0, C_2 and C_4 blue, red, and green respectively.

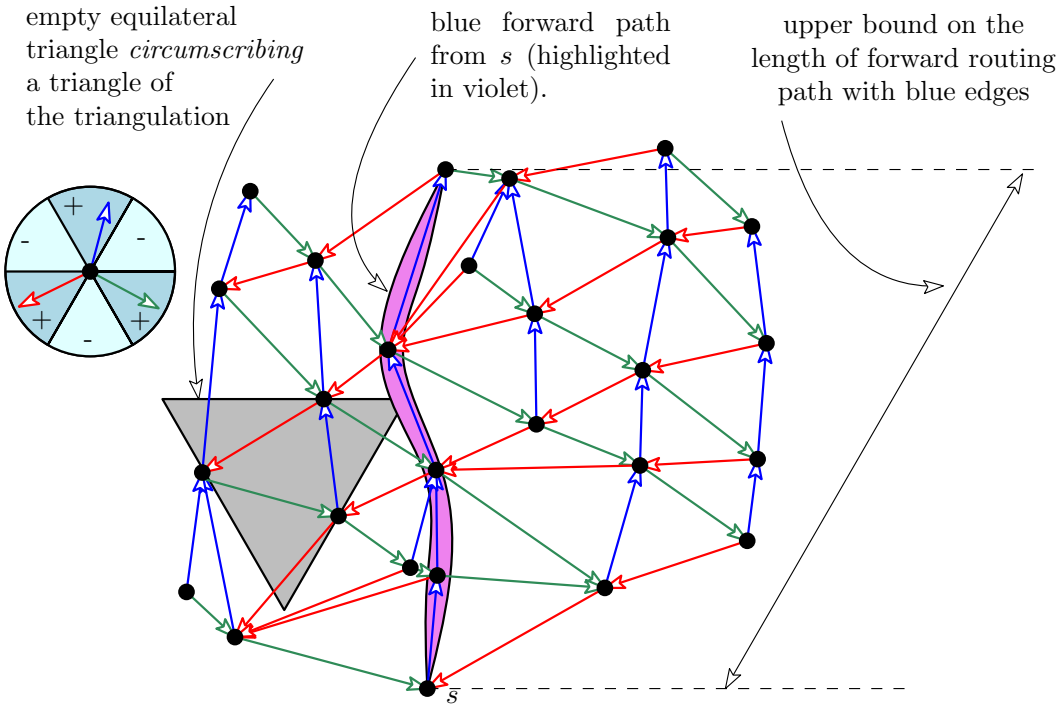


Figure 2 A *TD*-Delaunay triangulation (half-Θ_6-graph).

3.1 Forward-Routing Phase

Forward-routing consists of only following edges defined by a specific type of cone (i.e., a cone with the same color) until some specified stopping condition is met. For example, suppose the specific cone selected for forward routing is the blue cone. Thus, when forward-routing is invoked at a vertex x, the edge followed is xy where y is the vertex adjacent to x in x's blue cone. If the stopping condition is not met at y, then the next edge followed is yz where z is the vertex adjacent to y in y's blue cone. This process continues until a specified stopping condition is met. A path produced by forward routing consists of edges of the same color since edges are selected from one specific cone as illustrated in Fig. 2.

▶ **Lemma 3.1.** *Suppose that forward-routing is invoked at a vertex s and ends at a vertex t. The length of the path from s to t produced by forward-routing is at most the length of one side of the canonical triangle T_{st} which is $\frac{2}{\sqrt{3}}$ times the length of the orthogonal projection of st onto the bisector of C_0^s.*

Proof. This result follows from the fact that each edge along the path makes a maximum angle of $\frac{\pi}{6}$ with the cone bisector and the path is monotone in the direction of this bisector. ◀

3.2 Side-Routing Phase in the Half-Θ_6-graph

The *side-routing phase* is defined on the half-Θ_6-graph by using the fact that it is the *TD*-Delaunay triangulation, and thus planar. Consider a line ℓ parallel to one of the cone sides. W.l.o.g., we will assume ℓ is horizontal. We call the side of the line that bounds even cones the *positive side* of ℓ. (For a horizontal line, the positive side is below ℓ, for the lines with slopes $-\sqrt{3}$ and $\sqrt{3}$, respectively, the positive side is above the line.) Let s and t be two

vertices below ℓ and Δ_1, $\Delta_2, \ldots \Delta_j$ be an ordered sequence of consecutive triangles of the *TD*-Delaunay triangulation intersecting ℓ such that s is the bottom-left vertex of Δ_1, and t is the bottom-right vertex of Δ_j. Note that B is a path in the half-Θ_6-graph. Side-routing invoked at vertex s along ℓ stopping at t consists of walking from s to t along B (Fig. 3 for an example).

▶ **Lemma 3.2.** *Side-routing on the positive side of a line ℓ parallel to a cone boundary invoked at a vertex s and stopped at a vertex t in the half-Θ_6-graph results in a path whose length is bounded by twice the length of the orthogonal projection of st on ℓ. This path only uses edges of two colors and all vertices of the path have their successor of the third color on the other side of ℓ.*

Proof. W.l.o.g., assume ℓ is horizontal and the positive side is below ℓ. Consider the triangles Δ_i, $1 \leq i \leq j$ as defined above. The empty equilateral triangle ∇_i circumbscribing Δ_i has a vertex of Δ_i on each of its side by construction (∇_i are shown in grey in Fig. 3). If Δ_i has an edge of the path (i.e., below ℓ) then the vertex on the horizontal side of ∇_i is above the line while the two others are below. Thus, such an edge of the path goes from the left to the right side of ∇_i. Based on the slopes of the edges of ∇_i, we have the following:
–a– Each edge on the path has a length smaller than twice its horizontal projection. Therefore, summing the lengths of all the projections of the edges gives the claimed bound on the length.
–b– If the slope is negative (resp. positive), the path edge is green (resp. red).
–c– The blue successor of a vertex u on the lower sides of ∇_i is above ℓ since the part of C_o^u below ℓ is inside ∇_i and thus contains no other points. Blue edges are not on the path. ◀

3.3 Positive routing in the Half-Θ_6-graph (and the Θ_6-graph)

If t is in a positive cone of s, Bose et al. [4] (similar to Chew's algorithm) proposed a routing algorithm in the half-Θ_6-graph which they called *positive routing*. This algorithm can be rephrased in two phases: a *forward-routing* phase and a *side-routing* phase. The forward-routing phase is invoked with source s and destination t. It produces a path from s to the first vertex u outside the negative cone of t that contains s. The side-routing phase is invoked with source u and destination t and finds a path along the boundary of this negative cone. (Fig. 4-left). The stretch of such a path is proven to smaller than 2 [4]

4 Alternative Negative Routing Algorithms in the Half-Θ_6-graph

In this section, we outline two alternatives to the negative routing algorithm described by Bose et al. [4]. Our algorithms are a little simpler to describe, have the same worst-case routing ratio, and are easier to analyze in the random setting. The lower bound of $\frac{5}{\sqrt{3}} \simeq 2.89$ [4] applies to our alternative negative routing algorithms.

4.1 Memoryless Routing

Case 1. If t is in the positive cone C_i^s, take one step of forward-routing towards t

Case 2. If t is in the negative cone C_i^s and the successor u of s in C_{i-1}^s is outside T_{ts} (red triangle empty in Fig. 4-right), take one step of side-routing along the side of T_{ts} crossed by su.

Case 3. If t is in the negative cone C_i^s and the successor u of s in C_{i+1}^s is outside T_{ts} (green triangle empty in Fig. 4-right), take one step of side-routing along the side of T_{ts} crossed by su.

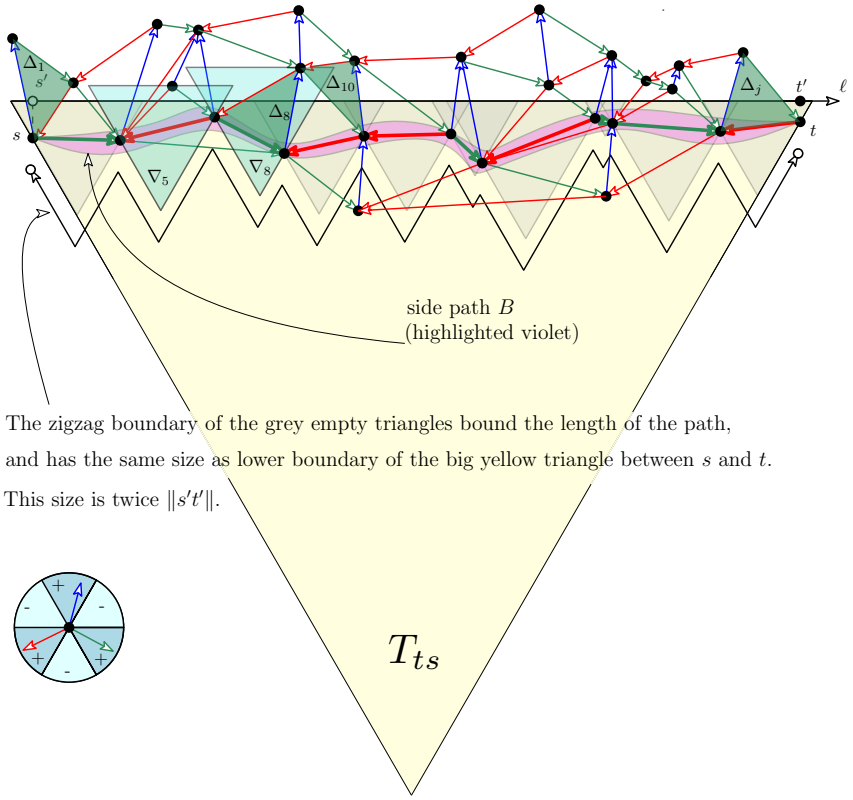


Figure 3 A side path below the horizontal line ℓ.

Case 4. If t is in the negative cone C_i^s and both successors of s in C_{i-1}^s and C_{i+1}^s are inside T_{ts} (green and red triangle non empty in Fig. 4-right), take one step of forward-routing in the direction of the side of T_{ts} incident to t closest to s (go to the green successor of s in Fig. 4-right).

Beyond the presentation, our strategy differs from the one of Bose et al. in Case 4 where Bose et al. follows a blue edge if one exists. We remark that, when we reach Case 3, we enter a side-routing phase that will continue until t is reached since a side-routing step ensures that at the next iteration side-routing will also be applicable. The same argument holds in Case 2, unless we reach a point s with both successors outside T_{ts} in which case we follow the other side of T_{ts}. To summarize, if t is in a positive cone of s, this routing algorithm will produce the path described at Section 3.3. If t is in a negative cone of s we use a forward phase in the green triangle, until we reach a vertex u whose edge in the green triangle intersects T_{ts} (recall that we assume that the green triangle is the smaller one). At this point, we invoke side-routing from u to t along the boundary of T_{ts}.

▶ **Lemma 4.1.** *Memoryless negative routing has a worst-case routing ratio of* $\frac{5}{\sqrt{3}} \simeq 2.89$.

Proof. Assume w.l.o.g. $s \in C_0^t$. Referring to Fig. 5-left, let w be the upper right vertex of T_{ts}, v be the orthogonal projection of u on tw and x its projection parallel to tw on sw. By Lemma 3.1, the path from s to u has length bounded by $\|sx\|$ and by Lemma 3.2, the path from u to t has length bounded by $2\|vt\|$. Combining the two paths, the length is bounded by $\|sx\| + 2\|vt\| \leq \|sw\| + 2\|wt\|$. Thus the stretch is smaller than $\frac{\|sw\|+2\|wt\|}{\|st\|}$.

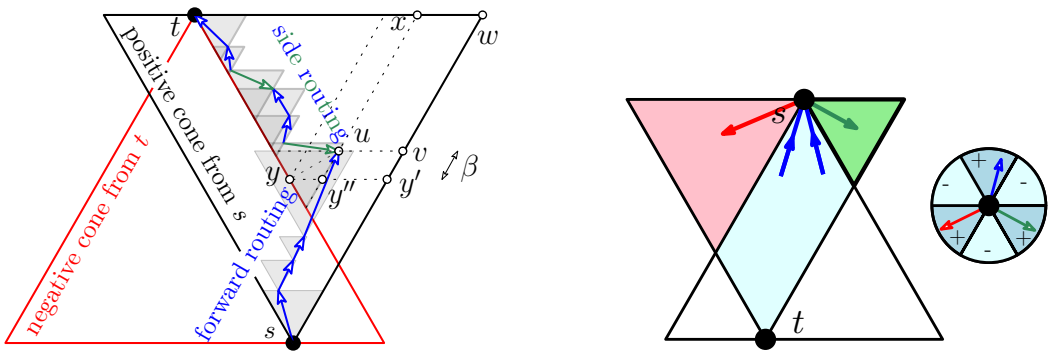


Figure 4 Positive and negative routing schemes [4].

Defining $\xi = \frac{\|wt\|}{\|sw\|}$ the stretch can be expressed as a function $\xi \rightsquigarrow \frac{2+\xi}{\sqrt{\frac{3}{4}+\left(\xi-\frac{1}{2}\right)^2}}$. It attains its maximum value of $\frac{5}{\sqrt{3}}$ when $\xi = \frac{1}{2}$ coresponding to s and t lie on a vertical line. ◀

4.2 Constant-Memory Negative Routing

We propose a second negative routing algorithm that has the same worst-case routing ratio, but we will prove that it has a better average routing ratio. However, it is no longer memoryless since it needs to remember the coordinates of one vertex, namely the source of the path. Let x'' be the intersection between T_{ts} and T_{st} closest to s. (Fig. 5-right). The idea is to use side-routing from s along sx'' and, just before exiting the green triangle, apply side-routing along $x''t$. This routing algorithm is identical to the one in the previous subsection, except that we replace Case 4 with the following, where u is the current vertex and s is the origin of the path whose coordinares are kept in memory:

Case 4' If t is in the negative cone C_i^u and both successors of u in C_{i-1}^u and C_{i+1}^u are inside T_{tu} (green and red triangle non empty): take one step of side-routing along the line sx''.

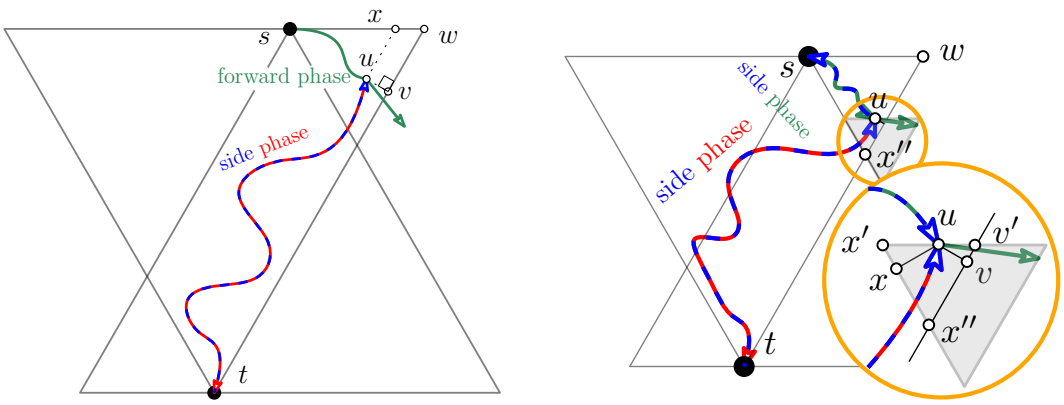


Figure 5 For Lemmas 4.1 and 4.2

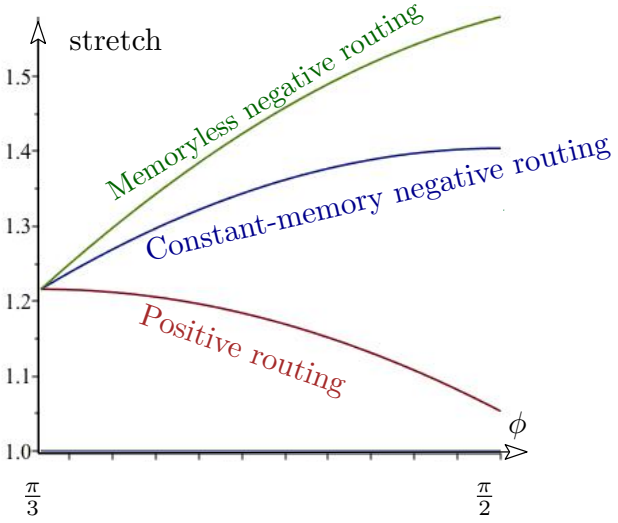
▶ **Lemma 4.2.** *Constant-memory negative routing has a worst-case routing ratio of* $\frac{5}{\sqrt{3}} \simeq 2.89$.

Proof. Assume w.l.o.g. $s \in C_0^t$. Referring to the Fig. 5-right, let x' and x be the horizontal and orthogonal projections of u on T_{st}, respectively, and v' and v be the horizontal and orthogonal projections of u on T_{ts}, respectively. By Lemma 3.2, the path from s to u has length bounded by $2\|sx\|$ and by Lemma 3.2 again, the path from u to t has length bounded by $2\|vt\|$. Combining the two paths the length is bounded by $2\|sx\| + 2\|vt\| \leq 2\|sx'\|+2\|x'x\|+2\|v't\| = 2\|wt\|+2\|xx'\|$. Since x is the orthogonal projection of u on the side $x'x''$ of the equilateral triangle $x'x''v'$, $\|xx'\|$ is smaller than the half side of the triangle $x'x''v'$ and we get a bound on the length of $2\|wt\|+2\|xx'\| \leq 2\|wt\|+2\frac{1}{2}\|x'v'\| \leq 2\|wt\|+\|sw\|$. ◀

5 Probabilistic Analysis

▶ **Theorem 5.1.** *Let X be a Poisson point process of intensity λ, s and t two points at unit distance and ϕ the angle of st with the horizontal axis. The the limits of the expected routing ratios of the different routing algorithms on the half-Θ_6-graph defined on $X \cup \{s,t\}$, as λ tends to ∞ are given in the following table and graph (with $\tau_1 := \frac{1}{4\sqrt{3}}(3\ln 3 + 4)$):*

Routing	$\mathbb{E}$ [routing ratio] (ϕ)	$\max_{s,t} \mathbb{E}$ [routing ratio]	$\mathbb{E}_{s,t}[\mathbb{E}$ [routing ratio]]
Positive routing	$\tau_1\left(\sin\phi + \frac{1}{\sqrt{3}}\cos\phi\right)$	$\frac{2}{\sqrt{3}}\tau_1 \simeq 1.2160$	$\frac{2\sqrt{3}}{\pi}\tau_1 \simeq 1.1612$
Constant-memory	$\frac{4}{3}\tau_1 \sin\phi$	$\frac{4}{3}\tau_1 \simeq 1.4041$	$\frac{4}{\pi}\tau_1 \simeq 1.3408$
Memoryless	$\tau_1\left(\frac{3}{2}\sin\phi - \frac{\sqrt{3}}{6}\cos\phi\right)$	$\frac{3}{2}\tau_1 \simeq 1.5800$	$\frac{6-\sqrt{3}}{\pi}\tau_1 \simeq 1.4306$



Proof. See full paper [3] ◀

References

1 Nicolas Bonichon, Cyril Gavoille, Nicolas Hanusse, and David Ilcinkas. Connections between Theta-graphs, Delaunay triangulations, and orthogonal surfaces. In *Proceedings of the 36th International Conference on Graph Theoretic Concepts in Computer Science (WG 2010)*, pages 266–278, 2010.

2 Nicolas Bonichon and Jean-François Marckert. Asymptotics of geometrical navigation on a random set of points in the plane. *Advances in Applied Probability*, 43(4):899–942, 2011. doi:10.1239/aap/1324045692.

3 Prosenjit Bose, Jean-Lou De Carufel, and Olivier Devillers. Expected Complexity of Routing in Θ_6 and Half-Θ_6 Graphs. Research report, INRIA, 2019. URL: https://hal.inria.fr/hal-02338733.

4 Prosenjit Bose, Rolf Fagerberg, André van Renssen, and Sander Verdonschot. Optimal local routing on Delaunay triangulations defined by empty equilateral triangles. *SIAM Journal on Computing*, 44(6):1626–1649, 2015. doi:10.1137/140988103.

5 Prosenjit Bose and Pat Morin. Online routing in triangulations. *SIAM Journal on Computing*, 33(4):937–951, 2004.

6 Prosenjit Bose and Michiel Smid. On plane geometric spanners: A survey and open problems. *Computational Geometry: Theory and Applications*, 46(7):818–830, 2013.

7 Paul Chew. There are planar graphs almost as good as the complete graph. *Journal of Computer and System Sciences*, 39(2):205–219, 1989.

8 E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

9 J. E. Hopcroft and R. E. Tarjan. Algorithm 447: Efficient algorithms for graph manipulation. *Communications of the ACM*, 16(6):372–378, 1973.

10 C. Y. Lee. An algorithm for path connection and its applications. *IRE Transaction on Electronic Computers*, EC-10(3):346–365, 1961.

11 E. F. Moore. The shortest path through a maze. In *Proceedings of the International Symposium on the Theory of Switching*, pages 285–292, 1959.

12 G. Narasimhan and M. Smid. *Geometric Spanner Networks*. Cambridge University Press, 2007.

Fréchet Distance Between Uncertain Trajectories: Computing Expected Value and Upper Bound*

Kevin Buchin[1], Maarten Löffler[2], Aleksandr Popov[†1], and Marcel Roeloffzen[‡1]

1 **Department of Mathematics and Computer Science, TU Eindhoven, Netherlands**
{k.a.buchin, a.popov, m.j.m.roeloffzen}@tue.nl

2 **Department of Information and Computing Sciences, Utrecht University, Netherlands**
m.loffler@uu.nl

Abstract

A trajectory is a sequence of time-stamped locations. Measurement uncertainty is an important factor to consider when analysing trajectory data. We define an uncertain trajectory as a trajectory where at each time stamp the true location lies within an uncertainty region—a disk, a line segment, or a set of points. In this paper we consider discrete and continuous Fréchet distance between uncertain trajectories.

We show that finding the largest possible discrete or continuous Fréchet distance among all possible realisations of two uncertain trajectories is NP-hard under all the uncertainty models we consider. Furthermore, computing the expected discrete or continuous Fréchet distance is #P-hard when the uncertainty regions are modelled as point sets or line segments. We also study the setting with time bands, where we restrict temporal alignment of the two trajectories, and give polynomial-time algorithms for largest possible and expected discrete and continuous Fréchet distance for uncertainty regions modelled as point sets.

1 Introduction

Trajectory data is ubiquitous. Whether tracking animals or dissecting a football game, we need to deal with automated analysis of measured trajectories. However, most existing approaches do not take into account the inherent uncertainty that arises due to the measurement procedure. In some settings this uncertainty is small on the scale of the analysis; in other settings, however, meaningful results can only be obtained when dealing with such uncertainty explicitly. In this paper, we aim to do that for a variety of uncertainty models when computing Fréchet distance and discrete Fréchet distance.

There are many results on trajectory analysis: on simplification of trajectories [1, 14, 21, 22, 29]; on trajectory segmentation [3, 4, 6]; on clustering trajectories [8, 17]. There are also many approaches to trajectory similarity [13, 25, 31], including (discrete) Fréchet distance [5, 16, 20] and variants [15]. There is some work tackling uncertainty in computational geometry [12, 24, 26, 27], including problems on moving points [10, 18].

Some authors suggest computing restricted versions of Fréchet distance and other distance metrics using time bands [7, 23, 30], restricting the alignment of trajectories. This is mostly useful when the trajectories are regularly sampled and are expected to be aligned in time, so we can use some fixed-size band on indices of the trajectory points as proxy for timestamps.

* This abstract presents partial results from joint work with Chenglin Fan and Benjamin Raichel [9].
† Supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 612.001.801.
‡ Supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 628.011.005.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Table 1 Summary of hardness results for the decision problems in this paper.

		indecisive	imprecise	
			disks	line segments
discrete FD	UB	NP-complete	NP-complete	NP-complete
	Exp	#P-hard	—	#P-hard
FD	UB	NP-complete	NP-complete	NP-complete
	Exp	#P-hard	—	—

There is some work on similarity of uncertain trajectories. Buchin and Sijben [11] study computing discrete Fréchet distance on uncertain trajectories with points defined by probability distributions. Ahn et al. [2] model each uncertain point by a disk, and the real location of a point may be any point in the disk. They compute the lowest possible discrete Fréchet distance using a dynamic programming approach. They also stipulate that finding largest possible Fréchet distance is hard; it is confirmed by Fan and Zhu [19] for the case of thin rectangles as imprecision model and is further explored in this paper.

We focus on Fréchet distance and discrete Fréchet distance. We make a distinction between *indecisive* points and *imprecise* points for location uncertainty, as explained in Section 1.1. We only model measurement uncertainty, so we assume linear motion on a straight line segment between two consecutive measurements. We consider *upper bound* Fréchet distance and *expected* Fréchet distance between trajectories, which correspond to the largest possible and expected Fréchet distance over every possible combination of real locations of the trajectory. Our contributions are:

1. NP-hardness and #P-hardness results.[1] We show NP-hardness for the upper bound on (discrete) Fréchet distance using simpler uncertainty regions and a simpler construction than Fan and Zhu [19]. We show #P-hardness for the expected value of (discrete) Fréchet distance in several settings. See Table 1 for details.
2. Algorithms for discrete and continuous Fréchet distance with Sakoe–Chiba time bands. Previous results suggest that there is little room for positive algorithmic results. If the trajectories are regularly sampled, or can be resampled appropriately at will, and are expected to align in time, we can restrict the computation to a fixed-width time window on indices of trajectory points, as explained in Section 3. We give algorithms to find, given *indecisive* trajectories, the upper bound and expected (discrete) Fréchet distance when constrained to Sakoe–Chiba bands of fixed width [30].

The results of this abstract are discussed further in the master thesis of A. Popov [28] and in joint work with C. Fan and B. Raichel [9]. In the latter paper, we additionally investigate the lower bound (continuous) Fréchet distance.

1.1 Notation

We denote a *polygonal curve* of length n on n points in d dimensions as $P = \langle p_1, p_2, \ldots, p_n \rangle$. A *trajectory* is a polygonal curve with timestamps associated to each point of the curve. Whenever timestamps are not relevant, we use the terms interchangeably. We denote a *subtrajectory* from point i to j of curve P as $P[i : j]$.

[1] Hardness class #P is a class of counting problems related to NP. For example, SAT ('Is there a satisfying assignment to a boolean formula?') is an NP-complete problem, whereas #SAT ('How many satisfying assignments to a boolean formula are there?') is a #P-complete problem.

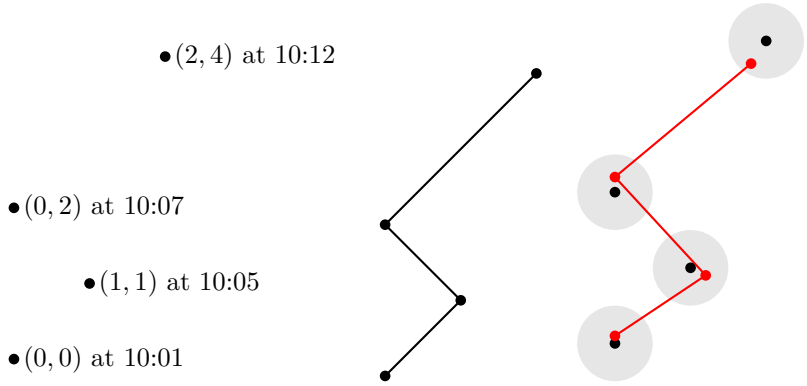


Figure 1 Left: Trajectory data. Centre: Polygonal curve on the data. Right: Imprecise trajectory with disks as imprecision regions and the real trajectory.

An *uncertain* point is commonly represented as a compact region $H \subset \mathbb{R}^d$. A *realisation* of such a point h is one of the points from the region H. An *indecisive* point is a special case of an uncertain point: it is a set of points $H = \{h_1, \ldots, h_k\}$. Similarly, an *imprecise* point is a compact connected region $H \subset \mathbb{R}^d$. We use disks or line segments as such regions. Note that a precise point is a special case of an indecisive point and an imprecise point.

Consider a sequence of uncertain points $\mathcal{H} = \langle H_1, \ldots, H_n \rangle$, referred to as an *uncertain trajectory*. A *realisation* $P \in \mathcal{H}$ of an uncertain trajectory is a polygonal curve $P = \langle p_1, \ldots, p_n \rangle$, where each p_i is a realisation of the corresponding uncertain point H_i. The concept of uncertain trajectories is illustrated in Figure 1.

Extending the notation to uncertain trajectories $\mathcal{H}$ and $\mathcal{V}$, we define the upper bound on the (discrete) Fréchet distance under different possible realisations:

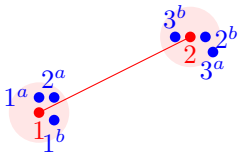
$$d_{\mathrm{dF}}^{\max}(\mathcal{H}, \mathcal{V}) = \max_{A \in \mathcal{H}, B \in \mathcal{V}} d_{\mathrm{dF}}(A, B)\,, \qquad d_{\mathrm{F}}^{\max}(\mathcal{H}, \mathcal{V}) = \max_{A \in \mathcal{H}, B \in \mathcal{V}} d_{\mathrm{F}}(A, B)\,.$$

We define *expected Fréchet distance* $d_{\mathrm{dF}}^{\mathbb{E}}$ and $d_{\mathrm{F}}^{\mathbb{E}}$ as the expected value of the Fréchet distance if the realisations are picked uniformly at random, independently for each trajectory point.

2 Hardness Results

We do not discuss the construction; see the master thesis for full proofs [28]. It is possible to provide a reduction from CNF-SAT to the decision problem for finding $d_{\mathrm{dF}}^{\max}$ and $d_{\mathrm{F}}^{\max}$ under different uncertainty models, establishing their NP-hardness. Furthermore, it is possible to provide reductions from the counting version of CNF-SAT to the decision problem for finding $d_{\mathrm{dF}}^{\mathbb{E}}$ and $d_{\mathrm{F}}^{\mathbb{E}}$ in some settings, establishing their #P-hardness. The construction has two trajectories, one precise and one uncertain; every realisation of the uncertain trajectory corresponds to a variable assignment in the CNF-SAT formula. We get two possible values of Fréchet distance for each realisation and can distinguish satisfying assignments. Then $d_{\mathrm{dF}}^{\max}$ tells us if the formula is satisfiable, and $d_{\mathrm{dF}}^{\mathbb{E}}$ gives us the count of satisfying assignments.

The proofs using our construction extend to other compact uncertainty regions of the same shape and size for the discrete Fréchet distance; the extension for the continuous Fréchet distance seems possible, but is less obvious. The expected case is a lot more difficult due to the complicated integral evaluations, so even for disks the results seem difficult to obtain. The list of settings we consider is shown in Table 1.



F F	F T	T T
T T	T F	F F

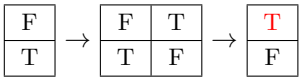


Figure 2 Left: An indecisive and a precise trajectory. Middle: Distance matrix. 'T T' in the bottom left cell means $\|1-1^a\| \leq \varepsilon$ and $\|1-1^b\| \leq \varepsilon$. Right: Computing reachability matrix, column by column. Note the two reachability vectors for the second column.

3 Algorithms with Time Bands

Here we use the *Sakoe–Chiba band,* which restricts aligning point k on one trajectory to points $k \pm w$ on the other trajectory, for all k and some fixed w [30]. In some settings the point indices act as proxy for timestamps, and the trajectories are expected to be aligned in time, so this restriction is reasonable. We develop polynomial-time algorithms for the restricted hard problems of the previous section on *indecisive* points.

3.1 Upper Bound Discrete Fréchet Distance

First of all, let us discuss a simple setting. Suppose we are given a trajectory $V = \langle q_1, \ldots, q_n \rangle$ of n precise points and $\mathcal{H} = \langle P_1, \ldots, P_n \rangle$ of n indecisive points, each of them having ℓ options, so for all $i \in \{1, \ldots, n\}$ we have $P_i = \{p_i^1, \ldots, p_i^\ell\}$. We would like to answer the following decision problem: *'If we restrict the couplings to a Sakoe–Chiba band of width w, is it true that $d_{\mathrm{dF}}^{\max}(\mathcal{H}, V) \leq \varepsilon$ for some given threshold $\varepsilon > 0$?'* We want to solve the decision problem for the upper bound discrete Fréchet distance between a precise and an indecisive trajectory.

In a fully precise setting the discrete Fréchet distance can be computed using dynamic programming [16]. We create a table where the rows correspond to the vertices of one trajectory, say V, and columns correspond to the vertices of the other trajectory, say H. Each table entry (i, j) then contains a TRUE or FALSE value indicating if there is a coupling between $V[1:j]$ and $H[1:i]$ with maximum distance at most ε. We use a similar approach.

Suppose we position $\mathcal{H}$ to go horizontally along the table, and V to go vertically. Consider an arbitrary column in the table and suppose that we fix the realisation of a part of $\mathcal{H}$ up to the previous column. Then we can simply consider the new column ℓ times, each time picking a different realisation for the new point on $\mathcal{H}$, and compute the resulting reachability. As we do this for the entire column at once, we can ensure consistency of our choice of realisation of $\mathcal{H}$. This procedure will give us a set of binary reachability vectors for the new column, each vector corresponding to a realisation of a prefix of $\mathcal{H}$. The *reachability vector* is a boolean vector that, for the cell (i, j) of the table, states whether for a particular realisation A of $\mathcal{H}[1:i]$ the discrete Fréchet distance between A and $V[1:j]$ is below some threshold ε.

An important observation is that we do not need to distinguish between the realisations of trajectory prefixes that give the same reachability vector: once we start filling out the next column, all we care about is the existence of some realisation leading to that particular reachability vector. So, we can keep a *set* of binary vectors of reachability in the column.

This procedure was suggested for a specific realisation of a prefix of $\mathcal{H}$. However, we can also repeat this for each previous reachability vector, only keeping the unique results. As all the realisation choices happen along $\mathcal{H}$, by treating the table column-by-column we ensure that we do not have issues with inconsistent choices. Therefore, repeating this procedure n times, we will fill out the last column of the table. At that point, if any vector has FALSE in

the top right cell, then there is some realisation $A \Subset \mathcal{H}$ such that $d_{\mathrm{dF}}(A, V) > \varepsilon$, and hence $d_{\mathrm{dF}}^{\max}(\mathcal{H}, V) > \varepsilon$; otherwise, $d_{\mathrm{dF}}^{\max}(\mathcal{H}, V) \leq \varepsilon$, as there are no 'bad' realisations.

In more detail, we use two tables, distance matrix D and reachability matrix R. First of all, we initialise the distance matrix D and the reachability of the first column for all possible locations of H_1. Then we fill out R column-by-column. We take the reachability of the previous column and note that any cell can be reached either with the horizontal step or with the diagonal step. We need to consider various extensions of the trajectory $\mathcal{H}$ with one of the ℓ realisations of the current point: the distance matrix should allow the specific coupling. Furthermore, assume we find that a certain cell is reachable; if allowed by the distance matrix, we can then go upwards, marking cells above the current cell reachable, even if they are not directly reachable with a horizontal or diagonal step. Then we just remember the newly computed vector; we make sure to only remember distinct vectors.

We check if there is a realisation that yields FALSE in the last cell; then this realisation is chosen by the upper bound, yielding FALSE. The computation is illustrated in Figure 2.

We can extend this approach to the setting where both trajectories are indecisive, so instead of V we have $\mathcal{V} = \langle V_1, \ldots, V_n \rangle$, with, for each $j \in \{1, \ldots, n\}$, $V_j = \{q_j^1, \ldots, q_j^\ell\}$.

Suppose we pick a realisation for trajectory $\mathcal{V}$. Then we can apply the algorithm we just described. We cannot run it separately for every realisation of $\mathcal{V}$; instead, note that the part of the realisation that matters for column i is the points from $i - w$ to $i + w$, since any previous or further points are outside the time band. We can fix these $2w + 1$ points and compute the column as before; we do so for each possible combination on these $2w + 1$ points.

▶ **Theorem 1.** *Suppose we are given two indecisive trajectories of length n with ℓ options per indecisive point. Then we can compute the upper bound discrete Fréchet distance restricted to a Sakoe–Chiba band of width w in time $\Theta(4^w n \sqrt{w} \ell^{2w})$.*

3.2 Expected Discrete Fréchet Distance

To compute the expected discrete Fréchet distance with time bands, we need two observations:

1. For any two precise trajectories, there is a single threshold ε where the answer to the decision problem changes from TRUE to FALSE—a *critical value*. That threshold corresponds to the distance between some two points on the trajectories.
2. We can modify our algorithm to store associated counts with each reachability vector, obtaining the fraction of realisations that yield the answer TRUE for a given threshold ε.

So, we can execute our algorithm for each of the critical values and obtain the cumulative distribution function $\mathbb{P}(d_{\mathrm{dF}}(A, B) > \varepsilon)$ for $A, B \Subset \mathcal{H}, \mathcal{V}$ following the uniform distribution. Since the cumulative distribution function is a step function, we can compute $d_{\mathrm{dF}}^{\mathbb{E}}$.

▶ **Theorem 2.** *Suppose we are given two indecisive trajectories $\mathcal{H}$ and $\mathcal{V}$ of length n with ℓ options per indecisive point. Then we can compute the expected discrete Fréchet distance when constrained to a Sakoe–Chiba band of width w in time $\Theta(4^w n^2 w^2 \ell^{2w})$ in the worst case.*

3.3 Continuous Fréchet Distance

We can adapt our time band algorithms to handle the continuous Fréchet distance. Instead of the boolean reachability vectors, we use columns of *free space* cells, introduced by Alt and Godau [5, 20], as illustrated in Figure 3. We store the reachability intervals on cell borders.

The specifics of handling intervals are very technical and can be found in the master thesis [28]. The number of possible intervals is bounded; this way we get an algorithm that runs in time polynomial in n. An extension to find the expected value is also possible.

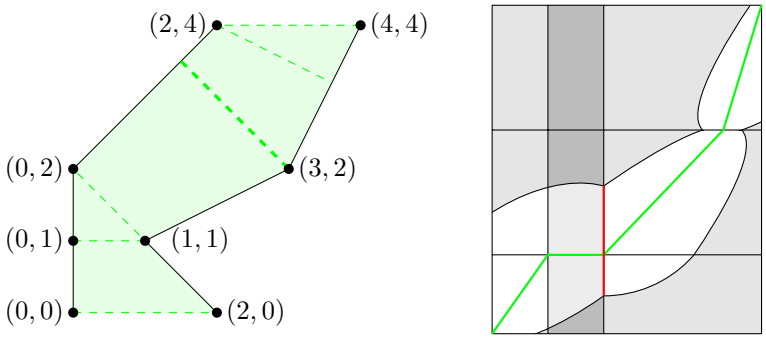


Figure 3 Left: Visualisation of Fréchet distance on precise trajectories. Right: Corresponding free-space diagram. The highlighted intervals are propagated per column in the uncertain case. The monotone path corresponds to the alignment depicted on the left.

▶ **Theorem 3.** *Suppose we are given two indecisive trajectories of length n with ℓ options per indecisive point. Then we can compute the upper bound Fréchet distance and the expected Fréchet distance restricted to a Sakoe–Chiba band of fixed width w in time polynomial in n.*

References

1. Pankaj K. Agarwal, Sariel Har-Peled, Nabil H. Mustafa, and Yusu Wang. Near-linear time approximation algorithms for curve simplification. *Algorithmica*, 42(3):203–219, July 2005. doi:10.1007/s00453-005-1165-y.
2. Hee-Kap Ahn, Christian Knauer, Marc Scherfenberg, Lena Schlipf, and Antoine Vigneron. Computing the discrete Fréchet distance with imprecise input. *International Journal of Computational Geometry & Applications*, 22(01):27–44, 2012. doi:10.1142/S0218195912600023.
3. Sander P. A. Alewijnse, Kevin Buchin, Maike Buchin, Andrea Kölzsch, Helmut Kruckenberg, and Michel A. Westenberg. A framework for trajectory segmentation by stable criteria. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '14, pages 351–360, New York, NY, USA, 2014. ACM. doi:10.1145/2666310.2666415.
4. Sander P. A. Alewijnse, Kevin Buchin, Maike Buchin, Stef Sijben, and Michel A. Westenberg. Model-based segmentation and classification of trajectories. *Algorithmica*, 80(8):2422–2452, August 2018. doi:10.1007/s00453-017-0329-x.
5. Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry and Applications*, 5(1):75–91, 1995. doi:10.1142/S0218195995000064.
6. Boris Aronov, Anne Driemel, Marc van Kreveld, Maarten Löffler, and Frank Staals. Segmentation of trajectories on non-monotone criteria. *ACM Transactions on Algorithms*, 12(2):26:1–26:28, December 2015. doi:10.1145/2660772.
7. Kevin Buchin, Maike Buchin, and Joachim Gudmundsson. Constrained free space diagrams: A tool for trajectory analysis. *International Journal of Geographical Information Science*, 24(7):1101–1125, July 2010. doi:10.1080/13658810903569598.
8. Kevin Buchin, Anne Driemel, Joachim Gudmundsson, Michael Horton, Irina Kostitsyna, Maarten Löffler, and Martijn Struijs. Approximating (k, ℓ)-center clustering for curves. In *Proceedings of the Thirtieth Annual ACM–SIAM Symposium on Discrete Algorithms*, pages 2922–2938, Philadelphia, PA, USA, 2019. SIAM. doi:10.1137/1.9781611975482.181.

9 Kevin Buchin, Chenglin Fan, Maarten Löffler, Aleksandr Popov, Benjamin Raichel, and Marcel Roeloffzen. Fréchet distance for uncertain curves. Unpublished manuscript, 2020.

10 Kevin Buchin, Stef Sijben, T. Jean Marie Arseneau, and Erik P. Willems. Detecting movement patterns using Brownian bridges. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '12, pages 119–128, New York, NY, USA, 2012. ACM. doi:10.1145/2424321.2424338.

11 Maike Buchin and Stef Sijben. Discrete Fréchet distance for uncertain points, 2016. Presented at EuroCG 2016, Lugano, Switzerland. URL: http://www.eurocg2016.usi.ch/sites/default/files/paper_72.pdf [cited 2019-07-10].

12 Leizhen Cai and Mark Keil. Computing visibility information in an inaccurate simple polygon. *International Journal of Computational Geometry & Applications*, 7:515–538, December 1997. doi:10.1142/S0218195997000326.

13 Lei Chen, M. Tamer Özsu, and Vincent Oria. Robust and fast similarity search for moving object trajectories. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, SIGMOD '05, pages 491–502, New York, NY, USA, 2005. ACM. doi:10.1145/1066157.1066213.

14 David H. Douglas and Thomas K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973. doi:10.3138/FM57-6770-U75U-7727.

15 Anne Driemel and Sariel Har-Peled. Jaywalking your dog: Computing the Fréchet distance with shortcuts. *SIAM Journal on Computing*, 42(5):1830–1866, October 2018. arXiv:1107.1720v4, doi:10.1137/120865112.

16 Thomas Eiter and Heikki Mannila. Computing discrete Fréchet distance. Technical Report CD-TR 94/64, Technishe Universität Wien, April 1994. URL: http://www.kr.tuwien.ac.at/staff/eiter/et-archive/cdtr9464.pdf [cited 2019-04-23].

17 Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 226–231, Menlo Park, CA, USA, 1996. AAAI Press. URL: https://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf [cited 2019-09-10].

18 William Evans, David Kirkpatrick, Maarten Löffler, and Frank Staals. Competitive query strategies for minimising the ply of the potential locations of moving points. In *Proceedings of the Twenty-Ninth Annual Symposium on Computational Geometry*, SoCG '13, pages 155–164, New York, NY, USA, 2013. ACM. doi:10.1145/2462356.2462395.

19 Chenglin Fan and Binhai Zhu. Complexity and algorithms for the discrete Fréchet distance upper bound with imprecise input, February 2018. arXiv:1509.02576v2.

20 Michael Godau. A natural metric for curves: Computing the distance for polygonal chains and approximation algorithms. In *STACS 91: Proceedings of 8th Annual Symposium on Theoretical Aspects of Computer Science*, number 480 in LNCS, pages 127–136, Berlin, Germany, 1991. Springer Berlin Heidelberg. doi:10.1007/BFb0020793.

21 Joachim Gudmundsson, Jyrki Katajainen, Damian Merrick, Cahya Ong, and Thomas Wolle. Compressing spatio-temporal trajectories. *Computational Geometry*, 42(9):825–841, November 2009. doi:10.1016/j.comgeo.2009.02.002.

22 Hiroshi Imai and Masao Iri. Computational-geometric methods for polygonal approximations of a curve. *Computer Vision, Graphics, and Image Processing*, 36(1):31–41, 1986. doi:10.1016/S0734-189X(86)80027-5.

23 Fumitada Itakura. Minimum prediction residual principle applied to speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 23(1):67–72, February 1975. doi:10.1109/TASSP.1975.1162641.

24 Christian Knauer, Maarten Löffler, Marc Scherfenberg, and Thomas Wolle. The directed Hausdorff distance between imprecise point sets. *Theoretical Computer Science*, 412(32):4173–4186, 2011. doi:10.1016/j.tcs.2011.01.039.

25 Joseph B. Kruskal and Mark Liberman. The symmetric time-warping problem: From continuous to discrete. In David Sankoff and Joseph B. Kruskal, editors, *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, chapter 4, pages 125–161. Addison–Wesley, Reading, MA, USA, 1983.

26 Maarten Löffler. *Data Imprecision in Computational Geometry.* PhD thesis, Universiteit Utrecht, October 2009. URL: https://dspace.library.uu.nl/bitstream/handle/1874/36022/loffler.pdf [cited 2019-06-15].

27 Maarten Löffler and Jeff M. Phillips. Shape fitting on point sets with probability distributions: ESA 2009. In *Algorithms*, number 5757 in LNCS, pages 313–324, Berlin, Germany, 2009. Springer Berlin Heidelberg. arXiv:0812.2967v1, doi:10.1007/978-3-642-04128-0_29.

28 Aleksandr Popov. Similarity of uncertain trajectories. Master's thesis, Eindhoven University of Technology, November 2019. URL: https://research.tue.nl/en/studentTheses/similarity-of-uncertain-trajectories [cited 2019-12-18].

29 Urs Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, 1(3):244–256, 1972. doi:10.1016/S0146-664X(72)80017-0.

30 Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, February 1978. doi:10.1109/TASSP.1978.1163055.

31 Michail Vlachos, George Kollios, and Dimitrios Gunopulos. Discovering similar multidimensional trajectories. In *Proceedings 18th International Conference on Data Engineering*, pages 673–684, Piscataway, NJ, USA, 2002. IEEE. doi:10.1109/ICDE.2002.994784.

Packing Squares into a Disk with Optimal Worst-Case Density

Sándor P. Fekete[1], Vijaykrishna Gurunathan[2], Kushagra Juneja[2], Phillip Keldenich[1], Linda Kleist[1], and Christian Scheffer[1]

1 **Department of Computer Science, TU Braunschweig, Germany**
{s.fektete, p.keldenich, l.kleist, c.scheffer}@tu-bs.de
2 **Department of Computer Science & Engineering, IIT Bombay, India**
krishnavijay1999@gmail.com, kuku12320@gmail.com

Abstract

We provide a tight result for a fundamental problem arising from packing squares into a circular container: The critical density of packing squares in a disk is $\delta = 8/5\pi \approx 0.509$. This implies that any set of (not necessarily equal) squares of total area $A \leq 8/5$ can always be packed into a unit disk; in contrast, for any $\varepsilon > 0$ there are sets of squares of area $8/5 + \varepsilon$ that cannot be packed. This settles the last case of packing circular or square objects into a circular or square container, as the critical densities for squares in a square ($1/2$), circles in a square ($\pi/3+\sqrt{2} \approx 0.539$) and circles in a circle ($1/2$) have already been established. The proof uses a careful manual analysis, complemented by a minor automatic part that is based on interval arithmetic. Beyond the basic mathematical importance, our result is also useful as a blackbox lemma for the analysis of recursive packing algorithms.

1 Introduction

Problems of geometric packing and covering arise in a wide range of natural applications. They also have a long history of spawning many extremely demanding (and often still unsolved) mathematical challenges. These difficulties are also notable from an algorithmic perspective, as relatively straightforward one-dimensional variants of packing and covering are already NP-hard; however, deciding whether a given set of one-dimensional segments can be packed into a given interval can be checked by computing their total length. This simple criterion is no longer available for two-dimensional, geometric packing or covering problems, for which the total volume often does not suffice to decide feasibility of a set, making it necessary to provide an explicit packing or covering.

We provide a provably optimal answer for a natural and previously unsolved case of *tight worst-case area bounds*, based on the notion of *critical packing density*: What is the largest number $\delta_p \leq 1$, such that any set S of squares with a total volume of at most δ_p can always be packed into a disk C of area 1, regardless of the individual sizes of the elements in S? We show that the correct answer is $\delta_p = 8/5\pi$: Any set of squares of total area at most $8/5$ can be packed into a unit disk (with radius 1), and for any value $A > 8/5$, there are sets that cannot be packed. This quantity is of mathematical importance, as it settles an open problem, as well as of algorithmic interest, because it provides a simple criterion for feasibility. It also settles the last remaining case of packing circular or square objects into a circular or square container, see Figure 1 for an overview.

1.1 Related Work

Problems of square packing have been studied for a long time. The decision problem whether it is possible to pack a given set of squares into the unit square was shown to be strongly

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

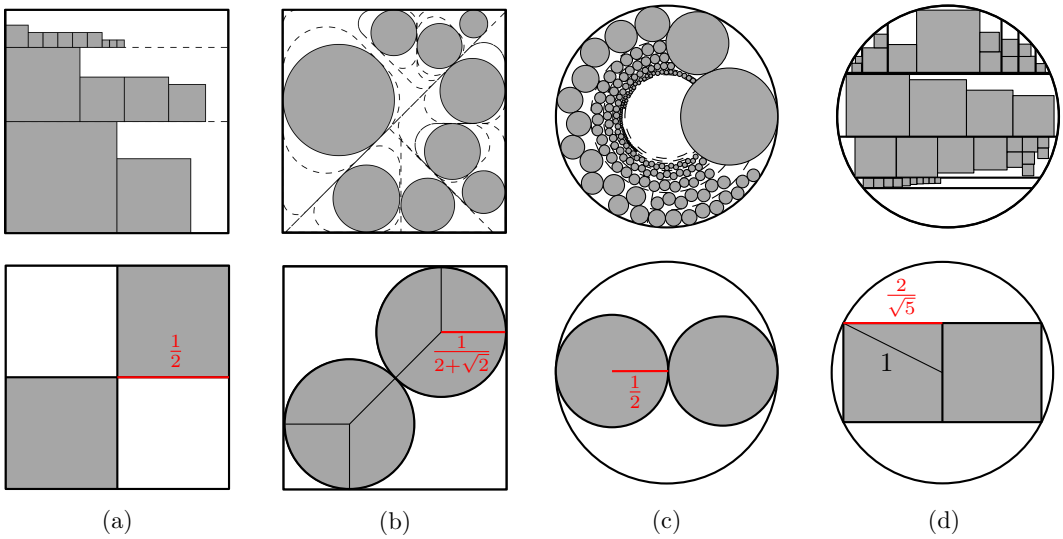


Figure 1 Illustration of the worst-case optimal approaches and worst case instances for packing (a) squares into a square with Shelf Packing by Moon and Moser [7]. (b) disks into a square by Fekete et al. [5]. (c) disks into a disk by Fekete et al. [4] (d) squares into a disk [this paper].

NP-complete by Leung et al. [6]. Already in 1967, Moon and Moser [7] proved that the critical packing density for squares into a square is $1/2$. As illustrated in Figure 1(a), this is best possible. Demaine, Fekete, and Lang [1] showed in 2010 that deciding whether a given set of disks can be packed into a unit square is NP-hard. Consequently, there is (most likely) no deterministic polynomial-time algorithm to decide whether a given set of disks can be packed into a given container. The problem of establishing the critical packing density for disks in a square was posed by Demaine, Fekete, and Lang [1] and resolved by Morr, Fekete and Scheffer [5, 8]. Using a recursive procedure for partitioning the container into triangular pieces, they proved that the critical packing density of disks in a square is $\pi/(3+2\sqrt{2})$. More recently, Fekete et al. [4] established the critical packing density of $1/2$ for packing disks into a disk by employing a number of algorithmic techniques in combination with interval arithmetic. Note that the main objective of this line of research is to compute tight worst-case bounds. For specific instances, a packing may still be possible, even if the density is higher; this also implies that proofs of infeasibility for specific instances may be trickier. However, the idea of using the total item volume for computing packing bounds can still be applied. See the work by Fekete and Schepers [2, 3], which shows how a *modified* volume for geometric objects can be computed, yielding good lower bounds for one- or higher-dimensional scenarios.

2 A Worst-Case Optimal Algorithm

The main result is a worst-case optimal algorithm for packing squares into a unit disk.

▶ **Theorem 2.1.** *Every set of squares with a total area of at most $8/5$ can be packed into a disk with radius 1. This is worst-case optimal, i.e., for every $\lambda > 8/5$ there exists a set of squares with a total area of λ that cannot be packed into the unit disk.*

A proof of Theorem 2.1 consists of (i) a class of instances that provide the upper bound of $8/5$ and (ii) an algorithm that achieves the lower bound by packing any set of squares with a total area of at most $8/5$ into the unit disk.

The upper bound is implied by any two squares with a side length of $\sqrt{4/5} + \varepsilon$, for arbitrary $\varepsilon > 0$, see Figure 1(d): When placed in the unit disk, either of them must contain the disk center in its interior, so both cannot be packed simultaneously.

In the following, we sketch a constructive proof for the lower bound by describing an algorithm that can pack any instance with total area $8/5$. Because our proof is constructive, it yields a constant-factor approximation algorithm for the smallest disk in which a given set of squares can be packed.

2.1 Description of the Algorithm

In the following, we consider a set of given squares with side lengths $s_1, \ldots, s_n$. We pack them in sequential order by decreasing size into the unit disk $\mathcal{D}$, and assume without loss of generality that $s_1 \geq \cdots \geq s_n$. Our algorithm distinguishes three types of instances:

1. All squares are small, i.e., $s_1 \leq 0.295$.
2. The first four squares are fairly large, i.e., $s_1 \leq \frac{1}{\sqrt{2}}$ and $s_1^2 + s_2^2 + s_3^2 + s_4^2 \geq \frac{8}{5} - \frac{1}{25}$.
3. All other cases.

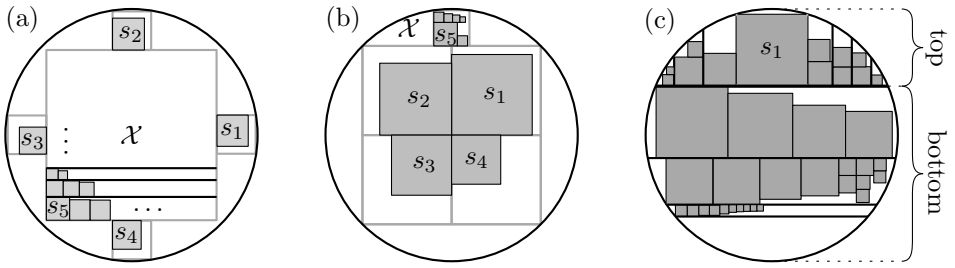


Figure 2 (a) Packing in case 1. (b) Packing in case 2. (c) The packing in the remaining cases is a combination of Top Packing (top) and Bottom Packing (bottom).

In the first case, we pack all but the first four squares into a large square container by Shelf Packing and each of the first four squares adjacent to one of the four sides as illustrated in Figure 2(a). In the second case, we pack the first four squares into a central square container, achieving high enough packed area that it suffices to pack the remaining squares into a smaller subsquare with the worst-case packing density of squares into a square. In the third case, we make extensive use of a refined shelf packing. Specifically, the largest square in the third case is packed into $\mathcal{D}$ as high as possible, see Figure 2(c) and Figure 3 for an illustration. The bottom of this square induces a horizontal split of disk into a *top* and a *bottom* part, which are then packed by two subroutines called Top Packing and Bottom Packing as described in Section 2.2. This yields the following description.

1. If $s_1 \leq 0.295$, place a square of side length $\mathcal{X} = 1.388$ concentric into $\mathcal{D}$ and place one square of side length $\mathcal{X}_i = 0.295$ to each side of $\mathcal{X}$, see Figure 2(a).
   - For $i = 1, 2, 3, 4$, pack each s_i into one of the squares of side length $\mathcal{X}_i = 0.295$.
   - For $i \geq 5$, use Shelf Packing for packing s_i into $\mathcal{X}$.
2. If $s_1 \leq \frac{1}{\sqrt{2}}$ and $s_1^2 + s_2^2 + s_3^2 + s_4^2 \geq \frac{39}{25}$, let $\mathcal{X}_1, \ldots, \mathcal{X}_4$ be the four equally sized maximal squares that fit into $\mathcal{D}$ and let be $\mathcal{X}$ the largest square that can be additionally packed into $\mathcal{D}$, see Figure 2(b).
   - For $i = 1, 2, 3, 4$, pack each s_i into one of the squares of side length $\mathcal{X}_i$.
   - For $i \geq 5$, use Shelf Packing for packing s_i into $\mathcal{X}$.

3. Otherwise
   - Pack s_1 as far as possible to the top into $\mathcal{D}$.
   - For $i \geq 2$,
     - **(3.1)** if possible, use Top Packing for packing s_i,
     - **(3.2)** otherwise, use Bottom Packing for packing s_i.

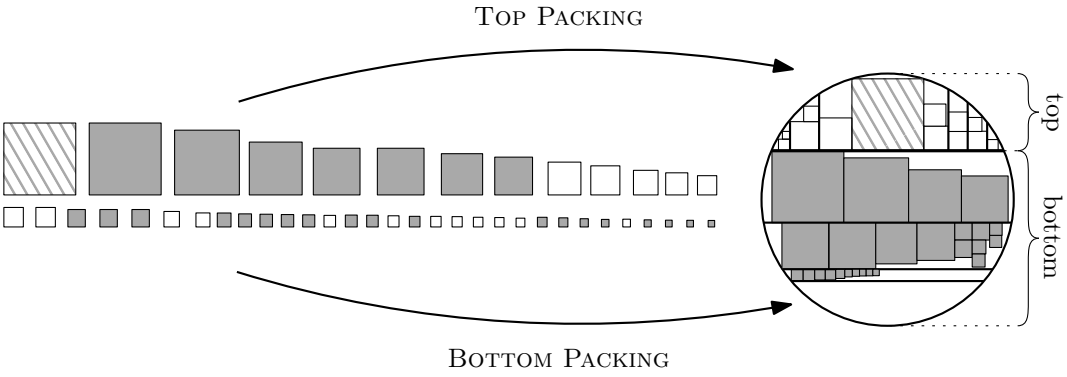


Figure 3 Our algorithm packs squares in decreasing order. The largest (hatched) square is packed as far as possible to the top, inducing a top and a bottom part, with the empty top space consisting of two congruent pockets. Subsequent (white) squares are packed into these top pockets with *Top Packing* (which uses shelf packing as a subroutine) if they fit; if they do not fit, they are shown in gray and packed into the bottom part with *Bottom Packing*, which uses horizontal subcontainer slicing, and vertical shelf packing within each slice.

2.2 Subroutines of Our Algorithm

In the following, we briefly describe the subroutines of our algorithm.

Refined Shelf Packing. In the classic shelf packing procedure by Moon and Moser [7], the objects are packed in the greedy manner by decreasing size in rectangular subcontainers called *shelves*; see top of Figure 1 (a). When an object does not fit in the current shelf, a new shelf is opened; the height of a shelf is determined by the first object that it accommodates. We use two modifications: (1) Parts of the shelf boundaries may be circular arcs; however, we still have a supporting straight axis-parallel boundary and a second, orthogonal straight boundary. (2) Our refined shelf packing uses the axis-parallel boundary line of a shelf as a support line for packing squares; in case of a collision with the circular boundary, we may move a square towards the middle of a shelf if this allows packing it.

Top Packing. The first square s_1 is packed as high as possible into the disk $\mathcal{D}$, see Figure 4 (a). Then the horizontal line ℓ_1 through the bottom of s_1 cuts the container into a top part that contains s_1, with two congruent empty pockets C_ℓ and C_r left and right of s_1; each such pocket has two straight axis-parallel boundaries, b_x and b_y. We use refined shelf packing with shelves parallel to the shorter straight boundary, as shown in Figure 4 (c) and (d). If a square s_i does not fit into either pocket, it is packed into the part below ℓ_1.

Bottom Packing. For packing a squares in the bottom part of $\mathcal{D}$, SubContainer Slicing subdivides the unused portion of the container disk into smaller pieces, by using straight horizontal cuts analougous to shelf packing; see Figure 5 (Left). The height of a subcontainer is determined by the first packed square. Within each subcontainer, (vertical) Refined Shelf Packing is used; see Figure 3 for the overall picture. These shelves are

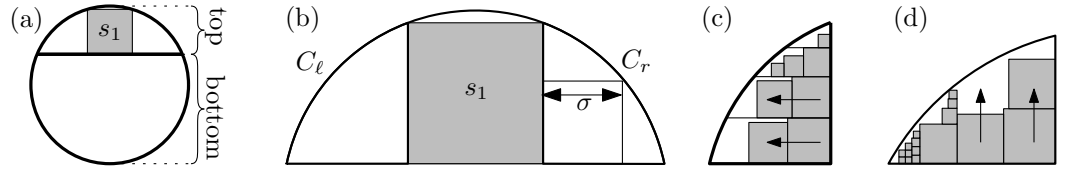


Figure 4 (a) Packing s_1 topmost into $\mathcal{D}$. (b) The top part of $\mathcal{D}$ with the pockets C_ℓ and C_r, and the size σ of the largest inscribed square. (c) A pocket C_ℓ where $b_x \leq b_y$, resulting in horizontal shelf packing. (d) A pocket C_ℓ where $b_x > b_y$, resulting in vertical shelf packing.

packed from the longer of the two horizontal cuts, i.e., away from the boundary that is closer to the disk center; see Figure 5 (Right) for packing the subcontainer.

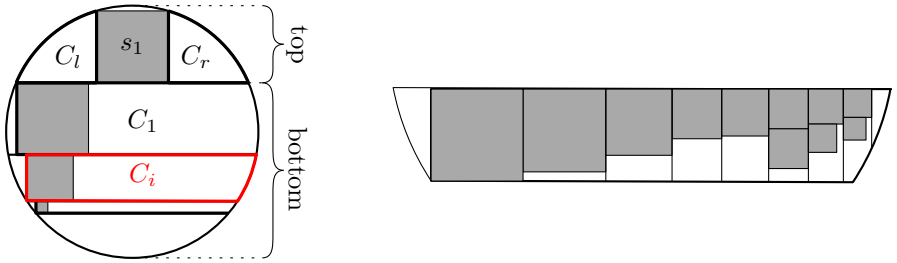


Figure 5 (Left) SubContainer Slicing partitions the lower part of $\mathcal{D}$ into subcontainers C_i, with the height corresponding to the first packed square. (Right) Within each subcontainer, SubContainer Packing places squares into C_i along vertical shelves, starting from the longer straight cut of the subcontainer.

2.3 Correctness of the Algorithm

Similar to the argument by Moon and Moser for squares packed into a square container, we use careful bookkeeping to prove that this algorithm only fails to pack a square in the decreasing if the total area of all squares exceeds the critical bound. The analysis uses an intricate combination of manual analysis and an automated analysis based on interval arithmetic. Details are omitted due to lack of space.

3 Complexity

We present the idea of an hardness proof for packing squares into a disk.

▶ **Theorem 3.1.** *It is* NP*-hard to decide whether a given set of squares fits into a disk.*

The proof uses a reduction from 3-Partition; it is somewhat similar to the one by Leung et al. [6] for deciding whether a given set of squares fits into a given square container, and the one by Demaine, Fekete, and Lang in 2010 [1] for deciding whether a give set of disks fits into a given square container; see Figure 6 for an illustration.

Eight (gray) *framing* squares can only be packed by leaving a central rectangular pocket P and some outside gaps. The numbers of the 3-Partition instance are mapped to a set of (red) *number* squares of almost equal size, with small modifications of size ε_i, such that a triple (i, j, k) of (red) number squares fits into P if and only if $\varepsilon_i + \varepsilon_j + \varepsilon_k \leq 0$, i.e., if there is a feasible 3-Partition. For filling the gaps outside the framing squares, a set of (yellow and

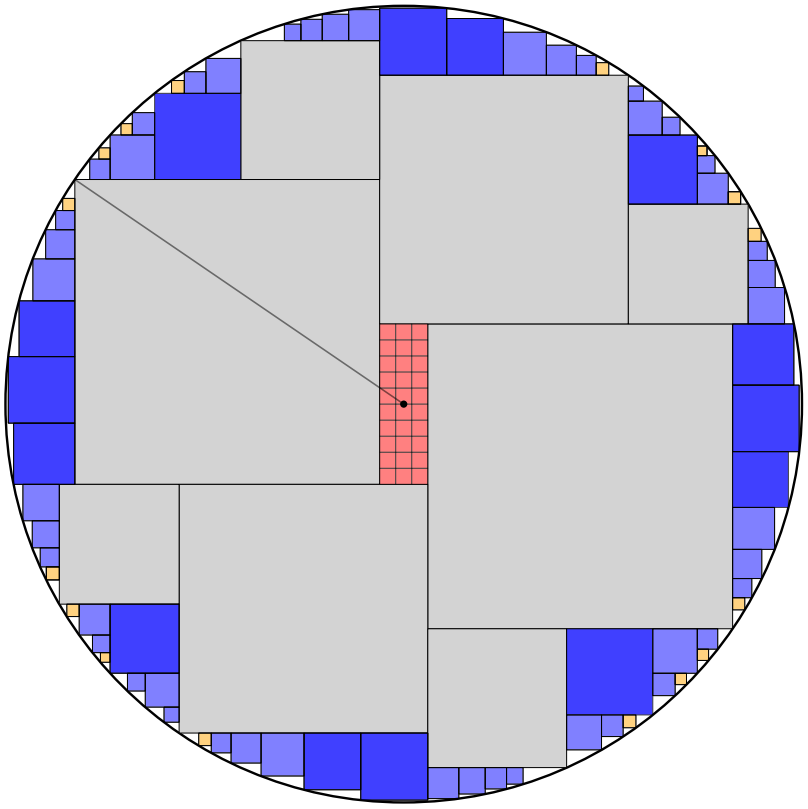


Figure 6 Illustration of the 3-PARTITION reduction.

blue) *filler* squares are constructed, so that no (red) number square can be packed outside P if all filler squares are packed outside P. A detailed proof establishes the following claims.

1. Up to symmetries, the framing squares can only be packed in one canonical way, leaving a central pocket P.
2. The filler squares fight tightly when packed in the canonical manner outside P.
3. When all filler squares are packed outside P, the number squares can only be packed into P. This is possible if and only if there is a feasible 3-partition.
4. Packing a filler square inside P forces an unpackable gap preventing a feasible packing.
5. The overall construction can be realized with squares of sufficiently approximated edge lengths of polynomial description size.

We omit details due to limited space, and the fact that the hardness proof is neither surprising nor central to this paper.

4 Conclusions

We have established the critical density for packing squares into a disk, based on a number of advanced techniques that are more involved than the ones used for packing squares or disks into a square. Numerous questions remain open, in particular the critical density for packing squares of bounded size into a disk. We are optimistic that our techniques will be useful.

References

1 E. D. Demaine, S.P. Fekete, and R. J. Lang. Circle packing for origami design is hard. In *Origami^5: 5th International Conference on Origami in Science, Mathematics and Education*, AK Peters/CRC Press, pages 609–626, 2011.

2 S. P. Fekete and J. Schepers. New classes of fast lower bounds for bin packing problems. *Mathematical Programming*, 91(1):11–31, 2001.

3 S. P. Fekete and J. Schepers. A general framework for bounds for higher-dimensional orthogonal packing problems. *Mathematical Methods of Operations Research*, 60:311–329, 2004.

4 Sándor P. Fekete, Phillip Keldenich, and Christian Scheffer. Packing Disks into Disks with Optimal Worst-Case Density. In *Proceedings 35th International Symposium on Computational Geometry (SoCG 2019)*, pages 35:1–35:19, 2019. doi:10.4230/LIPIcs.SoCG.2019.35.

5 Sándor P. Fekete, Sebastian Morr, and Christian Scheffer. Split packing: Algorithms for packing circles with optimal worst-case density. *Discrete & Computational Geometry*, 61(3):562–594, 2019.

6 J. Y. T. Leung, T. W. Tam, C. S. Wong, G. H. Young, and F. Y. L. Chin. Packing squares into a square. *Journal of Parallel and Distributed Computing*, 10(3):271–275, 1990.

7 J. W. Moon and L. Moser. Some packing and covering theorems. In *Colloquium Mathematicae*, volume 17, pages 103–110. Institute of Mathematics, Polish Academy of Sciences, 1967.

8 S. Morr. Split packing: An algorithm for packing circles with optimal worst-case density. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 99–109, 2017.

Worst-Case Optimal Covering of Rectangles by Disks*

Sándor P. Fekete[1], Utkarsh Gupta[2], Phillip Keldenich[1], Christian Scheffer[1], and Sahil Shah[2]

1 **Department of Computer Science, TU Braunschweig, Germany**
{s.fekete,p.keldenich,c.scheffer}@tu-bs.de
2 **Department of Computer Science & Engineering, IIT Bombay, India**
{utkarshgupta149,sahilshah00199}@gmail.com

Abstract

We provide the solution for a fundamental problem of geometric optimization by giving a complete characterization of worst-case optimal disk coverings of rectangles: For any $\lambda \geq 1$, the critical covering area $A^*(\lambda)$ is the minimum value for which any set of disks with total area at least $A^*(\lambda)$ can cover a rectangle of dimensions $\lambda \times 1$. We show that there is a threshold value $\lambda_2 = \sqrt{\sqrt{7}/2 - 1/4} \approx 1.035797\ldots$, such that for $\lambda < \lambda_2$ the critical covering area $A^*(\lambda)$ is $A^*(\lambda) = 3\pi\left(\frac{\lambda^2}{16} + \frac{5}{32} + \frac{9}{256\lambda^2}\right)$, and for $\lambda \geq \lambda_2$, the critical area is $A^*(\lambda) = \pi(\lambda^2+2)/4$; these values are tight. For the special case $\lambda = 1$, i.e., for covering a unit square, the critical covering area is $\frac{195\pi}{256} \approx 2.39301\ldots$. The proof uses a careful combination of manual and automatic analysis, demonstrating the power of the employed interval arithmetic technique.

1 Introduction

Given a collection of (not necessarily equal) disks, is it possible to arrange them so that they completely cover a given region, such as a square or a rectangle? Covering problems of this type are of fundamental theoretical interest, but also have a variety of different applications, most notably in sensor networks, communication networks and wireless communication [22], surveillance, robotics, and even gardening and sports facility management, as shown in Figure 1.

If the total area of the disks is small, it is clear that completely covering the region is impossible. On the other hand, if the total disk area is sufficiently large, finding a covering

* This is an extended abstract of our paper *Worst-Case Optimal Covering of Rectangles by Disks* [15]. A video presenting the main result can be found at `https://www.ibr.cs.tu-bs.de/users/fekete/Videos/Cover_full.mp4` .

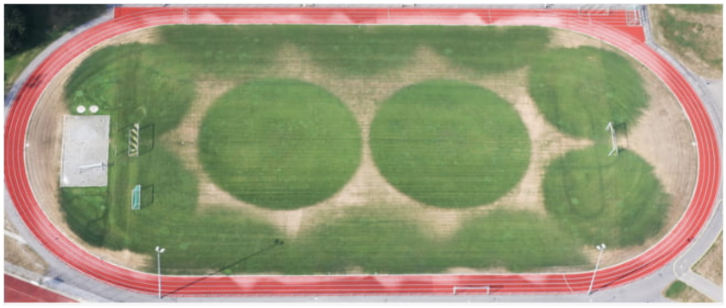


Figure 1 An incomplete covering of a rectangle by disks: Sprinklers on a soccer field during a drought. (Source: dpa [13].)

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

seems easy; however, for rectangles with large aspect ratio, a major fraction of the covering disks may be useless, so a relatively large total disk area may be required. The same issue is of clear importance for applications: What fraction of the total cost of disks can be put to efficient use for covering? This motivates the question of characterizing a critical threshold: For any given λ, find the minimum value $A^*(\lambda)$ for which any collection of disks with total area at least $A^*(\lambda)$ can cover a rectangle of dimensions $\lambda \times 1$. What is the critical covering area of $\lambda \times 1$ rectangles? In this paper we establish a complete and tight characterization that generalizes to arbitrary rectangles by scaling and rotating.

1.1 Related Work

Like many other packing and covering problems, disk covering is typically quite difficult, compounded by the geometric complications of dealing with irrational coordinates that arise when arranging circular objects. This is reflected by the limitations of provably optimal results for the largest disk, square or triangle that can be covered by n unit disks, and hence, the "thinnest" disk covering, i.e., a covering of optimal density. As early as 1915, Neville [27] computed the optimal arrangement for covering a disk by five unit disks, but reported a wrong optimal value; much later, Bezdek [6, 7] gave the correct value for $n = 5, 6$. As recently as 2005, Fejes Tóth [33] established optimal values for $n = 8, 9, 10$. Szalkai [32] gave an optimal solution for a small special case ($n = 3$) of a general problem posed by Connelly in 2008, who asked how one should place n small disks of radius r to cover the largest possible area of a disk of radius $R > r$. For covering arbitrary rectangles by n unit disks, Heppes and Mellissen [20] gave optimal solutions for $n \leq 5$; Melissen and Schuur [24] extended this for $n = 6, 7$. See Friedman [19] for the best known solutions for $n \leq 12$. Covering equilateral triangles by n unit disks has also been studied. Melissen [23] gave optimal results for $n \leq 10$, and conjectures for $n \leq 18$; the difficulty of these seemingly small problems is illustrated by the fact that Nurmela [28] gave conjectured optimal solutions for $n \leq 36$, improving the conjectured optimal covering for $n = 13$ of Melissen. Carmi, Katz and Lev-Tov [11] considered algorithms for covering point sets by unit disks at fixed locations. There are numerous other related problems and results; for relevant surveys, see Fejes Tóth [14] (Section 8), Fejes Tóth [34] (Chapter 2), Brass, Moser and Pach [10] (Chapter 2) and the book by Böröczky [9].

Even less is known for covering by non-uniform disks, with most previous research focusing on algorithmic aspects. Alt et al. [3] gave algorithmic results for minimum-cost covering of point sets by disks, where the cost function is $\sum_j r_j^\alpha$ for some $\alpha > 1$, which includes the case of total disk area for $\alpha = 2$. Agnetis et al. [2] discussed covering a line segment with variable radius disks. Abu-Affash et al. [1] studied covering a polygon minimizing the sum of areas; for recent improvements, see Bhowmick,Varadarajan and Xue [8]. Bánhelyi, Palatinus and Lévai [4] gave algorithmic results for the covering of polygons by variable disks with prescribed centers.

The dual question of *packing* unit disks into a square has also attracted attention. For $n = 13$, the optimal value for the densest square covering was only established in 2003 [18], while the optimal value for 14 unit disks is still unproven; densest packings of n disks in equilateral triangles are subject to a long-standing conjecture by Erdős and Oler from 1961 [29] that is still open for $n = 15$. Many authors have considered heuristics for circle packing problems, see [31, 21] for overviews of numerous heuristics and optimization methods. The best known solutions for packing equal disks into squares, triangles and other shapes are published on Specht's website `http://packomania.com` [30]. Establishing the critical packing density, i.e., the disk area that can always be packed into a unit square, for (not

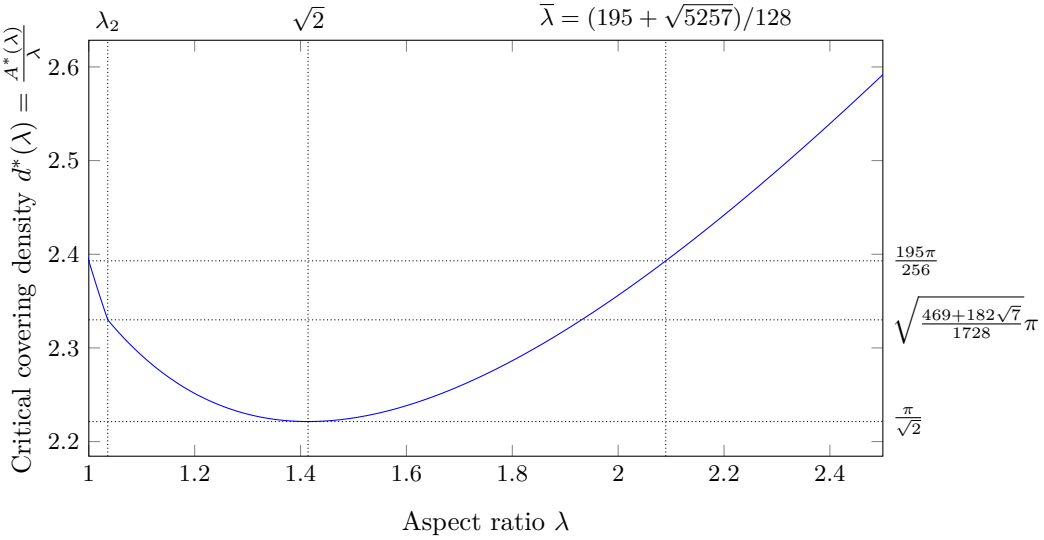


Figure 2 The critical covering density $d^*(\lambda)$ depending on λ and its values at the threshold value λ_2, the global minimum $\sqrt{2}$ and the aspect ratio $\overline{\lambda}$ at which the density becomes as bad as for the square.

necessarily equal) disks in a square was proposed by Demaine, Fekete, and Lang [12] and solved by Morr, Fekete and Scheffer [26, 17]. Using a recursive procedure for cutting the container into triangular pieces, they proved that the critical packing density of disks in a square is $\frac{\pi}{3+2\sqrt{2}} \approx 0.539$. The critical density for (not necessarily equal) disks in a disk was recently proven to be $1/2$ by Fekete, Keldenich and Scheffer [16]; see the video [5] for an overview and various animations. The critical packing density of (not necessarily equal) squares was established in 1967 by Moon and Moser [25], who used a shelf-packing approach to establish the value of $1/2$ for packing into a square.

For more related work, we refer the reader to the full version of our paper [15].

1.2 Our Contribution

We show that there is a threshold value $\lambda_2 = \sqrt{\sqrt{7}/2 - 1/4} \approx 1.035797\ldots$, such that for $\lambda < \lambda_2$ the critical covering area $A^*(\lambda)$ is $A^*(\lambda) = 3\pi\left(\frac{\lambda^2}{16} + \frac{5}{32} + \frac{9}{256\lambda^2}\right)$, and for $\lambda \geq \lambda_2$, the critical area is $A^*(\lambda) = \pi(\lambda^2+2)/4$. These values are tight: For any λ, any collection of disks of total area $A^*(\lambda)$ can be arranged to cover a $\lambda \times 1$-rectangle, and for any $a(\lambda) < A^*(\lambda)$, there is a collection of disks of total area $a(\lambda)$ such that a $\lambda \times 1$-rectangle cannot be covered. (See Figure 2 for a graph showing the (normalized) critical covering density, and Figure 3 for examples of worst-case configurations.) The point $\lambda = \lambda_2$ is the unique real number greater than 1 for which the two bounds $3\pi\left(\frac{\lambda^2}{16} + \frac{5}{32} + \frac{9}{256\lambda^2}\right)$ and $\pi\frac{\lambda^2+2}{4}$ coincide; see Figure 2. At this so-called *threshold value*, the worst case changes from three identical disks to two disks — the circumcircle $r_1^2 = \frac{\lambda^2+1}{4}$ and a disk $r_2^2 = \frac{1}{4}$; see Figure 3. For the special case $\lambda = 1$, i.e., for covering a unit square, the critical covering area is $\frac{195\pi}{256} \approx 2.39301\ldots$.

The proof uses a careful combination of manual and automatic analysis, demonstrating the power of the employed interval arithmetic technique.

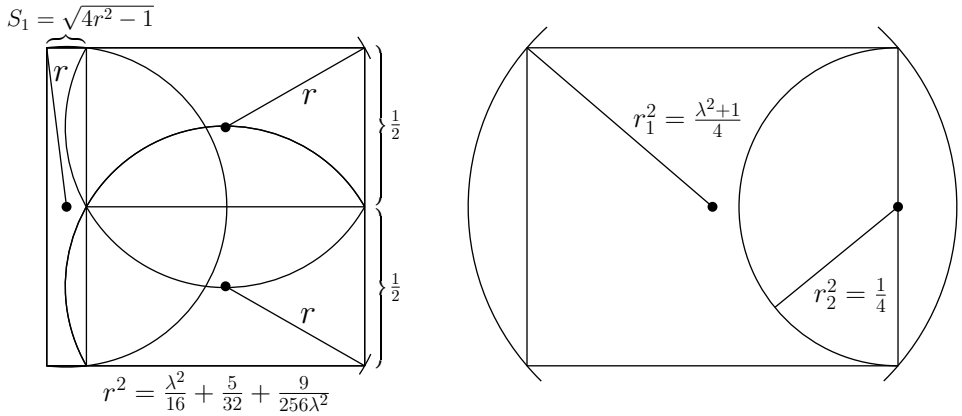


Figure 3 Worst-case configurations for small $\lambda \leq \lambda_2$ (left) and for large $\lambda \geq \lambda_2$ (right). Shrinking r or r_1 by any $\varepsilon > 0$ in either configuration leads to an instance that cannot be covered.

2 High-Level Description

Our main theorem gives a closed-form solution for the *critical covering area* $A^*(\lambda)$ for any $\lambda \geq 1$, i.e., for any given rectangle $\mathcal{R}$, we determine the total disk area that is (1) sometimes necessary and (2) always sufficient to cover $\mathcal{R}$. Due to limited space, we only sketch the overall approach; details are contained in the full version [15] of the paper.

▶ **Theorem 2.1.** *Let $\lambda \geq 1$ and let $\mathcal{R}$ be a rectangle of dimensions $\lambda \times 1$. Let*

$$\lambda_2 = \sqrt{\frac{\sqrt{7}}{2} - \frac{1}{4}} \approx 1.035797\ldots, \text{ and } A^*(\lambda) = \begin{cases} 3\pi\left(\frac{\lambda^2}{16} + \frac{5}{32} + \frac{9}{256\lambda^2}\right), & \text{if } \lambda < \lambda_2, \\ \pi\frac{\lambda^2+2}{4}, & \text{otherwise.} \end{cases}$$

(1) *For any $a < A^*(\lambda)$, there is a set D^- of disks with $A(D^-) = a$ that cannot cover $\mathcal{R}$.*
(2) *Let $D = \{r_1, \ldots, r_n\} \subset \mathbb{R}$, $r_1 \geq r_2 \geq \ldots \geq r_n > 0$ be any collection of disks identified by their radii. If $A(D) \geq A^*(\lambda)$, then D can cover $\mathcal{R}$.*

The critical covering area does not depend linearly on the area λ of the rectangle; it also depends on the rectangle's aspect ratio. Figure 2 shows a plot of the dependency of the critical covering density $d^*(\lambda) \coloneqq \frac{A^*(\lambda)}{\lambda}$, i.e., the amount of disk area required per rectangle area, on λ. In the following, to simplify notation, we factor out π if possible; instead of working with the areas $A(D)$ or $A^*(\lambda)$ of the disks, we use their *weight* $W(D)$, i.e., their area divided by π. Similarly, we work with the covering coefficient $E^*(\lambda) \coloneqq \frac{d^*(\lambda)}{\pi}$ instead of the density $d^*(\lambda)$; a lower covering coefficient corresponds to a more efficient covering.

As shown in Figure 2, the critical covering coefficient $E^*(\lambda)$ is monotonically decreasing from $\lambda = 1$ to $\sqrt{2}$ and monotonically increasing for $\lambda > \sqrt{2}$. For a square, $E^*(1) = \frac{195}{256}$; the point $\lambda > 1$ for which the covering coefficient becomes as bad as for the square is $\overline{\lambda} \coloneqq \frac{195+\sqrt{5257}}{128} \approx 2.08988\ldots$; for all $\lambda \leq \overline{\lambda}$, the covering coefficient is at most $\frac{195}{256}$.

2.1 Proof Components

The proof of Theorem 2.1 uses a number of components. First is a lemma that describes the worst-case configurations and shows tightness, i.e., claim (1), of Theorem 2.1 for all λ.

▶ **Lemma 2.2.** *Let $\lambda \geq 1$ and let $\mathcal{R}$ be a rectangle of dimensions $\lambda \times 1$. (1) Two disks of weight $r_1^2 = \frac{\lambda^2+1}{4}$ and $r_2^2 = \frac{1}{4}$ suffice to cover $\mathcal{R}$. (2) For any $\varepsilon > 0$, two disks of weight $r_1^2 - \varepsilon$ and r_2^2 do not suffice to cover $\mathcal{R}$. (3) Three identical disks of weight $r^2 = \frac{\lambda^2}{16} + \frac{5}{32} + \frac{9}{256\lambda^2}$ suffice to cover a rectangle $\mathcal{R}$ of dimensions $\lambda \times 1$. (4) For $\lambda \leq \lambda_2$ and any $\varepsilon > 0$, three identical disks of weight $r_-^2 := r^2 - \varepsilon$ do not suffice to cover $\mathcal{R}$.*

For large λ, the critical covering coefficient $E^*(\lambda)$ of Theorem 2.1 becomes worse, as large disks cannot be used to cover the rectangle efficiently. If the weight of each disk is bounded by some $\sigma \geq r_1^2$, we provide the following lemma achieving a better covering coefficient $E(\sigma)$ with $E^*(\overline{\lambda}) \leq E(\sigma) \leq E^*(\lambda)$. This coefficient is independent of the aspect ratio of $\mathcal{R}$.

▶ **Lemma 2.3.** *Let $\hat{\sigma} := \frac{195\sqrt{5257}}{16384} \approx 0.8629$. Let $\sigma \geq \hat{\sigma}$ and $E(\sigma) := \frac{1}{2}\sqrt{\sqrt{\sigma^2+1}+1}$. Let $\lambda \geq 1$ and $D = \{r_1, \ldots, r_n\}$ be any collection of disks with $\sigma \geq r_1^2 \geq \ldots \geq r_n^2$ and $W(D) = \sum_{i=1}^{n} r_i^2 \geq E(\sigma)\lambda$. Then D can cover a rectangle $\mathcal{R}$ of dimensions $\lambda \times 1$.*

Note that $E(\hat{\sigma}) = \frac{195}{256}$ is the best covering coefficient established by Lemma 2.3, coinciding with the critical covering coefficient of the square established by Theorem 2.1. Thus, we can cover any rectangle with covering coefficient $\frac{195}{256}$ if the largest disk satisfies $r_1^2 \leq \hat{\sigma}$.

The final component is the following Lemma 2.4, which also gives a better covering coefficient if the size of the largest disk is bounded. The bound on the largest radius that is required for Lemma 2.4 is smaller than for Lemma 2.3; in return, the covering coefficient that Lemma 2.4 yields is better. We remark that the result of Lemma 2.4 is not tight.

▶ **Lemma 2.4.** *Let $\lambda \geq 1$ and let $\mathcal{R}$ be a rectangle of dimensions $\lambda \times 1$. Let $D = \{r_1, \ldots, r_n\}$, $0.375 \geq r_1 \geq \ldots \geq r_n > 0$ be a collection of disks. If $W(D) \geq 0.61\lambda$, or equivalently $A(D) \geq 0.61\pi\lambda \approx 1.9164\lambda$, then D suffices to cover $\mathcal{R}$.*

2.2 Proof Overview

The proofs of Theorem 2.1 and Lemmas 2.3 and 2.4 work by induction on the number of disks. For proving Lemma 2.3 for n disks, we use Theorem 2.1 for n disks. For proving Theorem 2.1 for n disks, we use Lemma 2.4 for n disks; Lemma 2.3 is only used for fewer than n disks; see Figure 4. For proving Lemma 2.4 for n disks, we only use Theorem 2.1 and Lemma 2.3 for fewer than n disks. Therefore, there are no cyclic dependencies in our argument; however, we have to perform the induction for Theorem 2.1 and Lemmas 2.3 and 2.4 simultaneously.

Strategies. The proofs of Theorem 2.1 and Lemma 2.4 are constructive; they are based on an efficient recursive algorithm that uses a set of simple *strategies*. We go through the list of strategies in some fixed order. For each strategy, we check a sufficient criterion for the strategy to work. We call these criteria *success criteria*. They only depend on the total available weight and a constant number of largest disks. If we cannot guarantee that a strategy works by its success criterion, we simply disregard the strategy; this means that our algorithm does not have to backtrack. We prove that, regardless of the distribution of the disks' weight, at least one success criterion is met, implying that we can always apply at least one strategy. The number of strategies and thus success criteria is large — more than 40 strategies considering over 500 combinatorially different placements of the largest disks, which would presumably need to be considered in a manual analysis. This is where the need for automatic assistance comes from.

Recursion. Typical strategies are recursive; they consist of splitting the collection of disks into smaller parts, splitting the rectangle accordingly, and recursing, or recursing after fixing the position of a constant number of large disks.

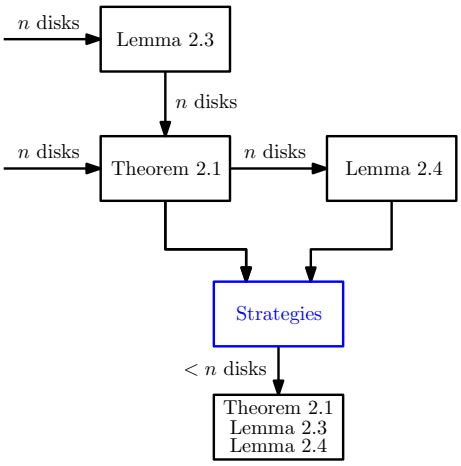


Figure 4 The inductive structure of the proof; the blue parts are computer-aided.

In the entire remaining proof, the criterion we use to guarantee that recursion works is as follows. Given a collection $D' \subsetneq D$ and a rectangular region $\mathcal{R}' \subsetneq \mathcal{R}$, we check whether the preconditions of Theorem 2.1 or Lemma 2.3 or 2.4 are met after appropriately scaling and rotating $\mathcal{R}'$ and the disks. Note that, due to the scaling, the radius bounds of Lemmas 2.3 and 2.4 depend on the length of the shorter side of $\mathcal{R}'$. In some cases where we apply recursion, we have more weight than necessary to satisfy the weight requirement for recursion according to Lemma 2.3 or 2.4, but these lemmas cannot be applied due to the radius bound. In that case, we also check whether we can apply Lemma 2.3 or 2.4 after increasing the length of the shorter side of $\mathcal{R}'$ as far as the disk weight allows. This excludes the case that we cannot recurse on $\mathcal{R}'$ due to the radius bound, but there is some $\mathcal{R}'' \supset \mathcal{R}'$ on which we could recurse.

2.3 Interval Arithmetic

We use interval arithmetic to prove that there always is a strategy that works. In interval arithmetic, operations like addition, multiplication or taking a square root are performed on intervals $[a, b] \subset \mathbb{R}$ instead of numbers. Arithmetic operations on intervals are derived from their real counterparts as follows. The result of an operation $\circ$ in interval arithmetic is

$$[a_1, b_1] \circ [a_2, b_2] := \left[\min_{x_1 \in [a_1, b_1], x_2 \in [a_2, b_2]} x_1 \circ x_2, \max_{x_1 \in [a_1, b_1], x_2 \in [a_2, b_2]} x_1 \circ x_2 \right].$$

Thus, the result of an operation is the smallest interval that contains all possible results of $x \circ y$ for $x \in [a_1, b_1], y \in [a_2, b_2]$. Unary operations are defined analogously.

3 Conclusion

Our worst-case values correspond to instances with only 2 or 3 relatively large disks; if we have an upper bound R on the size of the largest disk, this gives rise to the critical covering area $A^*_R(\lambda)$ for $\lambda \times 1$-rectangles. Getting some tight bounds on $A^*_R(\lambda)$ would be interesting and useful. Establishing the critical covering density for disks and triangles is also open. We are optimistic that an approach similar to the one of this paper can be used for a solution.

Computing optimal coverings by disks is quite difficult. Deciding whether a given collection of disks can be packed into a unit square, is known to be NP-hard [12], the complexity of deciding whether a given set of disks can be used to cover a unit square is still open.

References

1 A. K. Abu-Affash, P. Carmi, M. J. Katz, and G. Morgenstern. Multi cover of a polygon minimizing the sum of areas. *International Journal of Computational Geometry & Applications*, 21(06):685–698, 2011.

2 A. Agnetis, E. Grande, P. B. Mirchandani, and A. Pacifici. Covering a line segment with variable radius discs. *Computers & Operations Research*, 36(5):1423–1436, 2009.

3 H. Alt, E. M. Arkin, H. Brönnimann, J. Erickson, S. P. Fekete, C. Knauer, J. Lenchner, J. S. B. Mitchell, and K. Whittlesey. Minimum-cost coverage of point sets by disks. In *Proc. 22nd Annu. ACM Sympos. Comput. Geom.*, pages 449–458, 2006.

4 B. Bánhelyi, E. Palatinus, and B. L. Lévai. Optimal circle covering problems and their applications. *Central European Journal of Operations Research*, 23(4):815–832, 2015.

5 A. T. Becker, S. P. Fekete, P. Keldenich, S. Morr, and C. Scheffer. Packing Geometric Objects with Optimal Worst-Case Density (Multimedia Exposition). In *Proceedings 35th International Symposium on Computational Geometry (SoCG)*, pages 63:1–63:6, 2019. Video available at `https://www.ibr.cs.tu-bs.de/users/fekete/Videos/PackingCirclesInSquares.mp4`.

6 K. Bezdek. *Körök optimális fedései (Optimal covering of circles)*. PhD thesis, Eötvös Lorand University, 1979.

7 K. Bezdek. Über einige optimale Konfigurationen von Kreisen. *Ann. Univ. Sci. Budapest Rolando Eötvös Sect. Math*, 27:143–151, 1984.

8 S. Bhowmick, K. R. Varadarajan, and S. Xue. A constant-factor approximation for multi-covering with disks. *JoCG*, 6(1):220–234, 2015.

9 K. Böröczky Jr. *Finite packing and covering*, volume 154. Cambridge University Press, 2004.

10 P. Brass, W. O. Moser, and J. Pach. Density problems for packings and coverings. *Research Problems in Discrete Geometry*, pages 5–74, 2005.

11 P. Carmi, M. J. Katz, and N. Lev-Tov. Covering points by unit disks of fixed location. In *Proc. International Symposium on Algorithms and Computation (ISAAC)*, pages 644–655. Springer, 2007.

12 E. D. Demaine, S. P. Fekete, and R. J. Lang. Circle packing for Origami design is hard. In *Origami^5: 5th International Conference on Origami in Science, Mathematics and Education*, AK Peters/CRC Press, pages 609–626, 2011.

13 dpa. Rasensprenger zeichnet Kreise auf Fußballfeld, 2018.

14 G. Fejes Tóth. Recent progress on packing and covering. *Contemporary Mathematics*, 223:145–162, 1999.

15 S. P. Fekete, U. Gupta, P. Keldenich, C. Scheffer, and S. Shah. Worst-Case Optimal Covering of Rectangles by Disks. In *Proceedings 36th International Symposium on Computational Geometry (SoCG 2020)*, 2020. To appear. A video is available at `https://www.ibr.cs.tu-bs.de/users/fekete/Videos/Cover_full.mp4`.

16 S. P. Fekete, P. Keldenich, and C. Scheffer. Packing Disks into Disks with Optimal Worst-Case Density. In *Proceedings 35th International Symposium on Computational Geometry (SoCG 2019)*, pages 35:1–35:19, 2019.

17 S. P. Fekete, S. Morr, and C. Scheffer. Split packing: Algorithms for packing circles with optimal worst-case density. *Discrete & Computational Geometry*, 2018.

18 F. Fodor. The densest packing of 13 congruent circles in a circle. *Beiträge zur Algebra und Geometrie (Contributions to Algebra and Geometry)*, 44:431–440, 2003.

19 E. Friedman. Circles covering squares web page, 2014. `http://www2.stetson.edu/~efriedma/circovsqu/`.

20 A. Heppes and H. Melissen. Covering a rectangle with equal circles. *Periodica Mathematica Hungarica*, 34(1-2):65–81, 1997.

21 M. Hifi and R. M'hallah. A literature review on circle and sphere packing problems: models and methodologies. *Advances in Operations Research*, 2009. Article ID 150624.

22 C.-F. Huang and Y.-C. Tseng. A survey of solutions for the coverage problems in wireless sensor networks. *Journal of Internet Technology*, 6(1):1–8, 2005.

23 H. Melissen. Loosest circle coverings of an equilateral triangle. *Mathematics Magazine*, 70(2):118–124, 1997.

24 J. B. M. Melissen and P. C. Schuur. Covering a rectangle with six and seven circles. *Discrete Applied Mathematics*, 99(1-3):149–156, 2000.

25 J. W. Moon and L. Moser. Some packing and covering theorems. In *Colloquium Mathematicae*, volume 17, pages 103–110. Institute of Mathematics, Polish Academy of Sciences, 1967.

26 S. Morr. Split packing: An algorithm for packing circles with optimal worst-case density. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 99–109, 2017.

27 E. H. Neville. On the solution of numerical functional equations. *Proceedings of the London Mathematical Society*, 2(1):308–326, 1915.

28 K. J. Nurmela. Conjecturally optimal coverings of an equilateral triangle with up to 36 equal circles. *Experimental Mathematics*, 9(2):241–250, 2000.

29 N. Oler. A finite packing problem. *Canadian Mathematical Bulletin*, 4:153–155, 1961.

30 E. Specht. Packomania, 2015. `http://www.packomania.com/`.

31 P. G. Szabó, M. C. Markót, T. Csendes, E. Specht, L. G. Casado, and I. García. *New Approaches to Circle Packing in a Square*. Springer US, 2007.

32 B. Szalkai. Optimal cover of a disk with three smaller congruent disks. *Advances in Geometry*, 16(4):465–476, 2016.

33 G. F. Tóth. Thinnest covering of a circle by eight, nine, or ten congruent circles. *Combinatorial and computational geometry*, 52(361):59, 2005.

34 G. F. Tóth. Packing and covering. In *Handbook of Discrete and Computational Geometry, Third Edition*, pages 27–66. Chapman and Hall/CRC, 2017.

Connected Coordinated Motion Planning with Bounded Stretch

Sándor P. Fekete[1], Phillip Keldenich[1], Ramin Kosfeld[1], Christian Rieck[1], and Christian Scheffer[1]

1 **Departement of Computer Science, TU Braunschweig, Germany**
{s.fekete, p.keldenich, r.kosfeld, c.rieck, c.scheffer}@tu-bs.de

Abstract

We consider the problem of coordinated motion planning for a swarm of simple, identical robots: From a given start grid configuration of robots, we need to reach a desired target configuration via a sequence of parallel, continuous, collision-free robot motions, such that the set of robots stays connected at all times. The objective is to minimize the *makespan* of the motion schedule, i.e., to reach the new configuration in a minimum amount of time. We show that this problem is NP-hard, even for deciding whether a makespan of 2 can be achieved, while it is possible to check in polynomial time whether a makespan of 1 can be achieved. We also provide a constant-factor approximation for *fat* configurations. Our algorithm achieves a *constant stretch factor*: If mapping the start configuration to the target configuration requires a maximum Manhattan distance of d, then the total duration of our overall schedule is $\mathcal{O}(d)$, which is optimal up to constant factors.

1 Introduction

Consider a connected configuration of objects, e.g., a swarm of mobile robots, which needs to be transformed into a desired target configuration by a sequence of parallel, continuous, collision-free motions that keeps the overall arrangement connected at all times. Such problems occur in many contexts requiring relocation of autonomous agents; the connectivity constraint arises naturally in many physical scenarios, e.g., for reconfigurable matter in space. How can we coordinate the corresponding motion, such that a desired target configuration is reached within a minimum amount of time, called *makespan*, without losing connectivity? As it turns out, this problem is provably hard, even in relatively simple cases. We present methods that, provided sufficient fatness of the start and target configuration, realize *constant stretch*: If mapping the start configuration C_s to the target configuration C_t requires a maximum Manhattan distance of d, then the total duration of our overall schedule is $\mathcal{O}(d)$.

Our Contributions. We provide new results for questions arising from efficiently reconfiguring a connected, unlabeled swarm of robots from a given start configuration C_s into a desired target configuration C_t, aiming for minimizing the overall makespan, i.e., the total time required for running the full schedule, and maintaining connectivity in each step.

- There exists a polynomial time algorithm to decide whether there is a schedule with a makespan of 1 that transforms C_s into C_t, see Theorem 1.
- It is NP-hard to decide whether there is a schedule with a makespan of 2 that transforms C_s into C_t, see Theorem 2. This implies NP-hardness of approximating the minimum makespan within a constant of $(\frac{3}{2} - \varepsilon)$, for any $\varepsilon > 0$, see Corollary 3.
- There is a constant c such that for any pair of start and target configurations with fatness of at least c, a schedule with constant stretch can be computed in polynomial time, see Theorem 4. This implies that there is a constant-factor approximation for the problem of computing schedules with minimal makespan restricted to pairs of start and target configuration with a fatness of at least c, see Corollary 8.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Related Work. Coordinating the motion of many agents plays a central role when dealing with large numbers of moving robots, vehicles, aircraft, or people. Basic questions arise in many applications, such as ground swarm robotics [8, 9], aerial swarm robotics [2, 12], air traffic control [3], and vehicular traffic networks [6, 10], and goes back to work by Schwartz and Sharir [11]. Most previous work has largely focused on sequential schedules, where one robot moves at a time, with objectives such as minimizing the number of moves. In practice, however, robots usually move simultaneously, so we desire a parallel motion schedule, with the objective of minimizing the makespan. In recent work [1, 4, 5], we provide several fundamental insights into these problems of coordinated motion planning for the scenario with labeled robots without a connectivity constraint.

2 Preliminaries

We consider n unlabeled *robots* at integer grid positions, inducing a grid graph. A configuration C is *c-fat*, if for any vertex v there is a $(c \times c)$-block $B \subset C$, such that $v \in B$.

A robot can move in discrete time steps by changing its location from a grid position v to an adjacent grid position w; this is denoted by $v \to w$. Two moves $v_1 \to w_1$ and $v_2 \to w_2$ are *collision-free* if $v_1 \neq v_2$ and $w_1 \neq w_2$. A *transformation* between two configurations $C_1 = \{v_1, \ldots, v_n\}$ and $C_2 = \{w_1, \ldots, w_n\}$ is a set of collision-free moves $\{v_i \to w_i \mid i = 1, \ldots, n\}$. For $M \in \mathbb{N}$, a *schedule* is a sequence $C_1 \rightrightarrows C_{M+1} := C_1 \to \cdots \to C_{M+1}$ of transformations, with a *makespan* of M. A *stable* schedule, $C_s \rightrightarrows_\chi C_t := C_s \to_\chi \cdots \to_\chi C_t$ between the start configuration C_s and the target configuration C_t uses only connected configurations. A *bottleneck matching* between the vertices of C_s and C_t minimizes the maximal Manhattan distance d, called the *diameter* of (C_s, C_t), between two matched vertices from C_s and C_t. The *stretch (factor)* of a (stable) schedule is the ratio between the makespan M and the diameter d.

3 Makespan 1 and 2

It can efficiently be decided whether a stable schedule $C_s \to_\chi C_t$ with a makespan of 1 exist.

▶ **Theorem 1.** *For two configurations C_s and C_t, each with n vertices, it can be decided in polynomial time whether there is a schedule with a makespan of 1 transforming C_s into C_t.*

To decide this, it suffices to compute a maximum matching in the bipartite graph $G = (V_s \cup V_t, E)$ consisting of vertices for all occupied positions in both configurations, and edges between vertices if their respective positions are adjacent or identical. It is easy to see that there is a schedule with a makespan of 1 if and only if G admits a perfect matching. This can be checked with the method of Hopcraft and Karp in $\mathcal{O}(n^{5/2})$ time [7].

However, even for a makespan of 2, the same problem becomes provably hard.

▶ **Theorem 2.** *For a pair of configurations C_s and C_t, each with n vertices, deciding whether there is a stable schedule with a makespan of 2 transforming C_s into C_t is NP-hard.*

The proof establishes a reduction from Planar Monotone 3SAT, which asks to decide whether we can satisfy a Boolean 3-CNF formula φ for which in each clause the literals are either all positive or all negative. For every instance φ of Planar Monotone 3SAT, we construct an instance I_φ, consisting of a start configuration C_s and a target configuration C_t, as indicated in Figure 2. In the figure, we use three differently colored squares to indicate occupied positions in the C_s (red), in C_t (dark cyan), and in both configurations (gray).

We can argue that there is a stable schedule transforming the start configuration into the target configuration with a makespan of 2, if and only if φ is satisfiable. In order to transform C_s into C_t, the separation gadgets (yellow) ensure that in the single intermediate configuration, all clause and helper gadgets (shades of blue) are disconnected from each other. Therefore, to satisfy the connectivity constraint, some robots of the variable gadget (light red) have to move in a very particular way, such that these robots ensure connections between the variable gadget and the clause gadgets. At the same time, we ensure that robots representing a variable can either connect this variable to their positive or to their negative literal containing clauses (otherwise the connectivity within the variable gadget would be broken); thus, these movements can be used to determine a valid assignment for φ.

Most of the gadgets are straightforward. Because we use the movements in the variable gadget to determine a variable assignment for φ, we briefly explain how this works.

Thus, consider the arrangement consisting of start and target configurations in Figure 1. Without loss of generality, we consider the situation in which in the single intermediate configuration robots representing the positive arm segment are connected to their respective bridges. Hence, at least one robot of this segment (i.e., $p_1, \ldots, p_\ell$) has to move up. Then the robots on X_i and A_i have unique target locations, i.e., Y_i and B_i, respectively. Given all these moves, the robot R_i has to move up to maintain connectivity. Because R_i cannot simultaneously maintain connectivity for $\overline{X_i}$ and $\overline{A_i}$, its movement can be used to determine the variable assignment for φ.

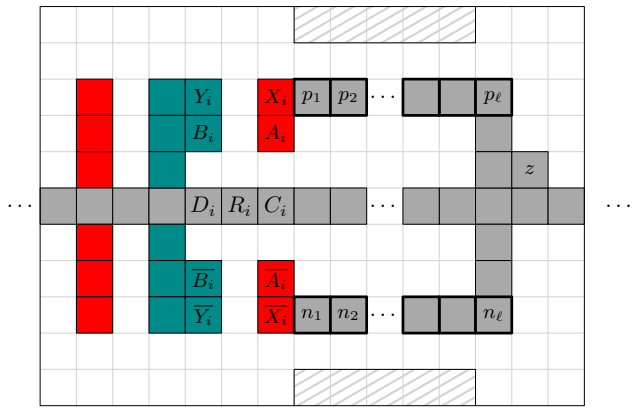


Figure 1 The variable gadget. The robot R_i is used to determine the variable assignment.

More technical details of the proof of Theorem 2 are omitted due to space constraints. As the proof of Theorem 2 shows that it is NP-hard to decide whether there is a stable schedule with a makespan of 2 transforming C_s into C_t, we obtain the following.

▶ **Corollary 3.** *It is NP-hard to compute for a pair of configurations C_s and C_t, each with n vertices, a stable schedule that transforms C_s into C_t within a constant of $(\frac{3}{2} - \varepsilon)$ (for any $\varepsilon > 0$) of the minimum makespan.*

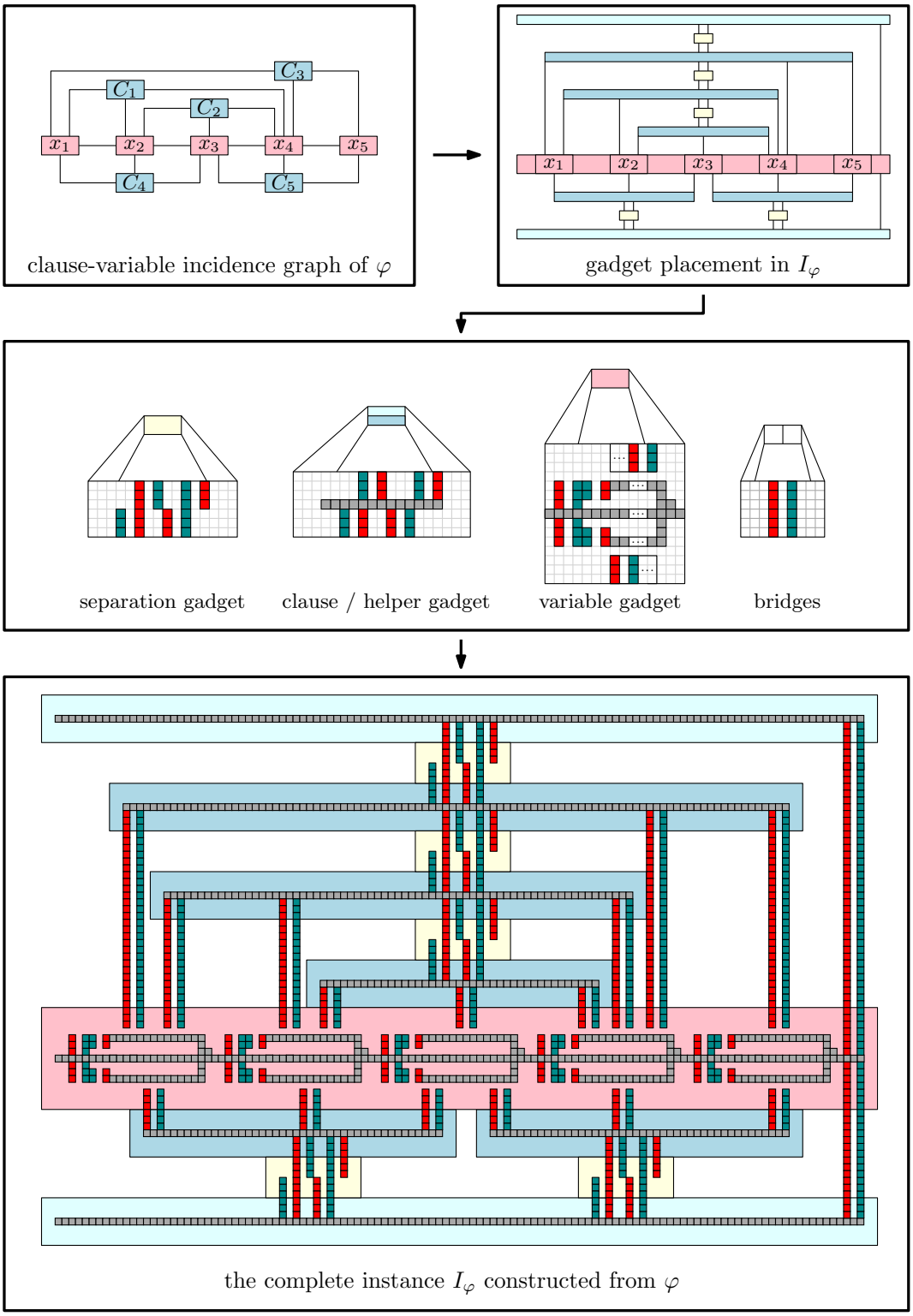


Figure 2 Symbolic overview of the NP-hardness reduction.

4 Bounded Stretch for Arbitrary Makespan

▶ **Theorem 4.** *There is a constant c such that for any pair of start and target configurations with a fatness of at least c, there is a stable schedule of constant stretch.*

On a high level, the overall schedule proceeds in four phases; see Figure 3 for an overview. In the preprocessing phase, we use a bottleneck matching algorithm for mapping the start configuration C_s to the target configuration C_t, minimizing the maximum distance d between a start and a target location. Furthermore, we establish the fatness in both configurations, set c to be the minimum of both fatness values, and compute a resulting set of *cd-tiles* that contain both C_s and C_t.

In the second phase, we build a scaffolding structure around C_s, based on the boundaries of *cd*-tiles, resulting in a *tiled configuration*, see Figure 3(a). This structure provides connectivity throughout the actual reconfiguration.

In the third phase, we perform the actual reconfiguration of the arrangement. This consists of refilling the tiles of the scaffolding structure, achieving the proper number of robots within each tile, based on elementary flow computations. As a subroutine, we transform the robots inside each tile into a canonical triangular configuration, see Figure 3(b), and Figures 5 to 7.

In the fourth and final phase, we disassemble the scaffolding structure and move the involved robots to their proper destinations, see Figure 3(c).

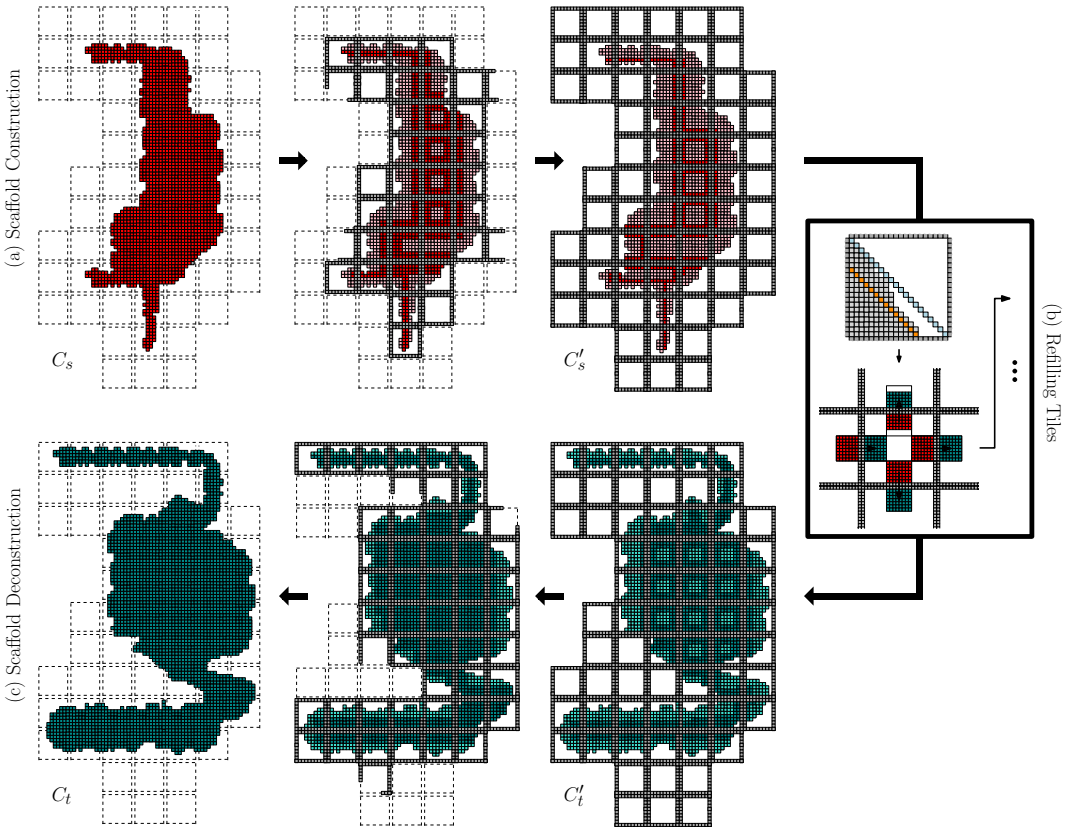


Figure 3 Overview of the computed schedule: (a) Constructing the scaffold: transforming the start configuration into a tiled configuration, (b) the refilling phase, and (c) deconstructing the scaffold: transforming the tiled configuration into the target configuration.

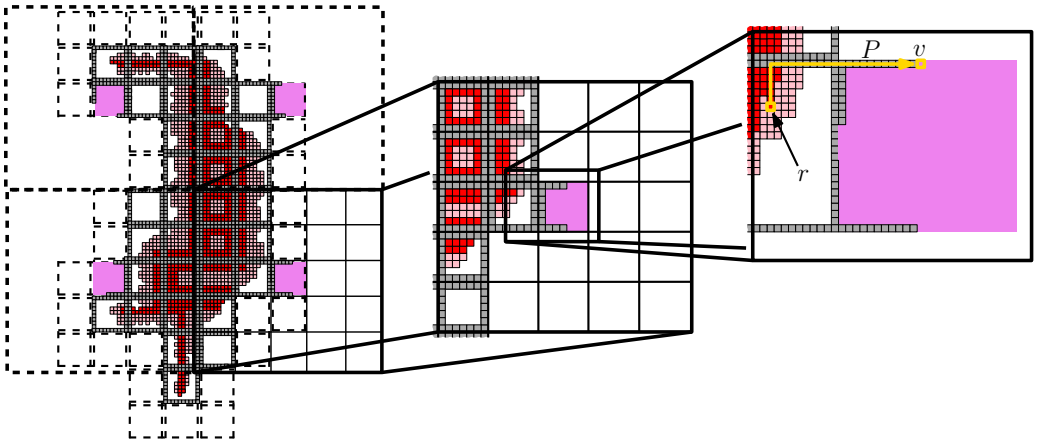


Figure 4 Constructing the scaffold. Tiles with currently constructed boundary are marked in pink, corresponding to one of the 25 tile classes. The zoom into the start configuration C_s shows the 5×5-neighborhood $N[T]$ of an active tile T (middle) and a further zoom into T with an associated robot motion (right). In each transformation step a robot from the interior of a tile $T' \in N[T]$ is swapped with a free position on the boundary of T based on a path P in a BFS-tree.

We will not go into detail regarding the preprocessing. Because disassembling the scaffold is the reverse of the building process, we will only describe the second and third phase:

Building the Scaffold. For the construction of the scaffold, we consider 25 different classes of tiles, based on x- and y-coordinates modulo $5cd$; see Figure 4. We process a single class as follows: For each tile T we consider its neighborhood $N[T]$ consisting of 5×5 tiles centered at T. Hence, the neighborhoods of different tiles of the same class are disjoint. For constructing the boundary of T, we make use of robots from the interior of a single tile in the neighborhood of T. In particular, we swap a free position on the boundary of T with an occupied position in the interior of a tile in $N[T]$, which is a leaf in a respective BFS-tree. Because the neighborhood of all tiles of the current class are disjoint, and a leaf cannot break connectivity, we obtain:

▶ **Lemma 5.** *There is a stable schedule within a makespan of $\mathcal{O}(d)$ transforming C_s into a tiled configuration C'_s, such that the interior of C'_s is a subset of the start configuration C_s.*

Reconfigure Single Tiles. We first compute $C_s \rightrightarrows_\chi C_s^m$ and $C_t \rightrightarrows_\chi C_t^m$, where C_s^m and C_t^m are monotone configurations. These reconfigurations are achieved by a specific sequence of down and left movements, maintaining connectivity after each move. Proceeding from these monotone configurations, the robots are arranged into a triangular configuration C_Δ that occupies the lower left positions (defined by a diagonal line with a slope of -1) of the interior of T. This is achieved by swapping pairs of occupied and empty positions within a carefully defined area in several one-step moves along L-shaped paths. The property of C_Δ is that it is the same for all initial configurations with equally many robots. Thus, to get the stable schedule $C_s \rightrightarrows_\chi C_\Delta \rightrightarrows_\chi C_t$ to reconfigure C_s into C_t, we can simply revert $C_t \rightrightarrows_\chi C_\Delta$ and combine the result with $C_s \rightrightarrows_\chi C_\Delta$; consider Figure 5 for illustration.

Because all distances are upper-bounded by $\mathcal{O}(d)$ and all robot movements stay within the interior of T, we can reconfigure all tiles in parallel to obtain the following:

▶ **Lemma 6.** *Let C'_s, C'_t be two tiled configurations such that C'_s and C'_t contain the same number of robots in the interior of T for each tile T. There is a stable schedule transforming C'_s into C'_t within a makespan of $\mathcal{O}(d)$.*

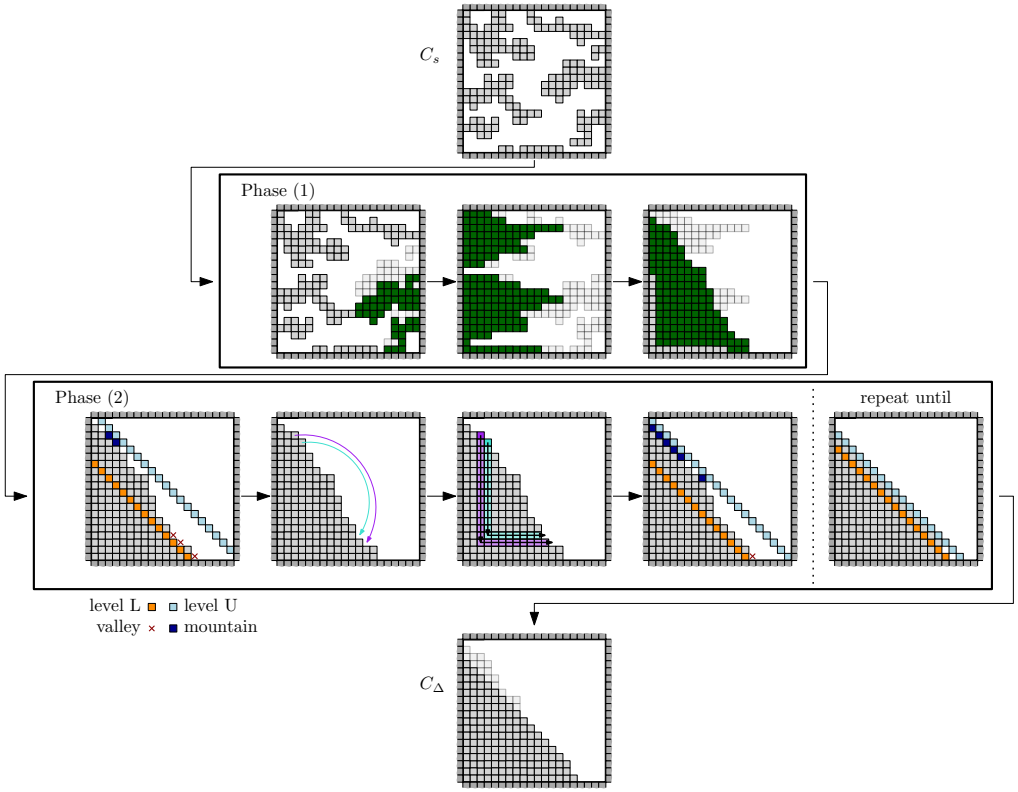


Figure 5 Turning arrangement C_s (top) into a canonical triangular configuration C_Δ (bottom). Phase (1) (left to right), achieves a monotonic arrangement; light grey indicates previous positions of active robots (shown in green). Phase (2) transforms the monotonic configuration into C_Δ.

Refilling Tiles. In general, tiles in C'_s and C'_t differ in the number of contained robots, so that we have to transfer robots between tiles. We model this robot transfer by a *supply and demand flow*, see Figure 6. By partitioning the flow into $\mathcal{O}(1)$ subflows, each subflow can be realized within a makespan of $\mathcal{O}(d)$. For realizing a single subflow, we use the reconfiguration of single tiles as a preprocessing step. In particular, we partition the interior of each tile T into nine *subtiles* with equal side lengths (up to rounding), see Figure 7. For each transfer path between T and T', we place the desired amount of robots inside the subtile of T that shares an edge with the boundary of T adjacent to T'. As mentioned, these configurations can be formed for all tiles in parallel in $\mathcal{O}(d)$, so that all robots can be moved into their respective target tiles. By repeating this approach, we obtain the following:

▶ **Lemma 7.** *We can efficiently compute a stable schedule transforming C'_s into C'_t within a makespan of $\mathcal{O}(d)$.*

The correctness of the approach (Theorem 4) follows from Lemmas 5, 6, and 7. As the diameter of the pair (C_s, C_t) is a lower bound for the makespan of any schedule transforming C_s into C_t, we obtain the following.

▶ **Corollary 8.** *There is a constant-factor approximation for computing stable schedules with minimal makespan between pairs of start and target configurations with a fatness of at least c, for some constant c.*

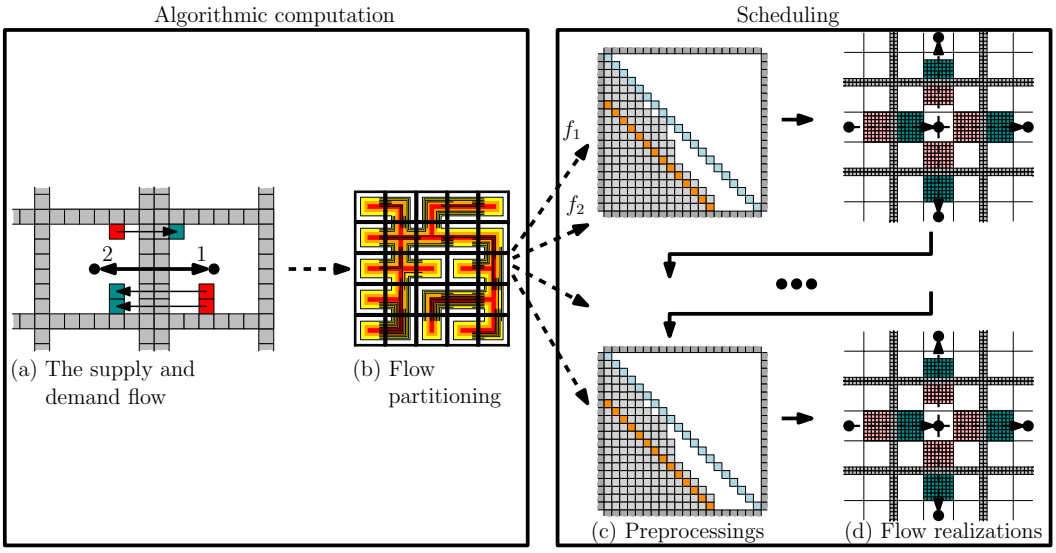


Figure 6 An overview of the schedule refilling tiles: transforming C'_s into C'_t by realizing a partition of a supply and demand flow that is computed in advance.

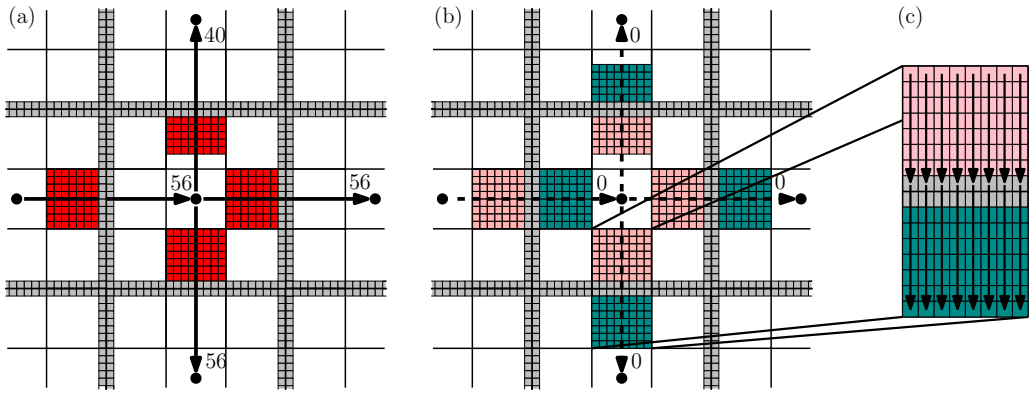


Figure 7 (a) A portion of a set of paths containing a vertex v: 56 paths are passing v from left to right, while $40 + 56 = 96$ paths are starting in v. (b) The configuration after realizing the set of paths shown in the previous figure. (c) How positions of robots have changed after realization.

5 Conclusion

We have shown that coordinated motion planning for a connected swarm of robots is a challenging problem, even in relatively simple cases. On the other hand, we have shown that (assuming sufficient connectivity of the swarm), it is possible to compute efficient reconfiguration schedules with constant stretch.

It is straightforward to extend our approach to other scenarios, e.g., to three-dimensional configurations. Other questions appear less clear. Can we show that constant stretch cannot be achieved for "thin" arrangements? Can we extend our methods to the labeled case? To what extent can the overall algorithm be turned into a set of distributed protocols, with only limited central computation and coordination?

References

1 Aaron T. Becker, Sándor P. Fekete, Phillip Keldenich, Matthias Konitzny, Lillian Lin, and Christian Scheffer. Coordinated motion planning: The video. In *Proc. Symposium on Computational Geometry (SoCG)*, pages 74:1–74:6, 2018. Video at `https://www.ibr.cs.tu-bs.de/users/fekete/Videos/CoordinatedMotionPlanning.mp4`.

2 Soon-Jo Chung, Aditya Avinash Paranjape, Philip Dames, Shaojie Shen, and Vijay Kumar. A survey on aerial swarm robotics. *IEEE Transactions on Robotics*, 34(4):837–855, 2018.

3 Daniel Delahaye, Stéphane Puechmorel, Panagiotis Tsiotras, and Eric Féron. Mathematical models for aircraft trajectory design: A survey. In *Air Traffic Management and Systems*, pages 205–247, 2014.

4 Erik D. Demaine, Sándor P. Fekete, Phillip Keldenich, Henk Meijer, and Christian Scheffer. Coordinated motion planning: Reconfiguring a swarm of labeled robots with bounded stretch. In *Proc. Symposium on Computational Geometry (SoCG)*, pages 29:1–29:17, 2018.

5 Erik D. Demaine, Sándor P. Fekete, Phillip Keldenich, Henk Meijer, and Christian Schefffer. Coordinated motion planning: Reconfiguring a swarm of labeled robots with bounded stretch. *SIAM Journal on Computing*, 48:1727–1762, 2019.

6 Sándor P. Fekete, Björn Hendriks, Christopher Tessars, Axel Wegener, Horst Hellbrück, Stefan Fischer, and Sebastian Ebers. Methods for improving the flow of traffic. In Christian Müller-Schloer, Hartmut Schmeck, and Theo Ungerer, editors, *Organic Computing — A Paradigm Shift for Complex Systems*, volume 1 of *Autonomic Systems*. Birkhäuser, 2011.

7 John E Hopcroft and Richard M Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973.

8 Michael Rubenstein, Alejandro Cornejo, and Radhika Nagpal. Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198):795–799, 2014.

9 Erol Šahin and Alan Winfield (editors). Special issue on swarm robotics. *Swarm Intelligence*, 2(2–4), 2008.

10 Michael Schreckenberg and Reinhard Selten (editors). *Human Behaviour and Traffic Networks*. Springer, 2004.

11 Jacob T. Schwartz and M. Sharir. On the piano movers' problem: III. Coordinating the motion of several independent bodies: the special case of circular bodies moving amidst polygonal barriers. *Int. J. Robotics Res.*, 2(3):46–75, 1983.

12 M. Turpin, K. Mohta, N. Michael, and V. Kumar. Goal assignment and trajectory planning for large teams of interchangeable robots. *Autonomous Robots*, 37(4):401–415, 2014.

Recognition and Reconfiguration of Lattice-Based Cellular Structures by Simple Robots

Amira Abdel-Rahman[1], Aaron T. Becker[2], Daniel E. Biediger[2], Kenneth C. Cheung[3], Sándor P. Fekete[4], Benjamin Jenett[1,3], Eike Niehs[4], Christian Scheffer[4], Arne Schmidt[4], and Mike Yannuzzi[2]

1 **Center for Bits and Atoms (CBA), Massachusetts Institute of Technology, Cambridge, MA, USA. bej@mit.edu, amira.abdel-rahman@cba.mit.edu**
2 **Department of Electrical and Computer Engineering, University of Houston, USA. {atbecker,dbiediger}@uh.edu**
3 **NASA Ames Research Center, Coded Structures Lab (CSL), Moffett Field, CA, USA. kenny@nasa.gov**
4 **Department of Computer Science, TU Braunschweig, Germany. {s.fekete, e.niehs, c.scheffer, arne.schmidt}@tu-bs.de**

Abstract

We consider recognition and reconfiguration of lattice-based cellular structures by very simple robots with only basic functionality. The underlying motivation is the construction and modification of space facilities of enormous dimensions, where the combination of new materials with extremely simple robots promises structures of previously unthinkable size and flexibility. We present algorithmic methods that are able to detect and reconfigure arbitrary polyominoes, based on finite-state robots, while also preserving connectivity of a structure during reconfiguration. Specific results include methods for determining a bounding box, scaling a given arrangement, and adapting more general algorithms for transforming polyominoes.

1 Introduction

Building and modifying large-scale structures is an important and natural objective in a vast array of applications. In many cases, the use of autonomous robots promises significant advantages, but also a number of additional difficulties, in particular when aiming for construction in orbit around earth. In recent years, a number of significant advances have been made to facilitate overall breakthroughs. One important step has been the development of ultra-light and scalable composite lattice materials [16] that allow the construction of modular, reconfigurable, lattice-based structures [18]. A second step has been the design of simple autonomous robots [17, 19] that are able to move on the resulting lattice structures and move their elementary cell components, thereby allowing the reconfiguration of the overall edifice.

In this paper, we address the next step in this hierarchy: Can we enable extremely simple robots to perform a more complex construction task for cellular structures in space, such as patrolling and marking the perimeter, scaling up a given seed construction, and a number of other design operations? As we demonstrate, even the extremely limited capabilities of machines with a finite number of states suffice for these tasks. In particular, we show that two robots suffice to construct the bounding box for a given arrangement, without losing connectivity. This is then used for other objectives, such as scaling up a given arrangement by a constant factor, as well as other, more general transformations.

1.1 Related Work

Assembly by simple robots has also been considered at the micro scale, where global control is used for supplying the necessary force for moving agents, e.g., see Becker et al. [3] for

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

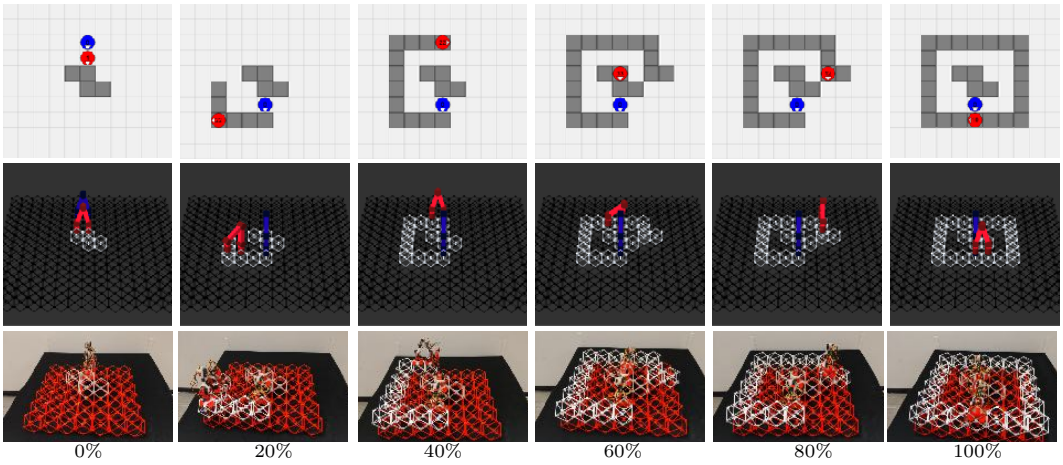


Figure 1 Snapshots from building a bounding box for a z-shaped polyomino using a 2D simulator, a 3D simulator, and staged hardware robots; shown are steps {0, 24, 48, 72, 96, 120}. See our video [1] for more context and animations.

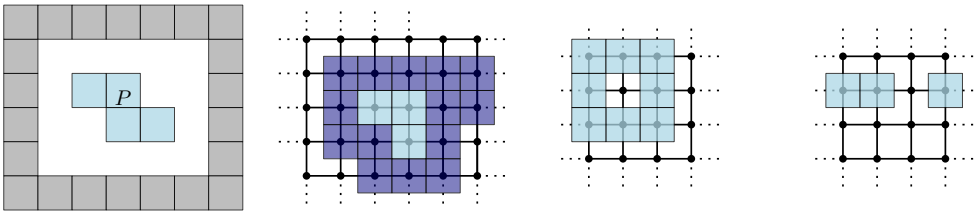


Figure 2 From left to right: Bounding box (gray) surrounding a polyomino (blue); the boundary ∂P (dark blue) of a shape P; a non-simple polyomino with one hole; a disconnected arrangement.

the corresponding problem of motion planning, Schmidt et al. [20] for using this model for assembling structures, and Balanza-Martinez et al. [2] for theoretical characterizations.

From an algorithmic view, we are interested in *different models representing programmable matter* and further recent results. Inspired by the single-celled amoeba, Derakhshandeh et al. introduced the Amoebot model [6, 10]. The Amoebot model provides a framework based on an equilateral triangular graph and active particles that can occupy a single vertex or a pair of adjacent vertices within that graph. Further related work to the amoebot model can be found in [4, 5, 7–9, 11]. In [15], Gmyr et al. introduced a model with two types of particles: active robots acting like a deterministic finite automaton and passive tile particles. Further results are shown in [14]. Fekete et al. [12] introduced more complex geometric algorithms for copying, reflecting, rotating, and scaling a given polyomino as well as an algorithm for constructing a bounding box surrounding a polyomino. However, their algorithms do not guarantee connectivity of intermediate arrangements. We build upon their model by allowing multiple robots working on the same grid environment.

2 Preliminaries

We consider an infinite *square grid graph* G, where $\mathbb{Z}^2$ defines the *vertices*, and for every two vertices with distance one there is a corresponding *edge* in G. We use the compass directions (N, E, S, W) for orientation when moving on the grid and may use *up, right, down*, and *left* synonymously.

Every vertex of G is either *occupied* by a tile or *unoccupied*. *Tiles* represent passive particles of programmable matter that cannot move or manipulate themselves. The maximal connected set of occupied vertices is called *polyomino*.

The *boundary* of a polyomino P is denoted by ∂P and includes all tiles of P that are (horizontally, vertically or diagonally) adjacent to an empty vertex (see also Figure 2). Polyominoes can have *holes*, i.e., finite maximal connected sets of empty vertices. Polyominoes without holes are called *simple*; otherwise, they are *non-simple*. The bounding box of a given polyomino P is defined as the boundary of the smallest rectangle enclosing P enlarged by one unit; it will be denoted by $bb(P)$ (see Figure 2).

We use *robots* as active particles in our model. These robots work like *finite deterministic automata* that can move around on the grid and manipulate the polyomino. A robot has the abilities to move along the edges of the grid graph and to change the state of the current vertex by placing or removing a tile on it. The robots work in a series of Look-Compute-Move (LCM) steps. Depending on the current state of the robot and the vertex it is positioned on (Look), the next step is computed according to a specific transition function δ (Compute), which determines the future state of robot and vertex, and the actual movement (Move). In the move phase, the robot can either remove a tile, place a tile (if there is not already a tile) or move to a adjacent vertex. *Moving* a tile is the same as deleting the tile at the start and placing a tile after the move sequence. In the case of multiple robots, we assume that they cannot be placed on the same vertex at the same time. Communication between robots is limited to adjacent vertices and can be implemented by expanding the Look phase by the states of all adjacent robots.

Connectivity is ensured if the union of all placed tiles and all used robots is connected. Accordingly, a robot can hold two components together (see blue robot in Figure 3).

3 Constructing a Bounding Box

In this section, we describe an algorithm to construct the bounding box while keeping connectivity of intermediate arrangements. Due to space constraints, we only sketch technical details; see the full version of the paper [13] for a full description. To accomplish the required connectivity, we specify, without any loss of generality, that the connection between $bb(P)$ and P must be on the south side of the boundary. For ease of presentation, the polyomino is shown in blue and the bounding box in gray; the robots cannot actually distinguish between those tiles. In the following, we assume that two robots are placed adjacent to each other on an arbitrary tile of the polyomino P, and that the first robot R_1 (shown in red in all figures) is the leader. As we will see, the second robot R_2 (blue in all figures) holds the polyomino and the bounding box together. In practice, we would rather use a special marker, called pebble, to mark a tile that holds P and $bb(P)$ together.

The construction can be split into three phases: (1) finding a start position, (2) constructing the bounding box, and (3) the clean-up. To find a suitable start position, we search for a locally y-minimal vertex that is occupied by a tile. This can be done by scanning the current row (i.e., moving left until we find an empty vertex and then moving right) and moving downwards whenever possible. Afterwards, R_1 starts the bounding box construction one vertex further down. This brings us to phase (2).

The construction of the bounding box is performed clockwise around P, i.e., whenever possible, R_1 makes a right turn. At some point, R_1 finds a tile either belonging to P or to the bounding box. To decide whether a tile t belongs to P or the current bounding box, we start moving around the boundary of the shape t belongs to. At some point, R_1 reaches R_2. If R_1 is above R_2 then t is a tile of P, otherwise t is a tile of the bounding box. To find

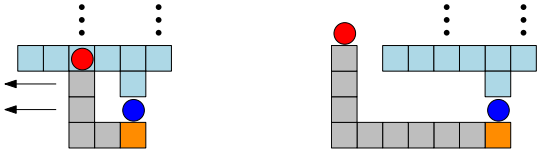


Figure 3 Left: R_1 (red) hits a tile belonging to P. Right: The triggered shifting process is finished.

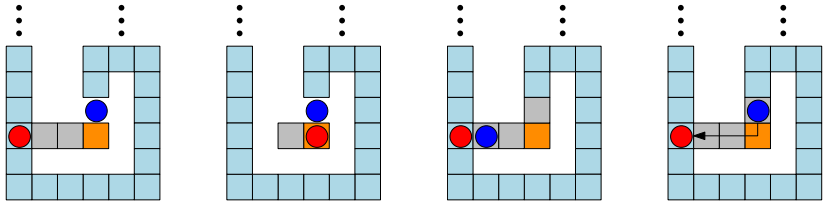


Figure 4 Traversing a gap by building a bridge. From left to right: R_1 finds a particle t not belonging to $bb(P)$. R_1 then picks up R_2, so both can move to t. Both, R_1 and R_2 reached t. Afterwards, R_2 deletes remaining pieces of the old bounding box.

t again, we move below R_2 and follow the construction until we cannot move any further. From there we can carry on building $bb(P)$.

Now, consider the two cases: If the tile does not belong to P, we are done with phase (2) and can proceed to phase (3). If it is a tile belonging to P, we need to shift the current line outwards until there is no more conflict, then continue the construction (see Figure 3). If the line to shift is the first line of the constructed bounding box, we know that there exists a tile of P that has the same y-coordinate than the current starting position. Therefore, we build a bridge to traverse this gap, as shown in Figure 4. Afterwards, we can restart from phase (1).

For phase (3), consider the case when R_1 reaches a tile from the bounding box. If the hit tile is not a corner tile, the current line needs to be shifted outwards until the next corner is reached (see Figure 5(a)). Then we can search for another suitable connection between P and $bb(P)$, place a tile there, and get to R_2 to remove unnecessary parts of the bounding box (see Figure 5(b)-(d)). To move to R_2, we move counterclockwise around $bb(P)$ until we find a tile with three adjacent tiles. From there we move north and follow the path until we find R_2. Because $bb(P)$ has only one tile with three adjacent tiles left, we can always find the connection between P and $bb(P)$.

▶ **Theorem 1.** *Given a polyominino P of width w and height h, building a bounding box surrounding P with the need that boundary and P are always connected, can be done with two robots in $O(\max(w,h) \cdot (wh + k \cdot |\partial P|))$ steps, where k is the number of convex corners in P.*

The proof of this theorem is analogous to that from [12]; see [13] for full details.

4 Scaling Polyominoes

Now we consider scaling a given shape by a factor c by building a scaled copy to the left of the bounding box. This copy process will be done column-wise from right to left. In the following we assume that the robot R_1 already built the bounding box and is positioned on one of its tiles.

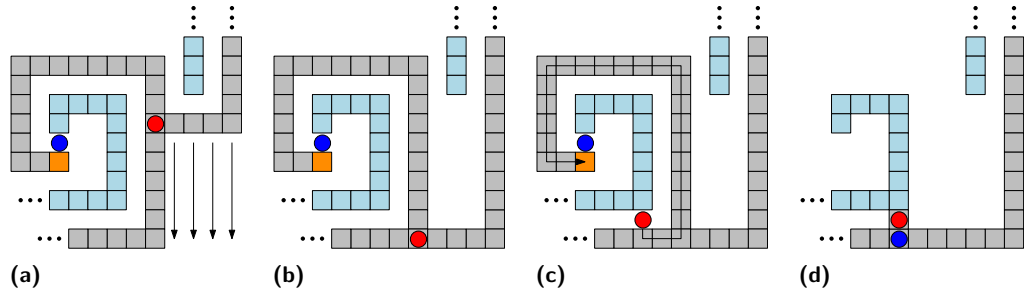


Figure 5 The second case of finishing the bounding box. (a) An already constructed part of the bounding box is hit. (b) The last boundary side is shifted. (c) R_1 found a suitable new connectivity vertex above the southern side, places a tile and retraces its path to the initial starting position. (d) The unnecessary part of the bounding box is removed and both robots catch up to the new connection.

4.1 Scaling

The scaling process can be divided into two phases: (1) the preparation phase, and (2) the scaling phase. In phase (1) we fill up the last column within $bb(P)$, add a tile in the second last column above the south side of $bb(P)$ and remove the lowest tile (called *column marker*) and third lowest tile (called *row marker*) on the east side of $bb(P)$ (see Figure 6). This gives us three columns within the bounding box (including $bb(P)$ itself). The first (from west to east) is the current column of P to scale. The second column, which is filled with tiles excepting the topmost row, is used to ensure connectivity and helps to recognize the end of the current column. The third column marks the current overall progress, i.e., we can find the tile in the correct current column and row that we want to scale next; we only have to move two steps to the west from the row marker to find the next vertex to scale.

In phase (2), we simply search for the vertex v to scale by moving to the column marker, then to the row marker, and afterwards moving two steps to the west. We then place the row marker one vertex upwards. For possible cases, see Figure 6. When we reach the top row of the bounding box, we move the column marker one vertex to the left and place a new row marker. Then we add a $c \times c$ square to the left of $bb(P)$, if v was occupied. If we did not move the column marker, we move left from the south side of $bb(P)$ until we reach an end and start moving up until we find the place to build the $c \times c$-square. If v was empty, then we leave out one tile within the square, e.g. the tile in the middle (see Figure 6 right).

After scaling a column that only contained empty vertices, we know that we are done with scaling. Thus, we can start removing all tiles, proceeding columnwise within $bb(P)$ from right to left. If necessary, all scaled empty tiles can also be removed by one scan through the scaled field. A formal proof can be found in the full version [13].

▶ **Theorem 2.** *After building $bb(P)$, scaling a polyomino P of width w and height h by a constant scaling factor c without loss of connectivity can be done with one robot in $O(wh \cdot (c^2 + cw + ch))$ steps.*

4.2 Adapting Algorithms

As shown in [12], there are algorithms that may not guarantee connectivity. An immediate consequence of being able to scale a given shape is that we can simulate any algorithm $\mathcal{A}$ within the Robot-on-Tiles model while guaranteeing connectivity: We first scale the polyomino by three and then execute $\mathcal{A}$ by always performing three steps into one direction if $\mathcal{A}$ does one step. If at some point the robot needs to move through empty vertices, then we

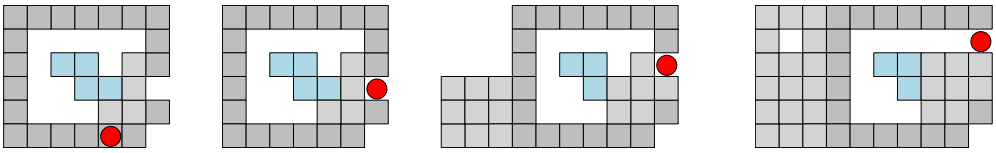


Figure 6 Left: Configuration after the preparation phase. Right three: Different states during phase (2): Scaling an occupied vertex, scaling an empty vertex, and reaching the end of a column.

place a 3×3-square with the middle vertex empty (if a clean up is desired at the end of $\mathcal{A}$, i.e., removing all scaled empty vertices, we fill up the complete row/column with these squares). This guarantees connectivity during the execution and we obtain the following theorem.

▶ **Theorem 3.** *If there is an algorithm $\mathcal{A}$ for some problem Π in the Robots-on-Tiles model with runtime $\mathcal{T}(\mathcal{A})$, such that the robot moves within a $w' \times h'$ rectangle, then there is an algorithm $\mathcal{A}'$ for Π with runtime $O(wh \cdot (c^2 + cw + ch) + \max((w' - w)h', (h' - h)w') + c \cdot \mathcal{T}(\mathcal{A}))$ guaranteeing connectivity during execution.*

5 Conclusion

We demonstrated how geometric algorithms for finite automata can be used to enable very simple robots to perform a number of fundamental but non-trivial construction tasks, such as building a bounding box and scaling a given shape by some constant, that guarantee connectivity between all tiles and robots during their execution.

Future work includes investigation of algorithms without the preceding bounding box construction (e.g. scaling) or distributed algorithms that do not rely on a scaling procedure.

References

1. A. Abdel-Rahman, A. T. Becker, D. E. Biediger, K. C. Cheung, S. P. Fekete, N. A. Gershenfeld, S. Hugo, B. Jenett, P. Keldenich, E. Niehs, C. Rieck, A. Schmidt, C. Scheffer, and M. Yannuzzi. Space ants: Constructing and reconfiguring large-scale structures with finite automata. 2020. Video at `https://www.ibr.cs.tu-bs.de/users/fekete/Videos/Space_submit.mp4`.
2. J. Balanza-Martinez, A. Luchsinger, D. Caballero, R. Reyes, A. A. Cantu, R. Schweller, L. A. Garcia, and T. Wylie. Full tilt: universal constructors for general shapes with uniform external forces. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2689–2708, 2019.
3. A. T. Becker, S. P. Fekete, P. Keldenich, D. Krupke, C. Rieck, C. Scheffer, and A. Schmidt. Tilt assembly: algorithms for micro-factories that build objects with uniform external forces. *Algorithmica*, pages 1–23, 2017.
4. J. J. Daymude, R. Gmyr, K. Hinnenthal, I. Kostitsyna, C. Scheideler, and A. W. Richa. Convex hull formation for programmable matter, 2018.
5. J. J. Daymude, R. Gmyr, A. W. Richa, C. Scheideler, and T. Strothmann. Improved leader election for self-organizing programmable matter. In A. Fernández Anta, T. Jurdzinski, M. A. Mosteiro, and Y. Zhang, editors, *Algorithms for Sensor Systems*, pages 127–140, Cham, 2017. Springer International Publishing.
6. Z. Derakhshandeh, S. Dolev, R. Gmyr, A. W. Richa, C. Scheideler, and T. Strothmann. Brief announcement: Amoebot – a new model for programmable matter. In *Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '14, pages 220–222, New York, NY, USA, 2014. ACM.

7 Z. Derakhshandeh, R. Gmyr, A. Porter, A. W. Richa, C. Scheideler, and T. Strothmann. On the runtime of universal coating for programmable matter. In Y. Rondelez and D. Woods, editors, *DNA Computing and Molecular Programming*, pages 148–164, Cham, 2016. Springer International Publishing.

8 Z. Derakhshandeh, R. Gmyr, A. W. Richa, C. Scheideler, and T. Strothmann. An algorithmic framework for shape formation problems in self-organizing particle systems. In *Proceedings of the 2nd Annual International Conference on Nanoscale Computing and Communication*, NANOCOM' 15, pages 21:1–21:2, New York, NY, USA, 2015. ACM.

9 Z. Derakhshandeh, R. Gmyr, A. W. Richa, C. Scheideler, and T. Strothmann. Universal coating for programmable matter. *CoRR*, abs/1601.01008, 2016.

10 Z. Derakhshandeh, R. Gmyr, T. Strothmann, R. Bazzi, A. W. Richa, and C. Scheideler. Leader election and shape formation with self-organizing programmable matter. In A. Phillips and P. Yin, editors, *DNA Computing and Molecular Programming*, pages 117–132, Cham, 2015. Springer International Publishing.

11 G. A. Di Luna, P. Flocchini, N. Santoro, G. Viglietta, and Y. Yamauchi. Shape formation by programmable particles. *CoRR*, abs/1705.03538, 2017.

12 S. P. Fekete, R. Gmyr, S. Hugo, P. Keldenich, C. Scheffer, and A. Schmidt. Cadbots: Algorithmic aspects of manipulating programmable matter with finite automata. *CoRR*, abs/1810.06360, 2018.

13 S. P. Fekete, E. Niehs, C. Scheffer, and A. Schmidt. Connected assembly and reconfiguration by finite automata. 2019.

14 R. Gmyr, K. Hinnenthal, I. Kostitsyna, F. Kuhn, D. Rudolph, C. Scheideler, and T. Strothmann. Forming tile shapes with simple robots. In D. Doty and H. Dietz, editors, *DNA Computing and Molecular Programming*, pages 122–138, Cham, 2018. Springer International Publishing.

15 R. Gmyr, I. Kostitsyna, F. Kuhn, C. Scheideler, and T. Strothmann. Forming tile shapes with a single robot. In *33rd European Workshop on Computational Geometry (EuroCG 2017)*, pages 9–12, 2017.

16 C. E. Gregg, J. H. Kim, and K. C. Cheung. Ultra-light and scalable composite lattice materials. *Advanced Engineering Materials*, 20(9):1800213, 2018.

17 B. Jenett and D. Cellucci. A mobile robot for locomotion through a 3D periodic lattice environment. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5474–5479, 2017.

18 B. Jenett, D. Cellucci, C. Gregg, and K. Cheung. Meso-scale digital materials: modular, reconfigurable, lattice-based structures. In *ASME 2016 11th International Manufacturing Science and Engineering Conference*. American Society of Mechanical Engineers Digital Collection, 2016.

19 B. Jenett and K. Cheung. Bill-e: Robotic platform for locomotion and manipulation of lightweight space structures. In *25th AIAA/AHS Adaptive Structures Conference*, page 1876, 2017.

20 A. Schmidt, S. Manzoor, L. Huang, A. T. Becker, and S. P. Fekete. Efficient parallel self-assembly under uniform control inputs. *IEEE Robotics and Automation Letters*, 3(4):3521–3528, 2018.

Targeted Drug Delivery: Algorithmic Methods for Collecting a Swarm of Particles with Uniform, External Forces

Aaron T. Becker[1], Sándor P. Fekete[2], Li Huang[1], Phillip Keldenich[2], Linda Kleist[2], Dominik Krupke[2], Christian Rieck[2], and Arne Schmidt[2]

1 **Department of Electrical and Computer Engineering, University of Houston, USA.** {atbecker,lhuang28}@uh.edu

2 **Department of Computer Science, TU Braunschweig, Germany.** {s.fekete, p.keldenich, l.kleist, d.krupke, c.rieck, arne.schmidt}@tu-bs.de

Abstract

We investigate algorithmic approaches for targeted drug delivery in a complex, maze-like environment, such as a vascular system. The basic scenario is given by a large swarm of micro-scale particles ("agents") and a particular target region ("tumor") within a system of passageways. Agents are too small to contain on-board power or computation and are instead controlled by a global external force that acts uniformly on all particles, such as an applied fluidic flow or electric field. The challenge is to deliver all agents to the target region with a minimum number of actuation steps. We provide a number of results for this challenge. We show that the underlying problem is NP-hard, which explains why previous work did not provide provably efficient algorithms. We also develop a number of algorithmic approaches that greatly improve the worst-case guarantees for the number of required actuation steps.

1 Introduction

A crucial challenge for a wide range of vital medical problems, such as the treatment of cancer, localized infections and inflammation, or internal bleeding is to deliver active substances to a specific location in an organism. The traditional approach of administering a sufficiently large supply of these substances into the circulating blood may cause serious side effects, as the outcome intended for the target site may also occur in other places, with often undesired, serious consequences. Moreover, novel custom-made substances that are specifically designed for precise effects are usually in too short supply to be generously poured into the blood stream. In the context of targeting brain tumors (see Figure 1), an additional difficulty is the blood-brain barrier. This makes it necessary to develop other, more focused methods for delivering agents to specific target regions.

Given the main scenario of medical applications, this requires dealing with navigation through complex vascular systems, in which access to a target location is provided by pathways (in the form of blood vessels) through a maze of obstacles. However, the microscopic size of particles necessary for passage through these vessels makes it prohibitively difficult to store sufficient energy in suitably sized microrobots, in particular in the presence of flowing blood.

A promising alternative is offered by employing a global external force, e.g., a fluidic flow or an electromagnetic field. When such a force is applied, all particles are subjected to the same direction and distance of motion, unless they are blocked by obstacles in their way. While this makes it possible to move all particles at once, it introduces the difficulty of using *uniform* forces for many particles in *different* locations with different local topology to

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

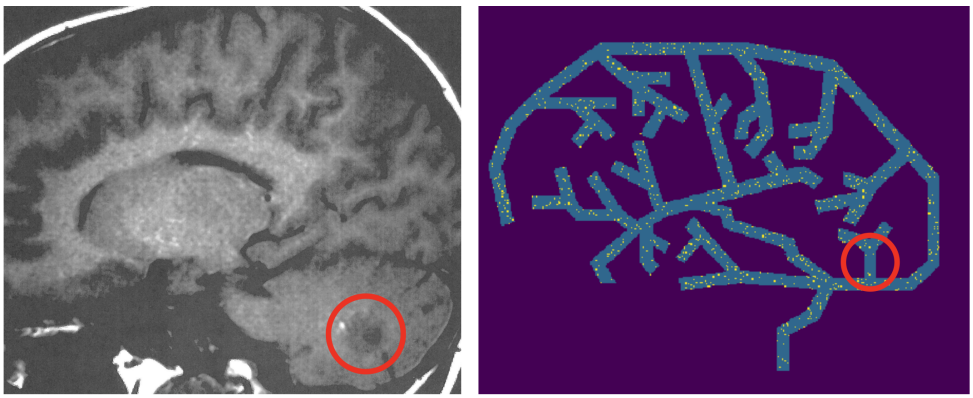


Figure 1 (Left) An MRI image of a brain tumor (red circle), located in the cerebellum. (Right) How can the swarm of particles (yellow dots) be delivered to the target region?

navigate them to *one* final destination. In this paper, we investigate how this objective can be achieved with a small number of actuator steps.

Previous work [11] described a basic approach that delivers all particles in a polyomino with n pixels to a target in at most $O(n^3)$ actuator steps. While a delivery time of this magnitude is usually impractical, we investigate how to improve this.

Our Contribution.

- We prove that minimizing the length of a command sequence for gathering all particles is NP-hard, even if the environments are modeled by polyominoes. Our reduction implies hardness for the related localization problem (as explained in Section 3 before Corollary 3.2).
- We develop an algorithmic strategy for gathering all particles in a polyomino with a worst-case guarantee of at most $O(kD^2)$ steps; here D denotes the maximum distance between any two pixels of the polyomino and k the number of its convex corners. Both k and D are usually much smaller than the number n of grid locations in the polyomino: n may be in $\Omega(D^2)$, for two-dimensional and in $\Omega(D^3)$ for three-dimensional environments.
- For the special case of hole-free polyominoes, we can gather all particles in $O(kD)$ steps.

Further details and algorithmic studies can be found in [7].

1.1 Related Work

This paper seeks to understand control for large numbers of microrobots, and uses a generalized model that could apply to a variety of drug-carrying microparticles. An example are particles with a magnetic core and a catalytic surface for carrying medicinal payloads [10, 15]. An alternative are aggregates of *superparamagnetic iron oxide microparticles*, 9 μm particles that are used as a contrast agent in MRI studies [14]. Real-time MRI scanning can allow feedback control using the location of a swarm of these particles.

Steering magnetic particles using the magnetic gradient coils in an MRI scanner was implemented in [12, 15]. 3D Maxwell-Helmholtz coils are often used for precise magnetic field control [14]. Still needed are motion planning algorithms to guide the swarms of robots through vascular networks. To this end, we build on the techniques for controlling many simple robots with uniform control inputs presented in [4–6]; see video and abstract [3] for a visualizing overview. For a recent survey on challenges related to controlling multiple microrobots (less than 64 robots at a time), see [8].

As the underlying problem consists of bringing together a number of agents in one location, a highly relevant algorithmic line of research considers *rendezvous search*, which requires two or more independent, intelligent agents to meet [1, 2, 9, 13].

2 Preliminaries

The "robots" in this paper are simple particles without autonomy. Every environment is modeled by a *polyomino*, i.e., a set of unit squares, so called *pixels*, in the plane which are joined edge to edge. An example of a polyomino is illustrated in Figure 2. Pixels in the plane not belonging to P are *blocked* because they stop the motion from an adjacent pixel. The particles are commanded in unison: In each step, all particles are relocated by one unit in one of the directions "Up" (u), "Down" (d), "Left" (l), or "Right" (r), unless the destination is a blocked pixel; in this case, a particle remains in its previous pixel. A motion plan is a command sequence $C = \langle c_1, c_2, c_3, \dots \rangle$, where each command $c_i \in \{u, d, l, r\}$.

We assume that the size of a particle is insignificant compared to a pixel. Hence, many of them can be located in the same pixel. During the course of a command sequence, two particles π_1 and π_2 may end up in the same pixel p, if π_1 moves into p, while π_2 remains in p due to a blocked pixel. Once two particles share a pixel, any subsequent command will relocate them in unison—they will not be separated, so they can be considered to be *merged*.

The *distance* between two pixels p and q is the length of a shortest path on the integer grid between p and q that stays within P. The *diameter* of a polyomino P describes the maximum distance between any two of its pixels; we denote it by D. A *configuration* of P is a set of pixels containing at least one particle. The set of all possible configurations of P is denoted by $\mathcal{P}$. We call a command sequence *gathering* if it transforms a configuration $A \in \mathcal{P}$ into a configuration A' such that $|A'| = 1$, i.e., if it merges all particles in the same pixel.

3 Hardness

We show that the following decision problem, which we call Min-Gathering, is hard: Given a polyomino P and a set of particles, is there a gathering sequence of length ℓ?

▶ **Theorem 3.1.** Min-Gathering *is NP-hard.*

Proof-Sketch. The proof is based on a reduction from 3-Sat. For every instance Φ of 3-Sat, we construct a polyomino P_Φ of diameter D containing a particle in every pixel such that there exists a gathering sequence of length $\ell := \frac{1}{2}(D + b)$ if and only if Φ is satisfiable.

P_Φ is constructed as follows, see Figure 2: For every variable, we insert a variable gadget. We join all variable gadgets vertically in a row to a *variable block*; we call the top row of each variable gadget its *variable row*. For every clause, we construct a clause gadget that contains a left (right) *literal arm* for each incident positive (negative) literal in the corresponding variable row and an exit arm in the bottom. To obtain P_Φ, we join all clause gadgets from left to right by a *bottom row* and insert a variable block at the left and right end of the bottom row of length b. Note that b denotes the number of pixels in the bottom row of P_Φ, and that the distance between the two *red* particles (i.e., the two leftmost particles above the variable blocks) realizes the diameter D.

We can argue that by applying a command sequence according to a satisfying assignment for Φ, the left (right) red particle moves to the left (right) pixel of the bottom row, where these particles can be merged, yielding a gathering sequence of length ℓ. Note that in this command sequence, particles in the clause gadgets traverse one of the literal arms, reaching the bottom row at the exact same time.

Conversely, if Φ is not satisfiable, then in at least one clause a particle uses the exit arm (see blue particle). Therefore, due to the different heights of the exit arms of the variable and the clause gadgets, particles remain in the exit arms and do not reach the bottom row simultaneously with the red particles. Thus, there is no gathering sequence of length ℓ. ◀

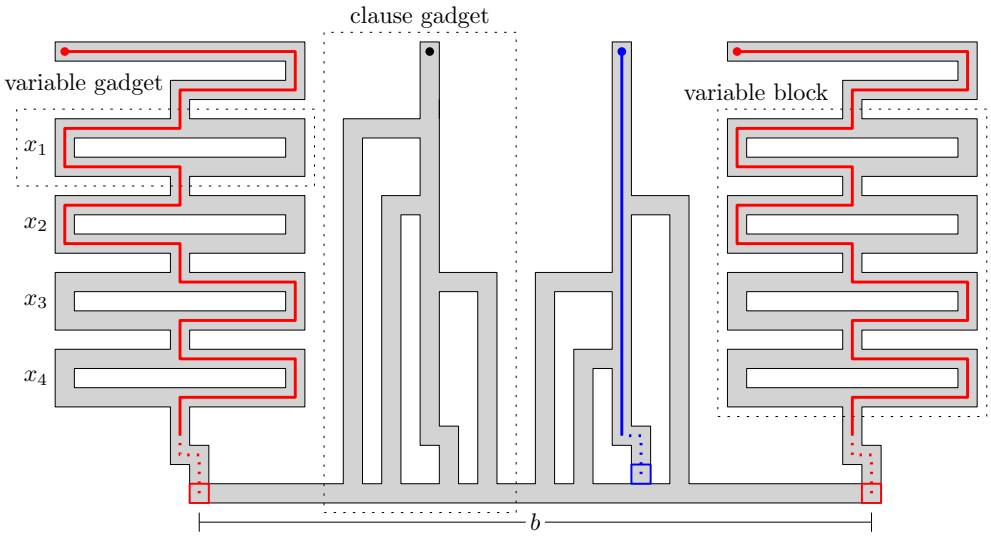


Figure 2 The polyomino P_Φ for the 3-SAT-instance $\Phi = (x_1 \vee x_2 \vee \overline{x}_3) \wedge (\overline{x}_2 \vee x_3 \vee x_4)$. A gathering sequence for the two red particles of length $\frac{1}{2}(D+b)$ corresponds to a variable assignment of Φ.

Note that the left pixel of the bottom row is one of two possible merge location for a gathering sequence of length $\frac{1}{2}(D+b)$. Therefore, the same reduction shows that the problem remains hard if a target location is prescribed. Because every pixel contains a particle, this implies that the decision problem of ROBOT LOCALIZATION is also hard. In an instance of this problem, we are given a sensorless robot r in a polyomino, and wonder whether there exists a command sequence of length ℓ such that we know the position of r afterwards. The above observations yield the following.

▶ **Corollary 3.2.** ROBOT LOCALIZATION *is NP-hard.*

4 Algorithmic Approaches

We start by merging two particles in a special class of polyominoes. A polyomino P is *simple* (or *hole-free*) if decomposing P with horizontal lines through pixel edges results in a set of rectangles $\mathcal{R}$ such that the edge-contact graph $\mathcal{C}(\mathcal{R})$ of $\mathcal{R}$ is a tree. The edge-contact graph of $\mathcal{R}$ has a vertex for each rectangle and an edge for each side contact; a point contact does not suffice.

▶ **Theorem 4.1.** *For any two particles in a simple polyomino P with diameter D, there exists a gathering sequence of length D.*

Proof-Sketch. Let $\mathcal{R}$ be a decomposition of P into rectangles by cutting P with horizontal lines through pixel edges. Then, because P is simple, the edge-contact graph $\mathcal{C}(\mathcal{R})$ of the rectangles $\mathcal{R}$ is a tree. For an example, consider Figure 3.

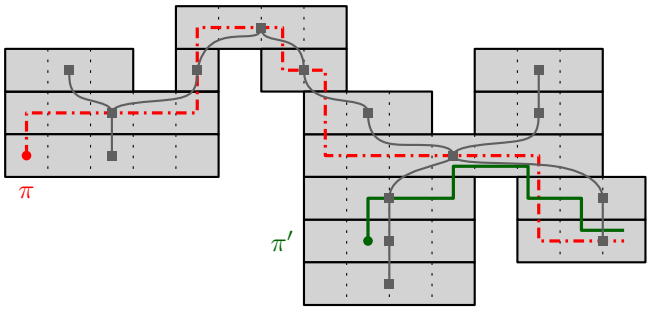


Figure 3 A simple polyomino P, and its edge-contact graph $\mathcal{C}(\mathcal{R})$ (in gray). When the red particle π moves towards the green particle π', π and π' follow the red and the green path, respectively.

For every t, let R_t and R'_t be the rectangles of P containing the two particles π and π' after applying t commands, respectively. Moreover, let S_t be a shortest path from R_t to R'_t in $\mathcal{C}(\mathcal{R})$. Moreover, let $S_t(1)$ be the successor of R_t on S_t (if it exists, i.e., $R_t \neq R'_t$).

We use the following strategy.

Phase 1: While $R_t \neq R'_t$, compute a shortest path S_t from R_t to R'_t in $\mathcal{C}(\mathcal{R})$. Move π to $S_t(1)$ via a shortest path in P. Update R_t and R'_t.

Phase 2: If $R_t = R'_t$, move π towards π' by a shortest (horizontal) path; note that this gathering sequence merges the particles within R_t.

In fact, the resulting sequence has the following property: For every $s > t$, the rectangles R_s and R'_s are either equal to R_t or lie in the connected component C of $\mathcal{C}(\mathcal{R} \setminus R_t)$ containing R'_t. This implies that the merge location and R'_t lie in C or are equal to R_t. Consequently, in every step, π moves towards the merge location on a shortest path and thus the gathering sequence is at most of length D. ◀

We call the strategy used to prove Theorem 4.1 DynamicShortestPath (DSP): Move one particle towards the other along a shortest path; update the shortest path if a shorter one exists. The example in Figure 4 shows that DSP may perform significantly worse in non-simple polyominoes, i.e., it may not yield a gathering sequence of length $O(D)$.

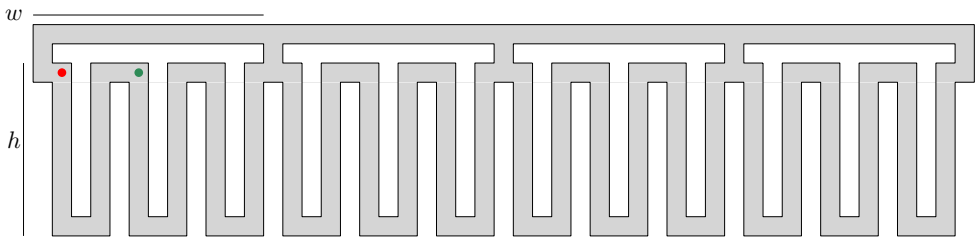


Figure 4 When the red particle π moves towards the green particle π' by shortest paths, π visits the entire bottom path.

Nevertheless, DSP always merges two particles: When a particle π follows π' in a polyomino with n pixels, then within n commands either the shortest path is updated or π' must meet a wall. Therefore, for every n commands, the distance between the particles decreases by at least 1. Therefore, the following holds true.

▶ **Proposition 4.2.** *For every polyomino P with n pixels and diameter D and every configuration with two particles, DSP yields a gathering sequence of length $O(nD)$.*

Using a different strategy yields a better bound.

▶ **Theorem 4.3.** *For any two particles in a polyomino P, there exists a gathering sequence of length at most D^2.*

Proof. Let q be the top-rightmost pixel of P. To merge the two particles in q, our strategy is as follows: Identify the particle π that is bottom-leftmost. Apply a command sequence that moves π to q on a shortest path. Repeat.

▶ **Claim.** *In each iteration, the sum of the distances Δ of the two particles to q decreases.*

Note that Δ decreases when the other particle π' has a collision. If π' had no collision, there exist a pixel that is higher or more to the right than q, contradicting the choice of q. Consequently, the sum of distances Δ, which is at most $2D$ at start, decreases at least by 1 for every D steps. Hence after $O(D^2)$ steps, Δ is reduced to 0. ◀

Note that there exist polyominoes, e.g., a square, where the number n of pixels is in $\Omega(D^2)$. Therefore, Theorem 4.3 significantly improves the bound of $O(n^3)$ in [11]. Finally, we note that a shortest gathering sequence for two particles in a non-simple polyomino may need to exceed D; Figure 5 illustrates the non-simple polyomino used to obtain Proposition 4.4.

▶ **Proposition 4.4.** *There exists a non-simple polyomino P with two particles such that a shortest gathering sequence has length $3/2D - O(\sqrt{D})$.*

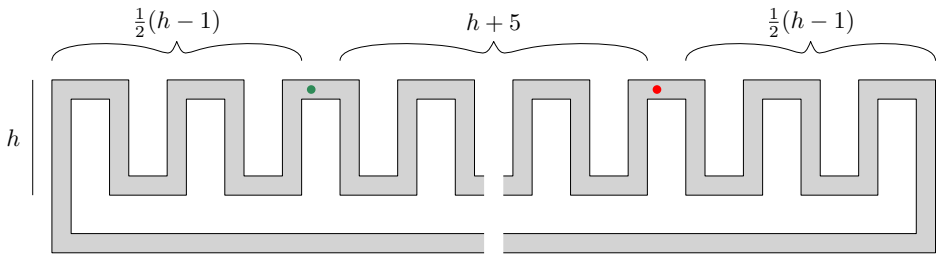


Figure 5 Merging the two particles in this non-simple polyomino needs to exceed D.

In the following, we show how to guarantee with few commands that the number of remaining particles is proportional to the complexity of the polyomino, namely the number of its convex corners.

▶ **Lemma 4.5.** *Let P be a polyomino with diameter D and k convex corners. For every configuration $A \in \mathcal{P}$, there exists a command sequence of length $2D$ which transforms A to a configuration $A' \in \mathcal{P}$ such that $|A'| \leq k/4$.*

Proof. We distinguish four types of convex corners; northwest (NW), northeast (NE), southwest (SW), southeast (SE). By the pigeon hole principle, one of the types occurs at most $k/4$ times; without loss of generality, let this be the NW corners.

We show that after applying the sequence $\langle l, u\rangle^D$, every particle lies in a NW corner: Consider a particle π in pixel p. Unless π lies in a NW corner, it moves for at least one command in $\{l, u\}$. Because P is finite, there exists an ℓ large enough such that π ends in a NW corner q when the command sequence $\langle l, u\rangle^\ell$ is applied, i.e., there exists an pq-path consisting of at most ℓ commands of types l and u, respectively. Because a monotone path is a shortest path, it holds that $\ell \leq D$. ◀

By combining Lemma 4.5 with Theorem 4.1 and Theorem 4.3, respectively, we obtain the following upper bounds.

▶ **Corollary 4.6.** *For a set of particles in a simple polyomino P with diameter D and k convex corners, there exists a gathering sequence of length $O(kD)$.*

▶ **Corollary 4.7.** *For any set of particles in a polyomino P with diameter D and k convex corners, there exists a gathering sequence of length at most $O(kD^2)$.*

References

1. Steve Alpern and Shmuel Gal. *The theory of search games and rendezvous*. International Series in Operations Research and Management Science. Kluwer Academic Publishers, Boston, Dordrecht, London, 2003.
2. Edward J Anderson and Sándor P Fekete. Two dimensional rendezvous search. *Operations Research*, 49(1):107–118, 2001.
3. A. T. Becker, Erik D. Demaine, Sándor P. Fekete, S. H. Mohtasham Shad, and R. Morris-Wright. Tilt: The video. Designing worlds to control robot swarms with only global signals. In *31st International Symposium on Computational Geometry (SoCG)*, pages 16–18, 2015.
4. Aaron T. Becker, Erik D. Demaine, S. P. Fekete, and James McLurkin. Particle computation: Designing worlds to control robot swarms with only global signals. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 6751–6756, 2014.
5. Aaron T. Becker, Erik D. Demaine, Sàndor P. Fekete, Golnaz Habibi, and James McLurkin. Reconfiguring massive particle swarms with limited, global control. In *Algorithms for Sensor Systems (ALGOSENSORS)*, pages 51–66, 2014.
6. Aaron T. Becker, Erik D. Demaine, Sándor P. Fekete, Jarrett Lonsford, and Rose Morris-Wright. Particle computation: Complexity, algorithms, and logic. *Natural Computing*, 18(1):181–201, 2019.
7. Aaron T. Becker, Sándor P. Fekete, Li Huang, Phillip Keldenich, Linda Kleist, Dominik Krupke, Christian Rieck, and Arne Schmidt. Targeted Drug Delivery: Algorithmic Methods for Collecting a Swarm of Particles with Uniform, External Forces. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020. To appear.
8. Sagar Chowdhury, Wuming Jing, and David J. Cappelleri. Controlling multiple microrobots: recent progress and future challenges. *Journal of Micro-Bio Robotics*, 10(1-4):1–11, 2015.
9. Paola Flocchini. Gathering. In Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro, editors, *Distributed Computing by Mobile Entities, Current Research in Moving and Computing*, pages 63–82. Springer, 2019.
10. Julia Litvinov, Azeem Nasrullah, Timothy Sherlock, Yi-Ju Wang, Paul Ruchhoeft, and Richard C Willson. High-throughput top-down fabrication of uniform magnetic particles. *PloS one*, 7(5):e37440, 2012.
11. Arun V. Mahadev, Dominik Krupke, Jan-Marc Reinhardt, Sándor P. Fekete, and Aaron T. Becker. Collecting a swarm in a 2D environment using shared, global inputs. In *13th Conference on Automation Science and Engineering (CASE)*, pages 1231–1236, 2016.
12. Jean-Baptiste Mathieu and Sylvain Martel. Magnetic microparticle steering within the constraints of an MRI system: proof of concept of a novel targeting approach. *Biomedical microdevices*, 9(6):801–808, 2007.
13. Malika Meghjani and Gregory Dudek. Multi-robot exploration and rendezvous on graphs. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5270–5276, 2012.
14. Lyes Mellal, David Folio, Karim Belharet, and Antoine Ferreira. Magnetic microbot design framework for antiangiogenic tumor therapy. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1397–1402, 2015.

15 Pierre Pouponneau, Jean-Christophe Leroux, and Sylvain Martel. Magnetic nanoparticles encapsulated into biodegradable microparticles steered with an upgraded magnetic resonance imaging system for tumor chemoembolization. *Biomaterials*, 30(31):6327–6332, 2009.

Coordinated Particle Relocation Using Finite Static Friction with Boundary Walls*

Victor M. Baez[1], Aaron T. Becker[1], Sándor P. Fekete[2], and Arne Schmidt[2]

1 **Department of Electrical & Computer Engineering, University of Houston, USA† {vjmontan, atbecker}@uh.edu**
2 **Department of Computer Science, TU Braunschweig, 38106 Braunschweig, Germany. {s.fekete, arne.schmidt}@tu-bs.de**

Abstract

We present methods for achieving *arbitrary* reconfiguration of two particles in convex workspaces, based on the use of external forces, such as a magnetic field or gravity. This concept can be used for a wide range of applications in which particles do not have their own energy supply.

A crucial challenge for achieving any desired target configuration is breaking global symmetry in a controlled fashion. Previous work made use of specifically placed barriers; however, introducing precisely located obstacles into the workspace is impractical for many scenarios. In this paper, we present a different, less intrusive method: making use of the interplay between static friction with a boundary and the external force to achieve arbitrary reconfiguration. Our key contributions are a precise characterization of the critical coefficient of friction that is sufficient for rearranging two particles in triangles, convex polygons, and regular polygons.

1 Introduction

Reconfiguring a large set of objects in a prespecified manner is a fundamental task for a large spectrum of applications, including swarm robotics, smart materials and advanced manufacturing. In many of these scenarios, the involved items are not equipped with individual motors or energy supplies, so actuation must be performed from the outside. Moreover, reaching into the workspace to manipulate individual particles of an arrangement is often impractical or even impossible; instead, global external forces (such as gravity or a magnetic force) may have to be employed, targeting each object in the same, uniform manner. These limitations of individual navigation apply even in scenarios of swarm robotics: For example, the well-known kilobots do have individual actuation and energy supply, but often make use of an external light source for navigation [10]; as a consequence, directing a swarm of kilobots by switching on a light beacon works just like activating an external force. This concept of global control has also been studied for using biological cells as reactive robots controlled by magnetic fields [2, 8]. Global control also has applications in assembling nano- and micro-structures. Related work shows how to assemble shapes by adding one particle at a time [7, 4], or combining multiple pairs of subassemblies in parallel in one time step [12].

Considering this approach of navigation by a global external force gives rise to a number of problems, including navigation of one particle from a start to a goal position [9], particle computation [5, 6], or emptying a polygon [1]. Zhang et al. [15, 16] show how to rearrange a rectangle of agents in a workspace that is only constant times larger than the number of agents.

* A video showing context and animations of our results can be found in [3].
† Work from these authors was partially supported by National Science Foundation IIS-1553063 and IIS-1619278.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

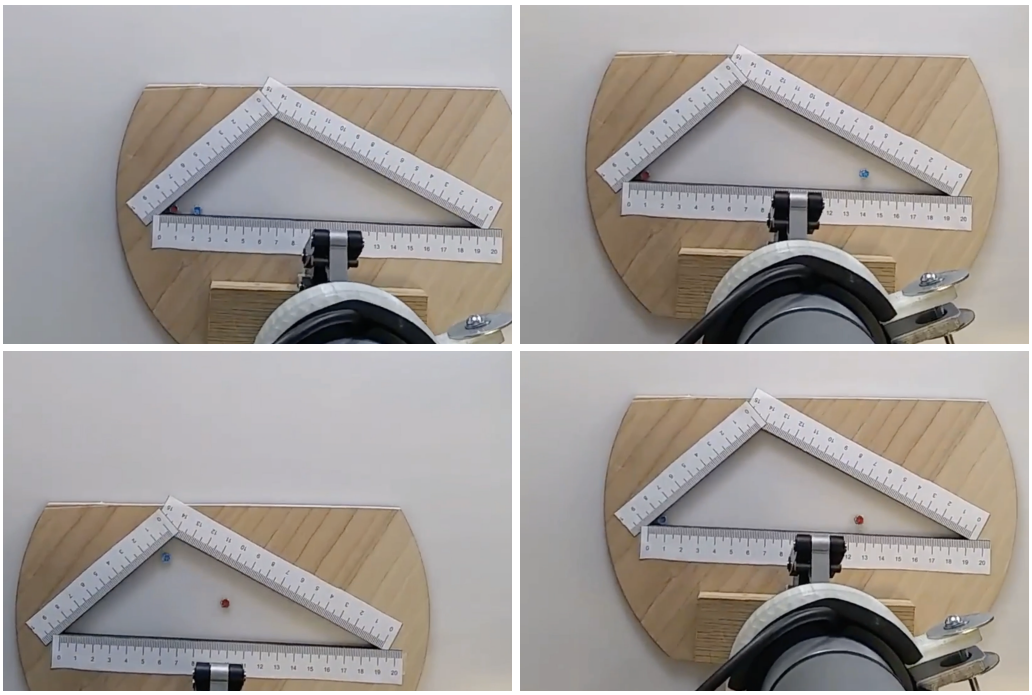


Figure 1 A robot arm moving a triangle to reconfigure two particles. Top left: Two particles are close together. Top right: Blue particles has been separated from the red particle with zig-zag moves. Bottom left: Situation after a south-east and a south-west move. Bottom right: After the blue particle is kept in the bottom left corner, the red particles is moved away with zig-zag moves.

A crucial issue for all these tasks is how to combine the use of a uniform force (which is the same for all involved items) with the individual requirements of object relocation (which may be distinct for different particles): How can we achieve an *arbitrary* arrangement of particles if all of them are subjected to the same external force? Previous work (such as [6]) has shown how arbitrary reconfiguration of an ensemble is possible with the help of specifically placed barriers; however, introducing precisely located obstacles into the workspace is impractical for many scenarios. In this paper, we present a different, less intrusive method: making use of the interplay between static friction with a boundary of the workspace and the external force to achieve any desired configuration. A real-world example is shown in Figure 1.

Shahrokhi et al. [13, 14] already considered reconfiguration problems of particles using friction at the walls. However, they assume walls have infinite friction, i.e., a particle lying at a wall cannot be moved when there is a movement parallel to the wall. This differs from the more realistic assumptions in this paper, in which we only consider finite friction as in [11]

1.1 Our Results.

We provide a fundamentally new approach to manipulating a swarm of objects by an external, global force, demonstrating how static boundary friction can be employed to achieve arbitrary reconfiguration. Our results include the following.

- We show that any two particles in an arrangement can be arbitrarily relocated in a convex workspace, provided sufficient friction as a function of the geometry.
- More specifically, for a triangle with second smallest angle β, we prove that an angle of friction of $\frac{\pi}{2} - \beta$ is always sufficient to guarantee any reconfiguration.

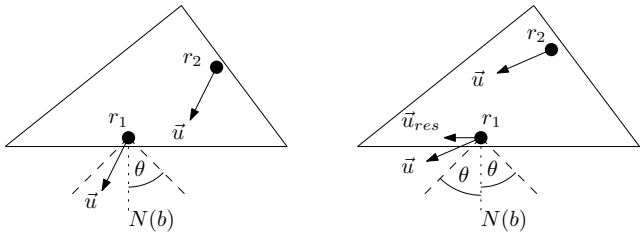


Figure 2 Left: An input force command $u(t)$ within the cone $\pm\theta$ about the normal to the boundary results in no motion of r_1. Right: An input force command $u(t)$ outside the cone results in a motion of both particles. Observe that r_1 slides along the boundary with a resulting force $u_{\text{res}}(t)$.

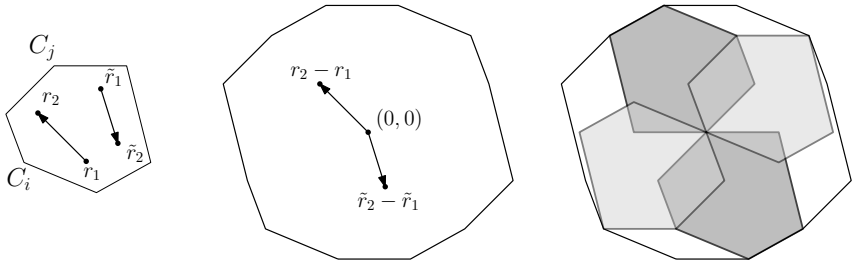


Figure 3 Left: A six-sided polygon P with start positions r_1 and r_2 for two particles and their goal positions $\tilde{r}_1$ and $\tilde{r}_2$. Middle: The Δ configuration of the polygon and the positions of the start and end configuration. Right: Lightgray (darkgray) area corresponds to the C_i-area (C_j-area, resp.).

2 Preliminaries

▶ **Definition 1.** Let θ be the *angle of friction* and $\mu := \tan\theta$ be the *coefficient of friction*. For a particle r lying at a boundary side b, let N(b) be the normal to b. If the angle between force command $\vec{u}$ and N(b) is at most θ, then r does not move at all. If the angle is larger than $\frac{\pi}{2}$ then r moves with full speed. In this paper we do not consider the remaining case.

▶ Problem 1. Given a *workspace*, i.e., a convex polygon with n corners $C_1, \ldots, C_n$, particles r_1 and r_2, and an angle of friction θ, is it possible to reach the configuration $\tilde{r}_1$ and $\tilde{r}_2$?

In this paper, we do not make any assumption on the initial positions of r_1 and r_2, except that they are well separated, i.e., they have a distance $\varepsilon > 0$ to each other.

▶ **Definition 2** (Δ Configuration)**.** The Δ configuration space Δ_P of a convex polygon P with vertices $C_1, \ldots, C_n$ is defined as $\Delta_P := \text{ch}\,(C_i - C_j \mid C_i, C_j \in P)$, where ch$(\cdot)$ denotes the convex hull (for an example see Figure 3). This gives us the set of all relative positions of r_2 to r_1.

From this definition follows that $\Delta_P = \Delta_{-P}$, where $-P$ is P rotated by π. This motivates the following definition.

▶ **Definition 3.** Let C be some vertex of P. The C-area in Δ_P is the union of P and $-P$ having C centered at the origin (see Figure 3 right).

Note that the union of C-areas for all $C \in P$ equals Δ_P.

3 Reconfiguration of two particles

Just like in the context of sorting algorithms in computer science or discrete mathematics, a critical component for achieving arbitrary reconfiguration of larger ensembles is the ability

to rearrange two specific particles. For our purposes of employing external forces and static friction, the additional aspects of geometry and physics have to be considered. These are addressed in this section.

The main idea for this first step is to try to completely cover the Δ configuration. We start by developing a strategy for separating two particles in Subsection 3.1, which gives us a lower bound for θ for every strategy in this section. This is followed by an upper bound for θ in triangles (Subsection 3.1) and arbitrary convex polygons (Subsection 3.2), i.e., we can guarantee any reconfiguration with any angle of friction higher than this bound. By each strategy we develop, more parts of the Δ configuration are covered. Thus, our goal is to give strategies, whose union of covered areas is exactly the Δ configuration.

3.1 Reconfiguration of two particles in arbitrary triangles

As a first step, we provide a sufficient large angle of friction to separate two specific particles.

▶ **Lemma 4.** *Assume particle r_1 is positioned in a corner with angle α, then we can move r_2 to any position in the polygon without moving r_1 by performing zig-zag moves, if $\theta > \frac{\alpha}{2}$ (see Fig. 1 and 4a).*

Proof. Omitted due to space constraints. ◀

Now, let T be a triangle with corners A, B and C, and angles α, β and γ. Furthermore, let α be the smallest angle in T and we assume that $\theta > \frac{\alpha}{2}$ is guaranteed. Consider two particles r_1 and r_2 within T and their goal positions $\tilde{r}_1$ and $\tilde{r}_2$. We have the following strategies to reach the goal positions (see also Fig. 4 for a graphical sketch):

Blue: Move r_1 to A. As shown in Figure 4a, use zig-zag moves to place r_2 in T while r_1 is fixed in A, such that $r_2 - r_1 = \tilde{r}_2 - \tilde{r}_1$. Then, translate r_1 and r_2 to their goal positions.

Red: First, place r_2 in A and move r_1 to B. Then, place r_2 anywhere in the area spanned by $\overline{AB}$ and the angle $\frac{\pi}{2} - \beta + \theta$. Afterwards, translate r_1 and r_2 to their goal positions.

Green: First, Place r_2 in A and move r_1 to C. Then, place r_2 in the area spanned by $\overline{AC}$ and the angle $\frac{\pi}{2} - \gamma + \theta$, such that $r_2 - r_1 = \tilde{r}_2 - \tilde{r}_1$. Afterwards, translate r_1 and r_2 to their goal positions.

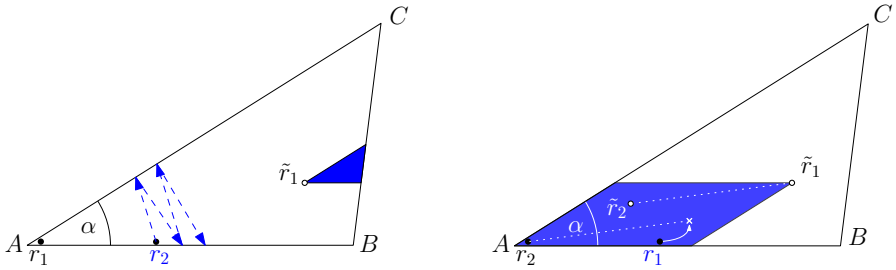
Orange: Place r_2 in C and r_1 in B (as we will see later, this is always possible if $\theta > \frac{\alpha}{2}$). Then, place r_2 in the area spanned by $\overline{BC}$ and the angle $\frac{\pi}{2} - \beta + \theta$, such that $r_2 - r_1 = \tilde{r}_2 - \tilde{r}_1$. Afterwards, translate both particles to their goal position.

Violet: Place r_2 in B and r_1 in C. Then, place r_2 anywhere in the area spanned by $\overline{CB}$ and the angle $\frac{\pi}{2} - \gamma + \theta$, such that $r_2 - r_1 = \tilde{r}_2 - \tilde{r}_1$ Finally, translate both particles to their goal position.

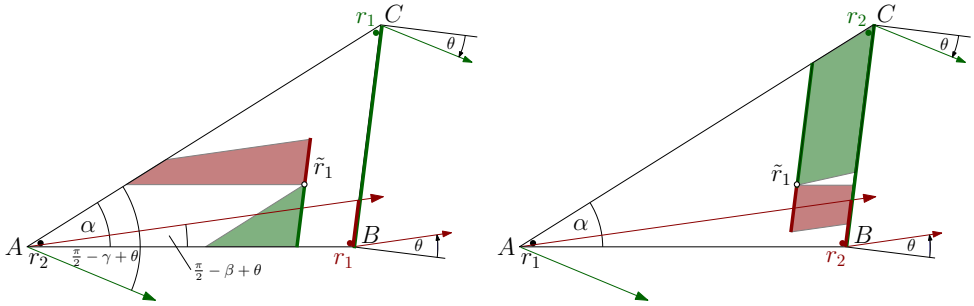
These strategies can also be used by switching the particles r_1 and r_2: Assume that r_1 lies in corner A. To switch r_1 and r_2, we separate both particles to corners B and C, then we use strategy orange or violet (depending on which particle is in which corner), and as a last step, we move r_2 to A.

▶ **Observation 1.** *In the Δ configuration, the covered areas of strategies that overlap are red with orange and green with violet. The blue strategy covers the A-area, red and orange cover parts of the B-area, and green and violet cover parts of the C-area (see Figure 5).*

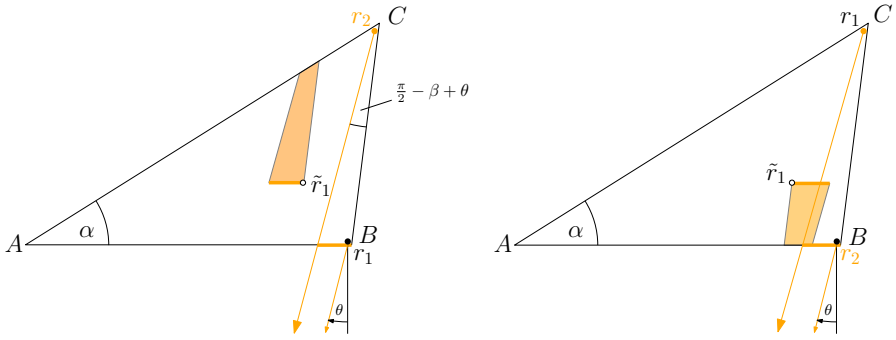
▶ **Lemma 5.** *If $\theta > \frac{\pi}{2} - \gamma$, then the area of the red and orange strategy cover the B-area.*



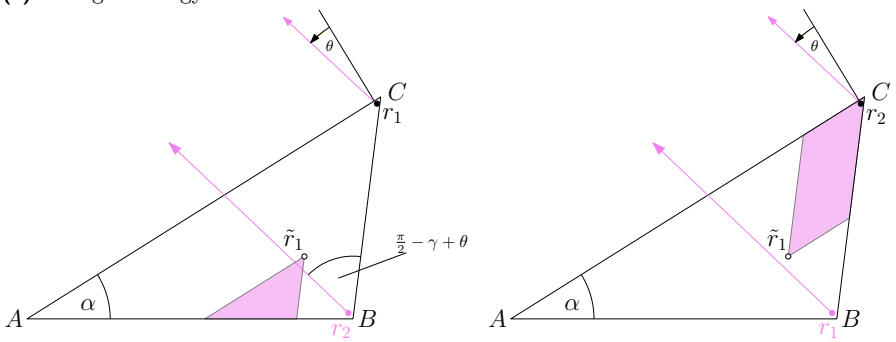
(a) Blue strategy. Dotted lines in the right figure represent the vector $\tilde{r}_2 - \tilde{r}_1$. We move r_1 to the cross with zig-zag moves and then translate both particles to their goal positions.



(b) Red and green strategy

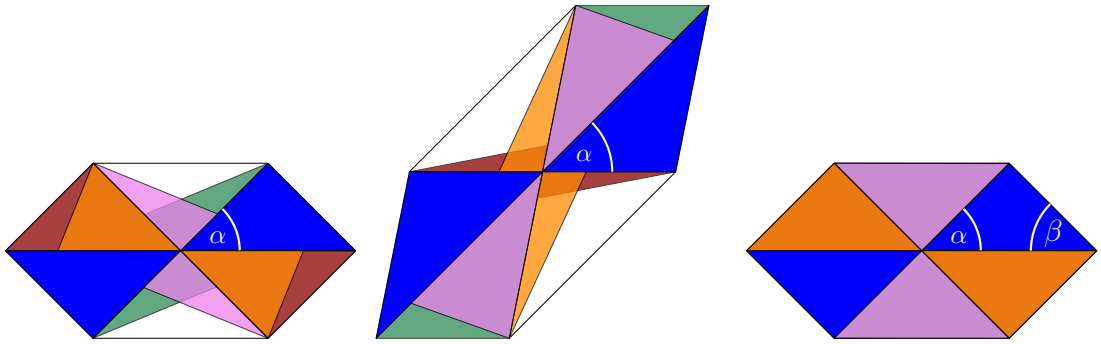


(c) Orange strategy



(d) Violet strategy

Figure 4 Illustration of the five strategies. Colored areas correspond to valid goal positions for r_2, if the goal position of r_1 is $\tilde{r}_1$. Left column: We fix r_1 and move r_2. Right column: We switch the intermediate locations of r_1 and r_2.



(a) (b) (c)

Figure 5 Shown in (a) and (b) are the Δ configurations of top right blue triangle. Colors represent the areas in the Δ configuration covered by our five strategies with an angle of friction of $\frac{\alpha}{2} + \varepsilon$ for some $\varepsilon > 0$. (a),(b): We observe that every strategy may cover areas not covered by any other strategy. (c): If $\theta > \frac{\pi}{2} - \beta$ and $\beta < \gamma$ then we can guarantee full coverage.

Proof. We can prove that the red strategy covers the B-area if $\theta > \frac{\pi}{2} - \gamma$, and that the orange strategy covers the B-area if $\theta > \frac{\pi}{2} - \alpha > \frac{\pi}{2} - \gamma$. Therefore, the lemma holds. Due to space constraints, full details are omitted. ◀

With a similar proof, we can show the following lemma:

▶ **Lemma 6.** *If $\theta > \frac{\pi}{2} - \beta$, then the area of the green and violet strategy cover the C-area.*

▶ **Theorem 7.** *Let T be a triangle with angles $\alpha \leq \beta \leq \gamma$. If $\theta > \frac{\pi}{2} - \beta$, then we can guarantee any reconfiguration of two particles, i.e., Δ_T is completely covered by our strategies.*

Proof. To cover the A-, B-, and C-area of the Δ configuration, the angle of friction θ must be greater than $\max(\frac{\alpha}{2}, \frac{\pi}{2} - \beta, \frac{\pi}{2} - \gamma)$. This is true for $\theta > \frac{\pi}{2} - \beta$. ◀

Because $\frac{\pi}{2} - \gamma = \frac{\pi - 2\gamma}{2} \leq \frac{\pi - \gamma - \beta}{2} = \frac{\alpha}{2}$, the B-area is always covered if $\theta > \frac{\alpha}{2}$. This leads to the following corollary.

▶ **Corollary 8.** *For a triangle T with angles $\alpha \leq \beta \leq \gamma$, at least two thirds of all configurations can be guaranteed if $\theta > \frac{\alpha}{2}$.*

3.2 Reconfiguration of two particles in convex polygons

In this section we generalize the strategy for triangles, i.e., for a particle r_1 in corner C_i and a particle r_2 in corner C_j, moving particle r_2 to cover the C_i-area. As shown in Figure 6, we cannot guarantee full coverage with this strategy, because any movement for r_2 in direction to C_1 would also move r_1. This happens for all pairs of vertices (C_i, C_j) of P, where the segment C_jC_{j+1} has a larger negative slope than the segment C_iC_{i-1}, i.e., if the sum of exterior angles between vertices C_i and C_j is smaller than γ_i. This motivates the following definition.

▶ **Definition 9.** For a vertex $C_i \in P$, let δ_i be the exterior angle at vertex C_i. Let $P^+_{i,j} := \{C_i, C_{i+1}, \ldots, C_{j-1}, C_j\}$ and $P^-_{i,j} := \{C_i, C_{i-1}, \ldots, C_{j+1}, C_j\}$. We define

$$\eta^+_{i,j} := \sum_{C_k \in P^+_{i+1,j-1}} \delta_k \text{ and } \eta^-_{i,j} := \sum_{C_k \in P^-_{i-1,j+1}} \delta_k$$

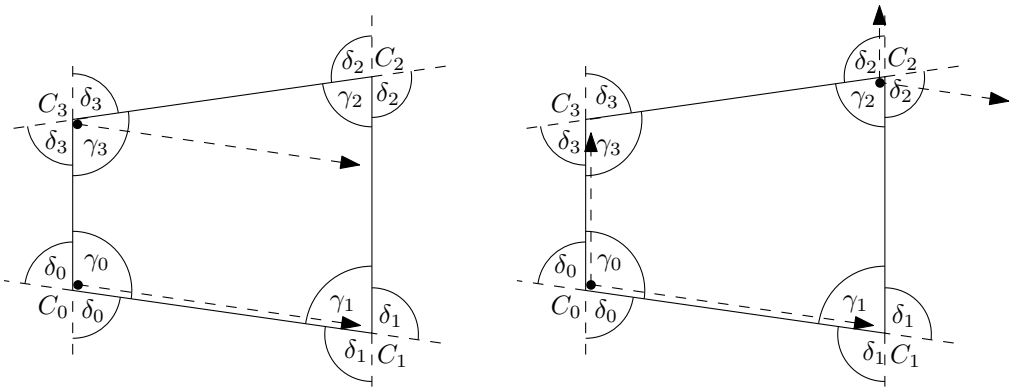


Figure 6 Left: Particle in C_0 cannot be moved without moving a particle in C_3, because $\delta_0 < \gamma_3$. Right: However, we can move the particle in C_0 to any place in the polygon without moving a particle in C_2 (unless the friction is too small), because $\delta_3 + \delta_0 > \gamma_2$ and $\delta_1 + \delta_0 > \gamma_2$.

Furthermore, let $P_i := \{C_j \in P \mid \eta^+_{i,j+1} \geq \gamma_i \wedge \eta^-_{i,j-1} \geq \gamma_i\}$, i.e., P_i contains every vertex of P such that we can use the strategy described in the beginning of this section. Note that all indices are modulo n.

▶ **Lemma 10.** *For a vertex C_i of P, we have $|P_i| \geq 1$.*

Proof. Assume that $|P_i| = 0$. W.l.o.g., let j be the largest index, such that $\eta^+_{i,j+1} < \gamma_i$. If $\eta^-_{i,j} > \gamma_i$, then it immediately follows that $C_{j+1} \in P_i$ and $|P_i| \geq 1$. Otherwise, we have two adjacent vertices C_j and C_{j+1} such that $\eta^+_{i,j+1} < \gamma_i$ and $\eta^-_{i,j} < \gamma_i$. This implies that $2\gamma_i > \eta^+_{i,j+1} + \eta^-_{i,j-1} = -\delta_i + \sum_{C_k \in P} \delta_k = -\delta_i + 2\pi > 2\pi - 2\delta_i = 2\gamma_i$. This is a contradiction and therefore $|P_i| \geq 1$. ◀

▶ **Lemma 11.** *Let P be a convex polygon with vertices $C_0, \ldots, C_{n-1}$ and angles $\gamma_0, \ldots, \gamma_{n-1}$. We can cover the C_i-area if $\theta > \min_{j \in P_i} \left(\frac{\gamma_i}{2}, \max\left(\frac{\gamma_j}{2}, \eta^+_{i,j} - \frac{\pi}{2}, \eta^-_{i,j} - \frac{\pi}{2}\right)\right)$.*

Proof. Omitted due to space constraints. ◀

Combining Lemmas 10 and 11 yields the following theorem.

▶ **Theorem 12.** *Let P be a convex Polygon with vertices $C_0, \ldots, C_{n-1}$ and angles $\gamma_0, \ldots, \gamma_{n-1}$. If $\theta > \max_{0 \leq i < n} \left(\min_{j \in P_i} \left(\frac{\gamma_i}{2}, \max\left(\frac{\gamma_j}{2}, \eta^+_{i,j} - \frac{\pi}{2}, \eta^-_{i,j} - \frac{\pi}{2}\right)\right)\right)$, then every configuration of two particles can be reached.*

▶ **Corollary 13.** *If P is a regular polygon with n vertices and if $\mu > \cot(\pi/n)$, then every reconfiguration is possible.*

4 Conclusion

We introduced a novel approach for rearranging the positions of particles by applying global uniform forces, making use of different local static friction to achieve arbitrary goal positions. To this end, we provided strategies enabling arbitrary rearrangements of two particles in convex workspaces, giving a characterization of the critical coefficient of friction in terms of the boundary geometry. Future work can now investigate optimal motion planning.

References

1. G. Aloupis, J. Cardinal, S. Collette, F. Hurtado, S. Langerman, and J. O'Rourke. Draining a polygon—or—rolling a ball out of a polygon. *Computational Geometry*, 47(2):316–328, 2014.
2. D. Arbuckle and A. A. Requicha. Self-assembly and self-repair of arbitrary shapes by a swarm of reactive robots: algorithms and simulations. *Autonomous Robots*, 28(2):197–211, 2010.
3. V. M. Baez, A. T. Becker, S. P. Fekete, and A. Schmidt. Coordinated particle relocation with global control and local friction, 2020. `https://www.ibr.cs.tu-bs.de/users/fekete/Videos/Friction_SoCG20.mp4`.
4. J. Balanza-Martinez, A. Luchsinger, D. Caballero, R. Reyes, A. A. Cantu, R. Schweller, L. A. Garcia, and T. Wylie. Full tilt: Universal constructors for general shapes with uniform external forces. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2689–2708, 2019.
5. A. Becker, E. D. Demaine, S. P. Fekete, and J. McLurkin. Particle computation: Designing worlds to control robot swarms with only global signals. In *IEEE ICRA*, pages 6751–6756, 2014.
6. A. T. Becker, E. D. Demaine, S. P. Fekete, J. Lonsford, and R. Morris-Wright. Particle computation: complexity, algorithms, and logic. *Natural Computing*, 18(1):181–201, 2019.
7. A. T. Becker, S. P. Fekete, P. Keldenich, D. Krupke, C. Rieck, C. Scheffer, and A. Schmidt. Tilt assembly: algorithms for micro-factories that build objects with uniform external forces. *Algorithmica*, pages 1–23, 2017.
8. P. S. S. Kim, A. T. Becker, Y. Ou, A. A. Julius, and M. J. Kim. Imparting magnetic dipole heterogeneity to internalized iron oxide nanoparticles for microorganism swarm control. *Journal of Nanoparticle Research*, 17(3):1–15, 2015.
9. J. S. Lewis and J. M. O'Kane. Planning for provably reliable navigation using an unreliable, nearly sensorless robot. *The International Journal of Robotics Research*, 32(11):1342–1357, 2013.
10. M. Rubenstein, C. Ahler, N. Hoff, A. Cabrera, and R. Nagpal. Kilobot: A low cost robot with scalable operations designed for collective behaviors. *Robotics and Autonomous Systems*, 62(7):966–975, 2014.
11. A. Schmidt, V. M. Baez, A. T. Becker, and S. P. Fekete. Coordinated particle relocation using finite static friction with boundary walls. *IEEE Robotics and Automation Letters*, 5(2):985–992, April 2020.
12. A. Schmidt, S. Manzoor, L. Huang, A. T. Becker, and S. P. Fekete. Efficient parallel self-assembly under uniform control inputs. *IEEE Robotics and Automation Letters*, 3(4):3521–3528, 2018.
13. S. Shahrokhi, A. Mahadev, and A. T. Becker. Algorithms for shaping a particle swarm with a shared input by exploiting non-slip wall contacts. In *IEEE/RSJ IROS*, pages 4304–4311, 2017.
14. S. Shahrokhi, J. Shi, B. Isichei, and A. T. Becker. Exploiting nonslip wall contacts to position two particles using the same control input. *IEEE Transactions on Robotics*, pages 1 – 12, 2019.
15. Y. Zhang, X. Chen, H. Qi, and D. Balkcom. Rearranging agents in a small space using global controls. In *IEEE/RSJ Int Conf on Intelligent Robots and Systems*, pages 3576–3582, 2017.
16. Y. Zhang, E. Whiting, and D. Balkcom. Assembling and disassembling planar structures with divisible and atomic components. *IEEE Transactions on Automation Science and Engineering*, 15(3):945–954, 2018.

Probing a Set of Trajectories to Maximize Captured Movement

Sándor P. Fekete[1], Alexander Hill[1], Dominik Krupke[1], Tyler Mayer[2], Joseph S. B. Mitchell[2], Ojas Parekh[3], and Cynthia A. Phillips[3]

1 **Department of Computer Science, TU Braunschweig, Germany**
{s.fekete,a.hill,d.krupke}@tu-bs.de
2 **Department of Applied Mathematics and Statistics, Stony Brook University, USA.**
tmayer93@gmail.com, jsbm@ams.sunysb.edu
3 **Sandia National Labs, USA**
{odparek,caphill}@sandia.gov

Abstract

We study the *Trajectory Capture Problem* (TCP), in which, for a given input set $\mathcal{T}$ of trajectories in the plane, and an integer $k \geq 2$, we seek to compute a set of k points ("portals") to maximize the total length of all subtrajectories of $\mathcal{T}$ between pairs of portals. This problem naturally arises in trajectory analysis and summarization.

We show that TCP is polynomial time solvable in 1D and NP-hard in 2D and then focus on tackling the TCP with integer linear programming to solve instances to provable optimality. We analyze this method on different classes of data, including benchmark instances that we generate. We demonstrate that we are able to compute provably optimal solutions for real-world instances.

1 Introduction

We study the Trajectory Capturing Problem (TCP): Given a set $\mathcal{T}$ of trajectories and an integer $k \geq 2$, determine a set of k *portals* (points) to maximize the sum of the lengths of the inter-portal subtrajectories in $\mathcal{T}$; such subtrajectories are said to be "captured" by the set of portals. This problem arises in placing a limited number of toll booths, cameras for average speed measurement, various other types of sensors, or abstract collections of focus points for sampling trajectories. For example, consider the scenario shown in Figure 1, which corresponds to more than 500,000 data points that arise from the trajectories of over 250 taxi cabs in San Francisco. Our goal is to identify a small subset of locations that allow us to capture as much of the movement pattern as possible.

Our main results are as follows. We mention proofs that the TCP is NP-hard and allows constant-factor approximation, based on geometric parameters. Our main focus is on demonstrating that even relatively large instances of the TCP can be solved to provable optimality with the help of Integer Programming.

Related Work. Related to the TCP is the well-studied geometric hitting set problem, in which one seeks a smallest cardinality set of points to hit a given set of lines, segments, or trajectories. These NP-hard problems are hard to approximate (below a threshold), and special cases have improved (constant-factor) approximation algorithms; see, e.g., the references in [5]. The difference to the TCP is the need to place at least two hit points on a trajectory in order to get (partial and weighted) credit for hitting it.

Buchin et al. [4] provide a set of positive and negative results for the similarly motivated problem of covering the most 'sites' in a weighted graph by a tree or a path of limited weight.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Figure 1 A set of taxi trajectories in the San Francisco Bay Area.

Limiting the weight and maximizing the covered vertices instead of limiting the spanning vertices and maximizing the covered weight in between, however, makes it fundamentally different to ours.

There is a vast literature on problems of analyzing, clustering, and summarizing a set of trajectories. In particular, notions of "flocks" and "meetings" have been formalized and studied algorithmically [2, 6, 9, 17]. Gudmundsson, van Kreveld, and Speckmann [7] define *leadership, convergence, and encounter* and provide exact and approximate algorithms to compute each. Andersson et al. [1] show that the *Leader-Problem* (LP) variants (*LP-Report-All, LP-max-Length, LP-Max-Size*) are all polynomially solvable and provide exact algorithms. Buchin et al. [3] present a framework to fully categorize trajectory grouping structures (grouping, merging, splitting, and ending of groups). They model grouping patterns in trajectory data using the Reeb graph in 3D and show several combinatorial bounds. Other approaches to trajectory summarization naturally include cluster analysis, of which there is a large body of related work. Li, Han, and Yang [12] consider rectilinear trajectories and show how to cluster with bounding rectangles of a given size. Several approaches (e.g., [14, 11, 10, 8]) consider density based methods for clustering sub-trajectories; Lee, Han, and Li [10] take it one step further by considering a two-level clustering hierarchy that first accounts for regional density and then considers lower-level movement patterns. Li, Ding, Han, and Kays [13] consider a problem (related to [7]) in which they seek to identify all *swarms* or groups of entities moving within an arbitrary shaped cluster for a certain, possibly disconnected, duration of time. Also, Uddin, Ravishankar, and Tsotras [16] consider finding what they call *regions of interest* in a trajectory database.

2 Preliminaries

We are given a set of trajectories $\mathcal{T}$, each specified by a sequence of points, e.g., in the Euclidean plane. We seek a set $P = \{p_1, \ldots, p_k\}$ of k *portals*, i.e., selected points that lie on some of the trajectories. While our practical study focuses on instances in which the trajectories $\mathcal{T}$ are purely spatial, e.g., given as polygonal chains or line segments in the plane, our methods apply equally well to more general portals and to trajectories that include the temporal component and live in space-time. More generally, we are given a graph $\mathcal{G}$, with length-weighted edges, and a set of (simple) paths within $\mathcal{G}$. We wish to determine a subset of k of the nodes of $\mathcal{G}$ that maximizes the sum of the (weighted) lengths of the subpaths (of the input paths) that link consecutive portals along the input paths.

We seek to compute a P that maximizes the total *captured weight* of subtrajectories between pairs of portals. For a trajectory $\tau \in \mathcal{T}$, if there are two or more portals of P that lie along τ, say $\{p_{i_1}, \ldots, p_{i_q}\}$ (for $q \geq 2$), then the subtrajectory, $\tau_{p_{i_1}, p_{i_q}}$, between p_{i_1} and p_{i_q} is *captured* by P, and we get credit for its weight $f(\tau_{p_{i_1}, p_{i_q}})$. (For many of our instances, $f(\tau_{p_{i_1}, p_{i_q}})$ corresponds to the Euclidean distances, denoted by $|\tau_{p_{i_1}, p_{i_q}}|$, but our methods generalize to other types of weights.) Let $f_P(\tau)$ denote the captured weight of trajectory τ by the portal set P. The Trajectory Capture Problem (TCP) is then to compute, for given $\mathcal{T}$ and k, a set of k portals $P = \{p_1, \ldots, p_k\}$ to maximize $\sum_{\tau \in \mathcal{T}} f_P(\tau)$.

3 One-Dimensional TCP

In the one-dimensional setting, the underlying graph $\mathcal{G}$ is a path, and the input trajectories $\mathcal{T} = \{(a_1, b_1), \ldots, (a_n, b_n)\}$ are a set of subpaths of $\mathcal{G}$, specified by pairs of integers, a_i, b_i. A solution to the TCP then consists of k points, $P = \{p_1, \ldots, p_k\}$, w.l.o.g. indexed in sorted order, $p_1 < p_2 < \cdots < p_k$.

▶ **Theorem 1.** *The one-dimensional TCP can be solved exactly in polynomial time.*

Proof. For $i = 1, 2, \ldots, k-1$, let $V_i(x)$ be the maximum possible length of $\mathcal{T}$ captured by points $(p_i, \ldots, p_k)$, with $p_i = x \in \{a_1, \ldots, a_n, b_1, \ldots, b_n\}$; let $V_k(x) = 0$, for any x. Then, the value functions V_i satisfy the following dynamic programming recursion, for $i = 1, 2, \ldots, k-1$, and each $x \in \{a_1, \ldots, a_n, b_1, \ldots, b_n\}$:

$$V_i(x) = \max_{x' \in \{a_1, \ldots, a_n, b_1, \ldots, b_n\}, x' > x} \{V_{i+1}(x') + \sum_{j:(x,x') \subseteq (a_j, b_j)} (x' - x)\}.$$

The summation counts the length $(x' - x)$ once for each input interval that contains the interval (x, x'). We can compute the $O(nk)$ values $V_i(x)$ in time $O(n^2 k)$ by incrementally updating the summation as we consider values of x' in increasing order. ◀

4 Two-Dimensional TCP

The TCP is NP-hard (and hard to approximate) in the general graph setting (in which there is a general graph $\mathcal{G}$ given, and a set of paths within $\mathcal{G}$ is given as the input trajectories), even in the case of unweighted (weight-1) edges: We can simply give as input paths the set of single-edge paths in $\mathcal{G}$; thus, the TCP that asks if we can achieve total capture weight of $\binom{k}{2}$ is asking if there is a clique of size k in $\mathcal{G}$.

For TCP with input trajectories $\mathcal{T}$ given by (straight) line segments, we can show that the problem is also NP-hard by a reduction from Hitting Lines, see Fig. 2.

▶ **Theorem 2.** *The TCP for an input set $\mathcal{T}$ of n line segments, arbitrarily overlapping in the plane, is NP-hard.*

By a more intricate construction, we can show hardness even for axis-aligned segments.

On the other hand, we can provide the following approximation results, based on specific geometric parameters of the instances. The proofs are omitted from this short abstract.

▶ **Theorem 3.** *The TCP for an input set $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2 \cup \cdots \cup \mathcal{T}_K$, with each subset $\mathcal{T}_i$ having no crossing points, has a polynomial-time K-approximation algorithm.*

▶ **Theorem 4.** *The TCP for an input set $\mathcal{T}$ of arbitrarily overlapping/crossing trajectory paths in the plane having bounded depth D (i.e., no point of $\Re^2$ lies in more than D input trajectories) has a polynomial-time D-approximation algorithm.*

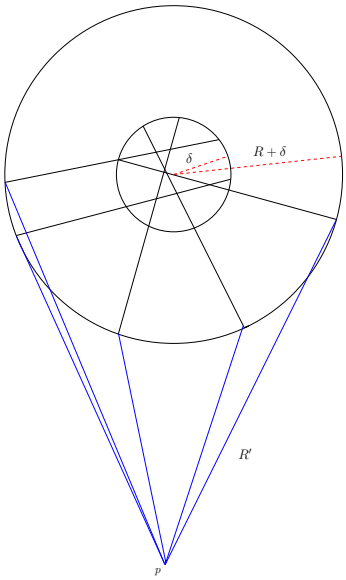


Figure 2 Reduction from Hitting Lines in which we have to decide whether we can hit all n (black) lines with l pins. There exists a circle that contains all intersections. For the reduction we create a second significantly larger circle and extend the lines to the lower half. We add n blue segments at the ends of the extended hitting lines to a point far below. The n blue lines are very long and captured by $n+1$ portals. We can additionally capture the part between the two circles with $l = k - (n+1)$ portals if there exists a hitting set with l pins.

5 Practical Results

5.1 Integer Linear Programming

We assume an input graph $\mathcal{G} = (V, E)$, and a collection of trajectories $\mathcal{T}$. Each trajectory $\tau \in \mathcal{T}$ is represented as a path in $\mathcal{G}$, and we refer to the two interchangeably. An edge e in a trajectory τ represents a portion of τ that is captured by any pair of nodes (portals) $p, q \in V$ that induces a subpath of τ containing e. Capturing an edge e earns an associated weight $f(e)$, and we seek to maximize the total weight earned by capturing edges subject to selecting at most k nodes. An IP can now be stated as follows.

$$\max \textstyle\sum_{\tau \in \mathcal{T}, e \in E(\tau)} f(e) x_{\tau,e}$$

$$\textstyle\sum_{v \in V} y_v \leq k \qquad \text{(Constraint 1)}$$

$$\forall \tau = (v_0, \ldots, v_l) \in \mathcal{T}:$$

$$\forall i \in 0, \ldots, l-1: \quad \begin{cases} x_{\tau,v_i v_{i+1}} \leq y_i & \text{if } i = 0, \\ x_{\tau,v_i v_{i+1}} \leq y_i + x_{\tau,v_{i-1} v_i} & \text{else} \end{cases} \qquad \text{(Constraint 2)}$$

$$\forall i \in 1, \ldots, l: \quad \begin{cases} x_{\tau,v_{i-1} v_i} \leq y_i & \text{if } i = l, \\ x_{\tau,v_{i-1} v_i} \leq y_i + x_{\tau,v_i v_{i+1}} & \text{else} \end{cases} \qquad \text{(Constraint 3)}$$

$$\forall v \in V, \tau \in e \in \tau: \quad x_{\tau,e}, y_v \in \{0,1\}$$

We have two types of Boolean variables: y_v, for $v \in V$, which indicates if node v is one of the k selected portals, and $x_{\tau,e}$, for edge $e \in E$ on trajectory τ, which indicates if the portion e of trajectory τ is captured by selected portals. For an edge e, there are distinct variables, $x_{\tau,e}, x_{\tau',e}$, for trajectories $\tau \neq \tau'$ because e may be captured in τ but not in τ'.

Our objective function maximizes the weighted sum of captured trajectory edges, where

$E(\tau)$ denotes the edges of τ in $\mathcal{G}$, and $f(e)$ is the weight (i.e. length) of edge e. (Optionally, we could have trajectory-dependent weights on edges.) Constraint 1 limits the number ($\leq k$) of selected portals. Constraints 2 and 3 enforce that, in order for an edge to be captured as part of trajectory τ, there must be a selected portal in each direction; either there is a selected portal at the next node, or the following trajectory edge is also captured. In the latter case, because τ has no cycle (it is a simple path), there must be a selected portal on τ at some point in that direction if any portion of τ is to be captured.

Figure 3 shows an example; using "x_i" as shorthand for the IP variables $x_{\tau,v_iv_{i+1}}$ that correspond to the edges along trajectory path $\tau = (v_0, v_1, \ldots, v_6)$. Constraints 2 are: $x_0 \leq y_0$, $x_1 \leq y_1 + x_0$, $x_2 \leq y_2 + x_1, \ldots, x_5 \leq y_5 + x_4$. Constraints 3 are: $x_5 \leq y_6$, $x_4 \leq y_5 + x_5$, $x_3 \leq y_4 + x_4, \ldots, x_0 \leq y_1 + x_1$. With portals selected at v_1 and v_4 (i.e., with $y_1 = y_4 = 1$, $y_i = 0$ otherwise), we get constraints $x_0 \leq 0$, $x_5 \leq 0$, and $x_4 \leq y_5 + x_5 = 0$, implying that only the subpath (v_1, v_2, v_3, v_4) of τ contributes to the objective function.

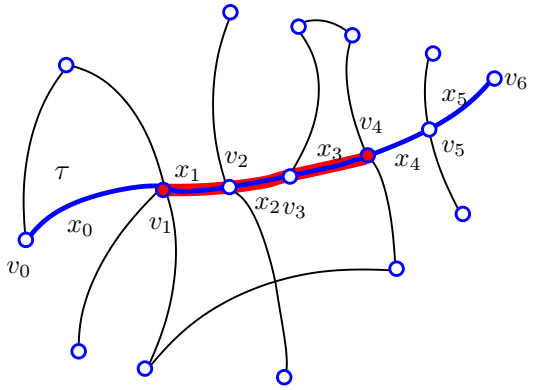


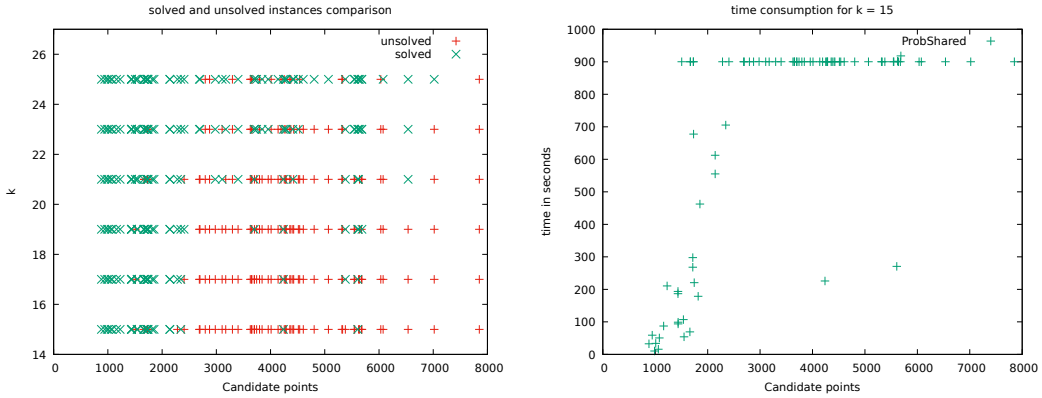
Figure 3 Formulating the IP: Trajectory $\tau = (v_0, v_1, \ldots, v_6)$ is highlighted in blue. With selected portals at v_1 and v_4, the portion highlighted in red is captured.

5.2 Experimental Evaluation

We used three kinds of instances for experimental evaluation: **(1)** Instances based on 10-20% of the segments/edges in a straight line embedding of a complete graph with 35-55 random points (likely to have high degree while most intersections only have degree 4) where the trajectories are the full edges, **(2)** random non-overlapping axis-parallel segments, and **(3)** trajectories based on real-world taxi tracking data. For all instances, we used CPLEX (V12.7.1 with default settings) with a time limit of 900 s on Intel(R) Core(TM) i7-4770 with 32 GB. The code and data is available at `https://github.com/ahillbs/trajectory_capturing`.

For type **(1)** we see in Figure 4 that instances with up to ~ 2500 candidate points (i.e., nodes at intersection points in the graph, where selected portals can be placed) can be solved for $15 \leq k \leq 23$ to provable optimality within the time limit. Instances with more than 2500 candidate points are most often not solved for $k < 23$. For instances with $k \geq 23$, the problem seems to be easier to solve. In Figure 4b we see that for $k = 15$ and between 1500 and 2000 candidate points, instances start to become very difficult to solve. On the other hand, for $k \geq 23$ instances are still solvable for more than 2500 candidate points.

For **(2)** axis-parallel non-overlapping segments (see Fig. 5), we do not know the hardness but one can devise a simple 2-approximation via dynamic programming. Indeed, these



(a) Instances solved within 900 s.

(b) Time used for instances with $k = 15$.

Figure 4 Instances with a percentage of segments between 35-55 random points. The problem is harder for placing only 15 portals than placing 25 portals.

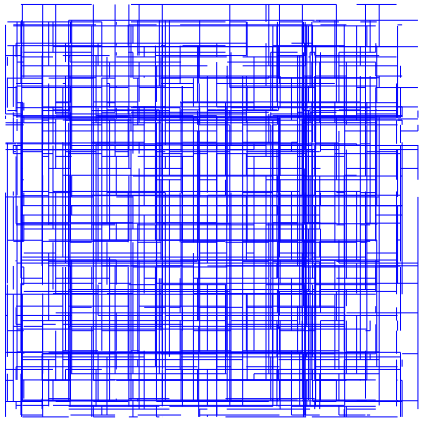


Figure 5 An instance with 1100 axis-parallel segments and 8200-8500 candidate points.

instances are much easier to solve, see Fig. 7. We have solved instances with 2000 segments and 19,000 candidate points. Furthermore, for instances with up to 20,000 candidate points, the integrality gap was never larger than 20% for $k = 5$, 7% for $k = 10$, 5% for $k = 15$ and less than 2.5% for larger k.

For **(3)** we studied taxi cab trajectories in the San Francisco Bay Area. Figure 6 shows our results for $k = 5$ optimal portals. (The data is based on 375 vehicles, sampled every 5 minutes, 288 times per day, for one week [15]; the satellite imagery is courtesy of Planet Labs.) Our experiments included runs on 30 instances, with k ranging from 5 to 11, on sets of 10 to 120 trajectories of varying lengths (comprised of 1300 to 3700 edges, and 600 to 1800 vertices). The measurements of the trajectories are snapped to a regular grid graph to counteract GPS inaccuracy. Solution times were up to 200 seconds of computation, with most instances taking less than 10 seconds.

6 Conclusion

The main theoretical questions that remain open are: (1) Is TCP NP-hard for axis-aligned segments that are allowed to intersect at single points (endpoints or crossing points) but

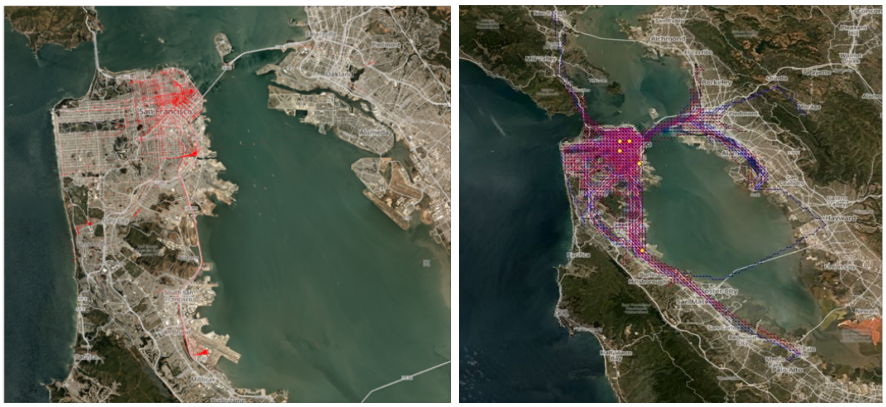


Figure 6 (Left) Data points before processing. (Right) Solution to real-world TCP instance: An optimal set of $k = 5$ portals are highlighted by yellow dots. The captured parts of the trajectories are highlighted in red while not captured parts are shown in blue.

not to overlap in subsegments? (our construction relied on overlapping segments); and, (2) Can we improve the approximation factor (of K) for a set of trajectories that is the union of K subsets, each of which is noncrossing? On the practical side, there is a wide range of real-world classes of instances that we can explore.

References

1 Mattias Andersson, Joachim Gudmundsson, Patrick Laube, and Thomas Wolle. Reporting leaders and followers among trajectories of moving point objects. *GeoInformatica*, 12(4):497–528, 2008.

2 Marc Benkert, Joachim Gudmundsson, Florian Hübner, and Thomas Wolle. Reporting flock patterns. *Computational Geometry*, 41(3):111–125, 2008.

3 Kevin Buchin, Maike Buchin, Marc van Kreveld, Bettina Speckmann, and Frank Staals. Trajectory grouping structure. In *Algorithms and data structures*, pages 219–230. Springer, 2013.

4 Kevin Buchin, Sergio Cabello, Joachim Gudmundsson, Maarten Löffler, Jun Luo, Günter Rote, Rodrigo I Silveira, Bettina Speckmann, and Thomas Wolle. Finding the most relevant fragments in networks. *J. Graph Algorithms Appl.*, 14(2):307–336, 2010.

5 Sándor P Fekete, Kan Huang, Joseph SB Mitchell, Ojas Parekh, and Cynthia A Phillips. Geometric hitting set for segments of few orientations. *Theory of Computing Systems*, 62(2):268–303, 2018.

6 Joachim Gudmundsson and Marc van Kreveld. Computing longest duration flocks in trajectory data. In *Proc. of the 14th Annual ACM International Symposium on Advances in Geographic Information Systems*, pages 35–42. ACM, 2006.

7 Joachim Gudmundsson, Marc van Kreveld, and Bettina Speckmann. Efficient detection of patterns in 2d trajectories of moving points. *Geoinformatica*, 11(2):195–215, 2007.

8 Marios Hadjieleftheriou, George Kollios, Dimitrios Gunopulos, and Vassilis J Tsotras. Online discovery of dense areas in spatio-temporal databases. In *Advances in Spatial and Temporal Databases*, pages 306–324. Springer, 2003.

9 Patrick Laube, Matt Duckham, and Thomas Wolle. Decentralized movement pattern detection amongst mobile geosensor nodes. In *Geographic Information Science*, pages 199–216. Springer, 2008.

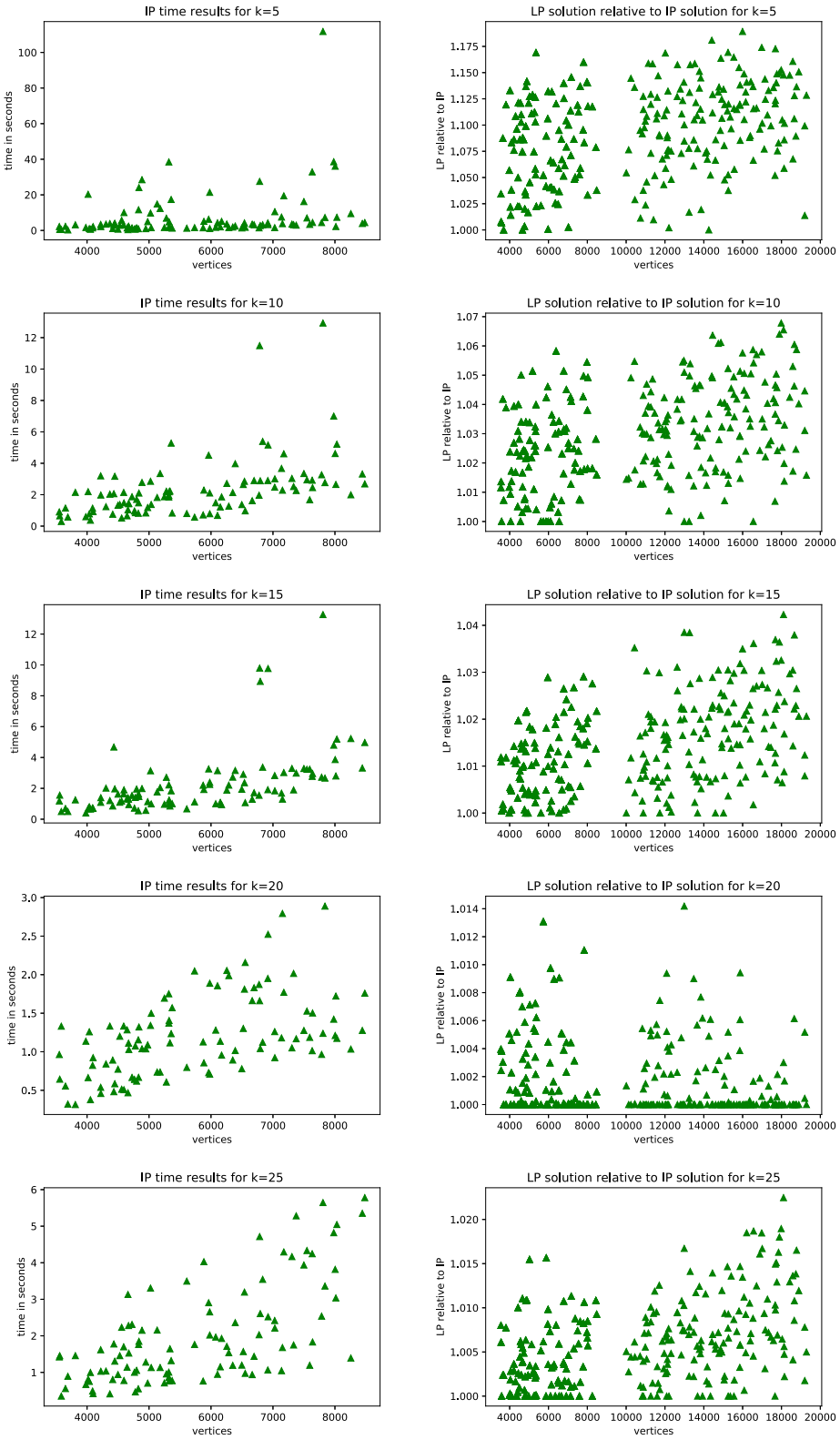


Figure 7 IP runtime and integrality gap for axis-parallel instances and $k = 5, 10, \ldots, 25$.

10 Jae-Gil Lee, Jiawei Han, Xiaolei Li, and Hector Gonzalez. Traclass: trajectory classification using hierarchical region-based and trajectory-based clustering. *Proceedings of the VLDB Endowment*, 1(1):1081–1094, 2008.

11 Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. Trajectory clustering: a partition-and-group framework. In *Proc. of the 2007 ACM SIGMOD International Conference on Management of Data*, pages 593–604. ACM, 2007.

12 Yifan Li, Jiawei Han, and Jiong Yang. Clustering moving objects. In *Proc. Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 617–622. ACM, 2004.

13 Zhenhui Li, Bolin Ding, Jiawei Han, and Roland Kays. Swarm: Mining relaxed temporal moving object clusters. *Proceedings of the VLDB Endowment*, 3(1-2):723–734, 2010.

14 Mirco Nanni and Dino Pedreschi. Time-focused clustering of trajectories of moving objects. *Journal of Intelligent Information Systems*, 27(3):267–289, 2006.

15 Michal Piorkowski, Natasa Sarafijanovic-Djukic, and Matthias Grossglauser. CRAWDAD dataset epfl/mobility (v. 2009-02-24), doi:10.15783/c7j010, February 2009.

16 Md Reaz Uddin, Chinya Ravishankar, and Vassilis J Tsotras. Finding regions of interest from trajectory data. In *Mobile Data Management (MDM), 2011 12th IEEE International Conference on*, volume 1, pages 39–48. IEEE, 2011.

17 Marcos R Vieira, Petko Bakalov, and Vassilis J Tsotras. On-line discovery of flock patterns in spatio-temporal data. In *Proc. of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 286–295. ACM, 2009.

On the Average Complexity of the k-Level*

Man-Kwun Chiu[1], Stefan Felsner[2], Manfred Scheucher[2], Patrick Schnider[3], Raphael Steiner[2], and Pavel Valtr[4]

1 **Department of Mathematics and Computer Science, Freie Universität Berlin, Germany,**
{chiumk}@zedat.fu-berlin.de
2 **Institut für Mathematik, Technische Universität Berlin, Germany,**
{felsner,scheucher,steiner}@math.tu-berlin.de
3 **Department of Computer Science, ETH Zürich, Switzerland**
{patrick.schnider}@inf.ethz.ch
4 **Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Czech Republic**

Abstract

Let $\mathcal{L}$ be an arrangement of n lines in the Euclidean plane. The *k-level* of $\mathcal{L}$ consists of all vertices v of the arrangement which have exactly k lines of $\mathcal{L}$ passing below v. The complexity (the maximum size) of the k-level in a line arrangement has been widely studied. In 1998 Dey proved an upper bound of $O(n\cdot(k+1)^{1/3})$. Due to the correspondence between lines in the plane and great-circles on the sphere, the asymptotic bounds carry over to arrangements of great-circles on the sphere, where the k-level denotes the vertices at distance at most k to the south pole.

We prove an upper bound of $O((k+1)^2)$ on the expected complexity of the k-level in great-circle arrangements if the south pole is chosen uniformly at random among all cells.

We also consider arrangements of great $(d-1)$-spheres on the sphere $\mathbb{S}^d$ which are orthogonal to a set of random points on $\mathbb{S}^d$. In this model, we prove that the expected complexity of the k-level is of order $\Theta((k+1)^{d-1})$.

1 Introduction

Let $\mathcal{L}$ be an arrangement of n lines in the Euclidean plane. The *vertices* of $\mathcal{L}$ are the intersection points of lines of $\mathcal{L}$. Throughout this article we consider arrangements with the properties that no line is vertical and no three lines intersect in a common vertex. The *k-level* of $\mathcal{L}$ consists of all vertices v which have exactly k lines of $\mathcal{L}$ below v. We denote the k-level by $V_k(\mathcal{L})$ and its size by $f_k(\mathcal{L})$. Moreover, by $f_k(n)$ we denote the maximum of $f_k(\mathcal{L})$ over all arrangements $\mathcal{L}$ of n lines, and by $f(n)=f_{\lfloor(n-2)/2\rfloor}(n)$ the maximum size of the *middle level*.

A *k-set* of a finite point set P in the Euclidean plane is a subset K of k elements of P that can be separated from $P\setminus K$ by a line. Paraboloid duality is a bijection $P\leftrightarrow\mathcal{L}_P$ between point sets and line arrangements (for details on this duality see [17, Chapter 6.5] or [8, Chapter 1.4]). The number of k-sets of P equals $|V_{k-1}(\mathcal{L}_P)\cup V_{n-1-k}(\mathcal{L}_P)|$.

* M.-K. Chiu was supported by ERC StG 757609. S. Felsner and M. Scheucher were supported by DFG Grant FE 340/12-1. R. Steiner was supported by DFG-GRK 2434. P. Schnider was supported by the SNSF Project 200021E-171681. P. Valtr was supported by the grant no. 18-19158S of the Czech Science Foundation (GAČR) and by the PRIMUS/17/SCI/3 project of Charles University. This work was initiated at a workshop of the collaborative DACH project *Arrangements and Drawings* in Schloss St. Martin, Graz. We thank the organizers for the inspiring atmosphere. We also thank Birgit Vogtenhuber for interesting discussions.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

In discrete and computational geometry bounds on the number of k-sets of a planar point set, or equivalently on the size of k-levels of a planar line arrangement have important applications. The complexity of k-levels was first studied by Lovász [14] and Erdős et al. [11]. They bound the size of the k-level by $O(n \cdot (k+1)^{1/2})$. Dey [6] used the crossing lemma to improve the bound to $O(n \cdot (k+1)^{1/3})$. In particular, the maximum size $f(n)$ of the middle level is $O(n^{4/3})$. Concerning the lower bound on the complexity, Erdős et al. [11] gave a construction showing that $f(2n) \geq 2f(n) + cn = \Omega(n \log n)$ and conjectured that $f(n) \geq \Omega(n^{1+\varepsilon})$. An alternative $\Omega(n \log n)$-construction was given by Edelsbrunner and Welzl [10]. The current best lower bound $f_k(n) \geq n \cdot e^{\Omega(\sqrt{\log k})}$ was obtained by Nivasch [16] improving on a bound by Tóth [22].

1.1 Generalized Zone Theorem

In order to define "zones", let us introduce the notion of "distances". For x and x' being a vertex, edge, line, or cell of an arrangement $\mathcal{L}$ of lines in $\mathbb{R}^2$ we let their *distance* $\mathrm{dist}_{\mathcal{L}}(x, x')$ be the minimum number of lines of $\mathcal{L}$ intersected by the interior of a curve connecting a point of x with a point of x'. Pause to note that the k-level of $\mathcal{L}$ is precisely the set of vertices which are at distance k to the bottom cell.

The $(\leq j)$-zone $Z_{\leq j}(\ell, \mathcal{L})$ of a line ℓ in an arrangement $\mathcal{L}$ is defined as the set of vertices, edges, and cells from $\mathcal{L}$ which have distance at most j from ℓ. See Figure 1a for an illustration.

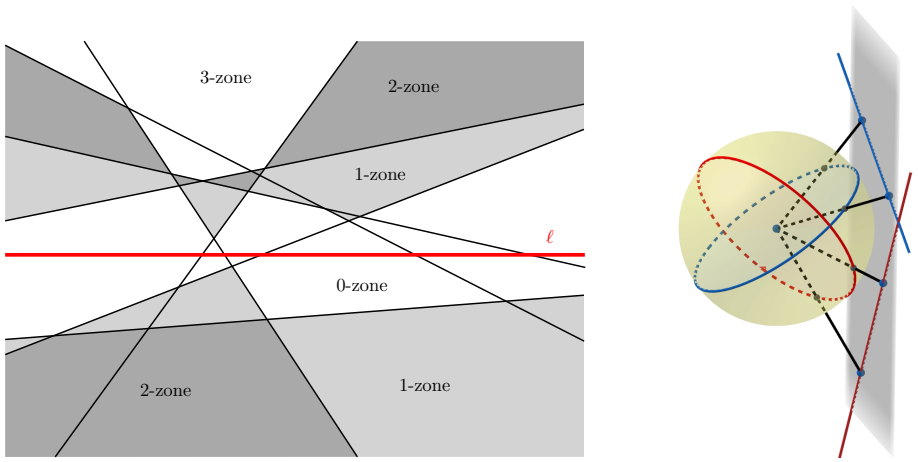


Figure 1 (a) The higher order zones of a line ℓ. (b) The correspondence between great-circles on the unit sphere and lines in a plane. Using the center of the sphere as the center of projection points on the sphere are projected to the points in the plane.

For arrangements of hyperplanes in $\mathbb{R}^d$ the $(\leq j)$-zone is defined alike. The classical zone theorem provides bounds for the zone ((≤ 0)-zone) of a hyperplane (cf. [9] and [15, Chapter 6.4]). A generalization with bounds for the complexity of the $(\leq j)$-zone appears as an exercise in Matoušek's book [15, Exercise 6.4.2]. In the proof of Theorem 2.1 we use a variant of the 2-dimensional case (Theorem 1.1). For the sake of completeness and to provide explicit constants, we include the proof in the full version [5].

▶ **Theorem 1.1.** *Let $\mathcal{L}$ be a simple arrangement of n lines in $\mathbb{R}^2$ and $\ell \in \mathcal{L}$. The $(\leq j)$-zone of ℓ contains at most $2e \cdot (j+2)n$ vertices strictly above ℓ.*

1.2 Arrangements of Great Circles

Let Π be a plane in 3-space which does not contain the origin and let $\mathbb{S}^2$ be a sphere in 3-space centered at the origin. The central projection Ψ_Π yields a bijection between arrangements of great circles on $\mathbb{S}^2$ and arrangements of lines in Π. Figure 1b gives an illustration.

The correspondence Ψ_Π preserves intersesting properties, e.g. simplicity of the arrangements. If $\Psi_\Pi(\mathcal{C}) = \mathcal{L}$, and $\mathcal{L}$ has no parallel lines, then Ψ_Π induces a bijection between pairs of antipodal vertices of $\mathcal{C}$ and vertices of $\mathcal{L}$.

As in the planar case, we define the *distance* between points x, y of $\mathbb{S}^2$ relative to a great-circle arrangement $\mathcal{C}$ as the minimum number of circles of $\mathcal{C}$ intersected by the interior of a curve connecting x with y. The *k-level* ($\leq k$-*zone* resp.) of $\mathcal{C}$ is the set of all the vertices of $\mathcal{C}$ at distance k (distance at most k resp.) from the south pole.

Let Π_1 and Π_2 be two parallel planes in 3 space with the origin between them and let Ψ_1 and Ψ_2 be the respective central projections. For a great-circle arrangement $\mathcal{C}$ we consider $\mathcal{L}_1 = \Psi_1(\mathcal{C})$ and $\mathcal{L}_2 = \Psi_2(\mathcal{C})$. A vertex v from the k-level of $\mathcal{C}$ maps to a vertex of the k-level in one of $\mathcal{L}_1$, $\mathcal{L}_2$ and to a vertex of the $(n-k-2)$-level in the other. Hence, bounds for the maximum size of the k-level of line arrangements carry over to the k-level of great-circle arrangements except for a multiplicative factor of 2.

The $(\leq j)$-zone of a great-circle C in $\mathcal{C}$ projects to a $(\leq j)$-zone of a line in each of $\mathcal{L}_1$ and $\mathcal{L}_2$. Hence, the complexity of a $(\leq j)$-zone in $\mathcal{C}$ is upper bounded by two times the maximum complexity of a $(\leq j)$-zone in a line arrangement. Theorem 1.1 implies that the $(\leq j)$-zone of a great-circle C in an arrangement of n great-circles contains at most $4e \cdot (j+2)n$ vertices.

1.3 Higher Dimensions

The problem of determining the complexity of the k-level admits a natural extension to higher dimensions. We consider arrangements in $\mathbb{R}^d$ of hyperplanes with the properties that no hyperplane is parallel to the x_d-axis and no $d+1$ hyperplanes intersect in a common point. The *k-level* $V_k(\mathcal{A})$ of $\mathcal{A}$ consists of all vertices (i.e. intersection points of d hyperplanes) which have exactly k hyperplanes of $\mathcal{A}$ below them (with respect to the d-th coordinate). We denote the k-level by $V_k(\mathcal{A})$ and its size by $f_k(\mathcal{A})$. Moreover, by $f_k^{(d)}(n)$ we denote the maximum of $f_k(\mathcal{A})$ among all arrangements $\mathcal{A}$ of n hyperplanes in $\mathbb{R}^d$.

As in the planar case, there remains a gap between lower and upper bounds;

$$\Omega(n^{\lfloor d/2 \rfloor} k^{\lceil d/2 \rceil - 1}) \leq f_k^{(d)}(n) \leq O(n^{\lfloor d/2 \rfloor} k^{\lceil d/2 \rceil - c_d}),$$

here $c_d > 0$ is a small positive constant only depending on d. Details and references can be found in Chapter 11 of Matoušek's book [15]. In dimensions 3 and 4 improved bounds have been established. For example, for $d = 3$, it is known that $f_k^{(3)}(n) \leq O(n(k+1)^{3/2})$ (see [21]). For the middle level in dimension $d \geq 2$ an improved lower bound $f^{(d)}(n) \geq n^{d-1} \cdot e^{\Omega(\sqrt{\log n})}$ is known (see [22] and [16]).

We call the intersection of $\mathbb{S}^d$ with a central hyperplane in $\mathbb{R}^{d+1}$ a *great-$(d-1)$-sphere* of $\mathbb{S}^d$. Similar to the planar case, arrangements of hyperplanes in $\mathbb{R}^d$ are in correspondence with arrangements of great-$(d-1)$-spheres on the unit sphere $\mathbb{S}^d$ (embedded in $\mathbb{R}^{d+1}$). The terms "distance" and "k-level" generalize in a natural way.

2 Our Results

In the first part of this paper we consider arrangements of great-circles on the sphere and investigate the average complexity of the k-level when the southpole is chosen uniformly

at random among the cells. This question was raised by Barba, Pilz, and Schnider while sharing a pizza [4, Question 4.2].

In Section 3 we prove the following bound on the average complexity.

▶ **Theorem 2.1.** *Let $\mathcal{C}$ be a simple arrangement of n great-circles. For $k < n/3$ the expected size of the k-level is at most $4e \cdot (k+2)^2$ when the southpole is chosen uniformly at random among the cells of $\mathcal{C}$.*

The condition $k < n/3$ is needed for Lemma 3.2 as for larger k we would have to double the multiplicative constant. However, for k in $\Omega(n^{3/5})$ the stated bound is implied by the $O(nk^{1/3})$ bound on the maximum size of a k-level. Still it is remarkable that the bound is independent of the number n of great-circles in the arrangement.

In the second part, we investigate arrangements of randomly chosen great-circles. Here we propose the following model of randomness. On $\mathbb{S}^2$ we have the duality between points and great-circles (each antipodal pair of points defines the normal vector of the plane containing a great-circle). Since we can choose points uniformly at random from $\mathbb{S}^2$, we get random arrangements of great-circles. The duality generalizes to higher dimensions so that we can talk about random arrangements on $\mathbb{S}^d$ for a fixed dimension $d \geq 2$. Using the duality between antipodal pairs of points on $\mathbb{S}^d$ and great-$(d-1)$-spheres, we prove the following bound on the expected size of the k-level in this random model (the proof can be found in the full version [5]). Again the bound does not depend on the size of the arrangement.

▶ **Theorem 2.2.** *Let $d \geq 2$ be fixed. In an arrangement of n great-$(d-1)$-spheres chosen uniformly at random on the unit sphere $\mathbb{S}^d$ (embedded in $\mathbb{R}^{d+1}$), the expected size of the k-level is of order $\Theta((k+1)^{d-1})$ for all $k \leq n/2$.*

3 Proof of Theorem 2.1

For the proof of Theorem 2.1, we fix a great-circle C from $\mathcal{C}$ and denote the closures of the two hemispheres of C on $\mathbb{S}^2$ as C^+ and C^-. As an intermediate step, we bound the size of the set $\mathcal{F}_k(C^+)$ of pairs (F, v), where F is a cell of C^- touching C and v is a vertex of C^+ whose distance to F is k. We show $|\mathcal{F}_k(C^+)| \leq 2e \cdot (k+1)^2 n$. In the case $k = 0$, vertex v must be one of the $2n$ vertices on C and F is one of the two cells of C^- which is adjacent to v. Hence, we obtain $|\mathcal{F}_0(C^+)| \leq 4n$. It remains to deal with the general case $k \geq 1$. Note that if $(v, F) \in \mathcal{F}_k(C^+)$ then v belongs to the $(\leq k-1)$-zone of C.

Consider a family $\mathcal{I}$ of half-intervals in $\mathbb{R}$, it consists of *left-intervals* of the form $(-\infty, a]$ and *right-intervals* $[b, \infty)$. A subset J of k half-intervals from $\mathcal{I}$ is a *k-clique* if there is a point $p \in \mathbb{R}$ that lies in all the half-intervals of J but not in any half-interval of $\mathcal{I} \setminus J$.

▶ **Lemma 3.1.** *Any family $\mathcal{H}$ of half-intervals in $\mathbb{R}$ contains at most $k+1$ different k-cliques.*

Proof. For $p \in \mathbb{R}$, let $l(p)$ be the number of left-intervals and $r(p)$ the number of right-intervals containing p. A point p certifies a k-clique if and only if $l(p) + r(p) = k$. From the monotonicity of the functions l and r it follows that if $(l(p_1), r(p_1)) = (l(p_2), r(p_2))$ for two points p_1 and p_2, then they are contained in the same intervals. Thus the number of k-cliques is at most the number of pairs (l, r) such that $l + r = k$ and $l, r \geq 0$, which is $k+1$. ◀

The next lemma is a corresponding result for half-circles on the circle $\mathbb{S}^1$.

▶ **Lemma 3.2.** *Any family $\mathcal{H}$ of n half-circles in $\mathbb{S}^1$ with $n > 3k$ contains at most $k+1$ different k-cliques.*

Proof. For this proof, we embed $\mathbb{S}^1$ as the unit-circle in $\mathbb{R}^2$, which is centered at the origin $\mathbf{o}$. We consider the set X of all points from $\mathbb{S}^1$, which are contained in precisely k of the half-circles of $\mathcal{I}$, and distinguish the following two cases.

Case 1: The origin $\mathbf{o}$ is not contained in the convex hull of X. There is a line separating $\mathbf{o}$ from X and rotational symmetry allows us to assume that X is contained in $\Pi^+ = \{(x, y) \in \mathbb{R}^2 : y > 0\}$. For each half-circle $C \in \mathcal{H}$, the central projection of $C \cap \Pi^+$ to the line $y = 1$ is a half-interval. Since k-cliques of $\mathcal{H}$ and k-cliques of the half-intervals are in bijection we get from Lemma 3.1 that $\mathcal{H}$ has at most $k + 1$ different k-cliques.

Case 2: The origin $\mathbf{o}$ is contained in the convex hull of X. By Carathéodory's theorem, we can find three points p_1, p_2, p_3 such that $\mathbf{o}$ lies in the convex hull of p_1, p_2, p_3. Since each of the n half-circles from $\mathcal{H}$ contains at least one of these three points, and each of these three points lies on precisely k half-circles, we have $n \leq 3k$ – a contradiction to $n > 3k$. ◀

For a fixed vertex v in the $(\leq k-1)$-zone of C with $v \in C^+$, let $\mathcal{B}_{C^+}(v)$ be the set of cells F such that $(F, v) \in \mathcal{F}_k(C^+)$, in particular $\mathrm{dist}(F, v) = k$.

Claim. For $k \geq 1$, we have $|\mathcal{B}_{C^+}(v)| \leq k$.

Proof. Consider a great-circle $D \neq C$ from $\mathcal{C}$. For a point $x \in C$, we say that (v, x) is D-*separated* if every path from v to x in C^+ intersects D. The set of all D-*separated* points forms a half-circle H_D on C. Let $\mathcal{H}$ be the set of these half-circles, i.e., $\mathcal{H} = \{H_D : D \in \mathcal{C}, D \neq C\}$. See Figure 2.

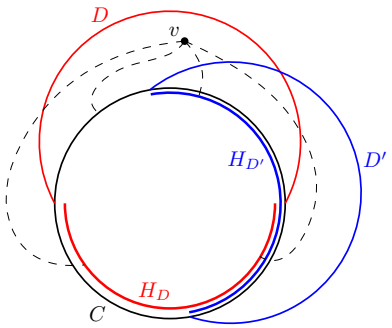


Figure 2 An illustration of the cyclic half-circles $\mathcal{H}$.

We claim that there is a bijection between $\mathcal{B}_{C^+}(v)$ and the $(k-1)$-cliques in $\mathcal{H}$. Indeed, if the intersection of the half-circles of a clique K, viewed as a subset of C, is I_K, then I_K is the interval of C which is reachable from v by crossing the circles corresponding to the half-circles of K. If F is a cell from C^- at distance k from v, then C and a subset of $k-1$ additional circles have to be crossed to reach v from F, i.e., there is a $(k-1)$-clique in $\mathcal{H}$ whose intersection is $F \cap C$. The number of $(k-1)$-cliques in $\mathcal{H}$ is at most k by Lemma 3.2. ◀

Claim. For $k \geq 1$, we have $|\mathcal{F}_k(C^+)| \leq 2e \cdot k(k+1)n$.

Proof. By definition, the set $\mathcal{F}_k(C^+)$ is the set of pairs (F, v) such that $v \in C^+$ is in the $(\leq k)$-zone of C and $F \in \mathcal{B}_{C^+}(v)$. As already noted in Section 1.2, the $(\leq k)$-zone contains at most $4e \cdot (k+1)n$ vertices of $\mathcal{C}$ and at most $2e \cdot (k+1)n$ vertices in C^+. From the above claim we have $|\mathcal{B}_{C^+}(v)| \leq k$, hence we conclude that $|\mathcal{F}_k(C^+)| \leq 2e \cdot k(k+1)n$. ◀

To include the case $k = 0$ we relax the bound to $|\mathcal{F}_k(C^+)| \leq 2e \cdot (k+1)^2 n$. Since C was chosen arbitrarily among all great-circles from $\mathcal{C}$ and C^+ was chosen arbitrarily among the two hemispheres of C, the upper bound from the above claim holds for any induced hemisphere of $\mathcal{C}$. For the union $\mathcal{F}_k$ of the $\mathcal{F}_k(C^+)$ over all the $2n$ choices of the hemisphere C^+, we have

$$|\mathcal{F}_k| \leq \sum_{C^+ \text{ hemisphere}} |\mathcal{F}_k(C^+)| \leq 4e(k+1)^2 n^2.$$

Proof of Theorem 2.1. The k-level with the southpole chosen in cell F consists of the vertices at distance k from F. Thus, the expected complexity of the k-level when choosing F uniformly at random equals $|\mathcal{F}_k|$ divided by the number of cells. Since the number of cells in an arrangement of n great-circles is $2\binom{n}{2} + 2$ and $|\mathcal{F}_k| \leq 4e(k+1)^2 n^2$, we can conclude the statement from

$$\frac{4e \cdot (k+1)^2 \cdot n^2}{2\binom{n}{2} + 2} \leq 4e \cdot (k+1)^2 \cdot \frac{n}{n-1} \leq 4e \cdot (k+2)^2 \cdot \underbrace{\frac{k+1}{k+2} \cdot \frac{n}{n-1}}_{\leq 1}. \qquad \blacktriangleleft$$

4 Discussion

Theorem 2.1 is about arrangements of great-circles. All the elements of the proof, however, carry over to great-pseudocircles whence the result could also be stated for arrangements of great-pseudocircles. Projective arrangements of lines are obtained by antipodal identification from arrangements of great-circles. Hence, if you pick a cell u.a.r. in a projective arrangement of lines (pseudo-lines) the the expected number of vertices at distance k from the cell is as in Theorem 2.1. If the projection Ψ_Π is used to project an arrangements $\mathcal{C}$ of great-pseudocircles to an Euclidean arrangement $\mathcal{L}$ on Π such that the south-poles coincide, then the k-level of $\mathcal{C}$ corresponds to the union of the k- and the $(n-k-2)$-level of $\mathcal{L}$.

With respect to lower bounds we would like to know the answer to:

▶ Question 1. Is there a family of arrangements where the expected size of the middle level is superlinear when the southpole is chosen uniformly at random?

Recursive constructions from [10] and [11] show that the size of the $(n/2 - s)$-level can be in $\Omega(n \log n)$ for any fixed s. Nevertheless computer experiments suggest that if we choose a random southpole for these examples the expected size of the middle level drops to be linear.

Theorem 2.2 deals with the average size of the k-level in arrangements of randomly chosen great-circles. In our model, great-circles are chosen independently and uniformly at random from the sphere. Since point sets, line arrangements, and great-circle arrangements are in strong correspondence the bound from Theorem 2.2 also applies to k-sets in point sets and k-levels of line arrangements from a specific random distribution.

In the context of Erdős–Szekeres-type problems, several articles made use of point sets which are sampled uniformly at random from a convex shape [3, 23, 2, 1]. Also the average size of the convex hull (0-level) is well-studied for sets of points which are sampled uniformly at random from a convex shape K. If K is a disk, the convex hull has expected size $O(n^{1/3})$, and if K is a convex polygon with k sides, the expected size is $O(k \log n)$ [13, 18, 19, 20]. In particular, the expected size of the convex hull is not constant, which is a substantial contrast to our setting. In fact, our setting appears to be closer to the setting of random order types, for which the expected size of the convex hull was recently shown to be $4 + o(1)$

[12]. Hence it would be very interesting to obtain bounds on the average number of k-sets also in this setting. Last but not least, Edelman [7] showed that the expected number of k-sets of an allowable sequence is of order $\Theta(\sqrt{kn})$.

References

1. M. Balko, M. Scheucher, and P. Valtr. Holes and islands in random point sets, 2020. To appear at the 36th International Symposium on Computational Geometry (SoCG 2020).
2. J. Balogh, H. González-Aguilar, and G. Salazar. Large convex holes in random point sets. *Computational Geometry*, 46(6):725–733, 2013.
3. I. Bárány and Z. Füredi. Empty simplices in Euclidean space. *Canadian Mathematical Bulletin*, 30(4):436–445, 1987.
4. L. Barba, A. Pilz, and P. Schnider. Sharing a pizza: bisecting masses with two cuts, 2019. arXiv:1904.02502.
5. M.-K. Chiu, S. Felsner, M. Scheucher, P. Schnider, R. Steiner, and P. Valtr. On the Average Complexity of the k-Level, 2019. arXiv:1911.02408.
6. T. K. Dey. Improved bounds for planar k-sets and related problems. *Discrete & Computational Geometry*, 19(3):373–382, 1998.
7. P. H. Edelman. On the average number of k-sets. *Discrete & Computational Geometry*, 8:209–213, 1992.
8. H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer, 1987.
9. H. Edelsbrunner, R. Seidel, and M. Sharir. On the Zone Theorem for Hyperplane Arrangements. In *New Results and New Trends in Computer Science*, pages 108–123. Springer, 1991.
10. H. Edelsbrunner and E. Welzl. On the number of line separations of a finite set in the plane. *Journal of Combinatorial Theory, Series A*, 38:15–29, 1985.
11. P. Erdős, L. Lovász, G. J. Simmons, and E. G. Straus. Chapter 13 - dissection graphs of planar point sets. In *A Survey of Combinatorial Theory*, pages 139–149. North-Holland, 1973.
12. X. Goaoc and E. Welzl. Convex hulls of random order types. To appear at the 36th International Symposium on Computational Geometry (SoCG 2020), 2020.
13. S. Har-Peled. On the expected complexity of random convex hulls, 2011. arXiv:1111.5340.
14. L. Lovász. On the number of halving lines. *Annales Universitatis Scientiarum Budapestinensis de Rolando Eötvös Nominatae Sectio Mathematica*, 14:107–108, 1971.
15. J. Matoušek. *Lectures on Discrete Geometry*. Springer, 2002.
16. G. Nivasch. An improved, simple construction of many halving edges. *Contemporary Mathematics*, 453:299–305, 2008.
17. J. O'Rourke. *Computational Geometry in C*. Cambridge University Press, 1994.
18. F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer, 1985.
19. H. Raynaud. Sur l'enveloppe convexe des nuages de points aleatoires dans $\mathbb{R}^n$. *Journal of Applied Probability*, 7(1):35–48, 1970.
20. A. Rényi and R. Sulanke. Über die konvexe Hülle von n zufällig gewählten Punkten. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 2(1):75–84, 1963.
21. M. Sharir, S. Smorodinsky, and G. Tardos. An improved bound for k-sets in three dimensions. *Discrete & Computational Geometry*, 26(2):195–204, Jan 2001.
22. G. Tóth. Point Sets with Many k-Sets. *Discrete & Computational Geometry*, 26(2):187–194, 2001.
23. P. Valtr. On the minimum number of empty polygons in planar point sets. *Studia Scientiarum Mathematicarum Hungarica*, pages 155–163, 1995.

Topological Drawings meet Classical Theorems from Convex Geometry*

Helena Bergold[1], Stefan Felsner[2], Manfred Scheucher[2], Felix Schröder[2], and Raphael Steiner[2]

1 **Fakultät für Mathematik und Informatik, FernUniversität in Hagen, Germany,**
{helena.bergold}@fernuni-hagen.de

2 **Institut für Mathematik, Technische Universität Berlin, Germany,**
{felsner,scheucher,fschroed,steiner}@math.tu-berlin.de

Abstract

In this article, we discuss classical theorems from Convex Geometry in the context of simple topological drawings of the complete graph K_n. In a simple topological drawing, any two edges share at most one point: either a common vertex or a point where they cross.

We present generalizations of Kirchberger's Theorem and the Erdős–Szekeres Theorem, a family of simple topological drawings with arbitrarily large Helly number, a new proof of the generalized Carathéodory's Theorem, and discuss further classical theorems from Convex Geometry in the context of simple topological drawings.

1 Introduction

A point set in the plane (in general position) induces a straight-line drawing of the complete graph K_n. In this article we investigate topological drawings of K_n and use the triangles of such drawings for convexity related studies. In a *topological drawing* D of K_n, vertices are mapped to points in the plane and edges are mapped to simple curves connecting the corresponding end-points such that every pair of edges has at most one common point, which is either a common endpoint or a crossing; see Figure 1. Note that several edges may cross in a single point. A topological drawing is called *straight-line* if all edges are drawn as line segments, and *pseudolinear* if all arcs of the drawing can be extended to bi-infinite curves such that any two of these curves cross at most once (the family of curves is an arrangement of pseudolines).

Figure 1 Forbidden patterns in topological drawings: self-crossings, double-crossings, touchings, and crossings of adjacent edges.

1.1 Our Results

In Section 2, we present a new combinatorial generalization of topological drawings – which we name *generalized signotopes*. They allow us to prove a generalization of Kirchberger's

* S. Felsner and M. Scheucher were supported by DFG Grant FE 340/12-1. R. Steiner was supported by DFG-GRK 2434. We thank Winfried Hochstättler for valuable discussions and helpful comments.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Theorem in Section 3. We present a family of topological drawings with arbitrarily large Helly number in Section 4, and, in Section 5, we discuss generalizations of Carathéodory's Theorem, Erdős–Szekeres Theorem, and colorful variants of the classical theorems.

In the full version, we also describe the SAT model, which we used to test hypotheses about simple topological drawings, and give a more detailed analysis of the named theorems in terms of the *convexity* hierarchy introduced by Arroyo, McQuillan, Richter, and Salazar [1]. In particular, some of the theorems turn out to generalize to pseudolinear drawings but not to *pseudocircular* drawings, i.e., topological drawings where all arcs can be simultaneously extended to pseudocircles such that any two do not touch and cross at most twice.

2 Preliminaries

Given a topological drawing D of K_n, we call the induced subdrawing of three vertices a *triangle*. Note that the edges of a triangle in a topological drawing do not cross. The removal of a triangle separates the plane into two connected components. A point p is in the *interior* of a triangle or more generally of a topological drawing D if p is in a bounded connected component of $\mathbb{R}^2 - D$.

In a topological drawing, we assign an *orientation* $\chi(abc) \in \{+, -\}$ to each ordered triple abc of vertices. The sign $\chi(abc)$ indicates whether we go counterclockwise or clockwise around the triangle when visiting the vertices a, b, c in this order.

In a straight-line drawing of K_n the underlying point set $S = \{s_1, \ldots, s_n\}$ is in general position (no three points lie on a line). If the points are sorted from left to right, then for every 4-tuple s_i, s_j, s_k, s_l with $i < j < k < l$ the sequence $\chi(ijk), \chi(ijl), \chi(ikl), \chi(jkl)$ (index-triples in lexicographic order) is *monotone*, i.e., there is at most one sign-change. A *signotope* is a mapping $\chi : \binom{[n]}{3} \to \{+, -\}$ with the above monotonicity property. Signotopes are in bijection with Euclidean pseudoline arrangements and can be used to characterize pseudolinear drawings [7, 3].

Let us now consider topological drawings of the complete graph. There are two types of drawings of K_4 on the sphere: type I has a crossing and type II has no crossing. Type I can be drawn in two different ways in the plane: in type I_a the crossing is only incident to bounded faces and in type I_b the crossing lies on the outer face; see Figure 2.

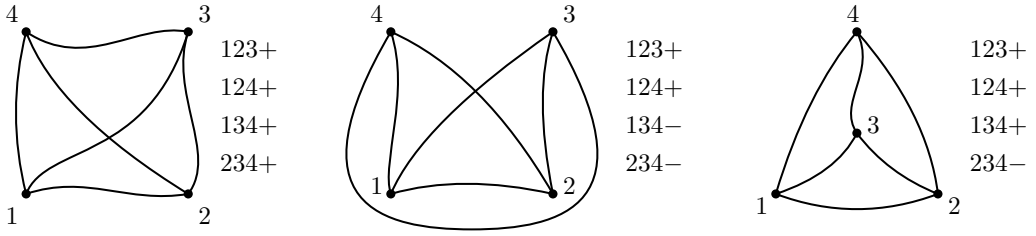


Figure 2 The three types of topological drawings of K_4 in the plane.

A drawing of K_4 with vertices a, b, c, d can be characterized in terms of the sequence of orientations $\chi(abc), \chi(abd), \chi(acd), \chi(bcd)$. The drawing is

- of type I_a or type I_b iff the sequence is $++++$, $++--$, $+--+$, $-++-$, $--++$, or $----$; and
- of type II iff the number of $+$'s (and $-$'s respectively) in the sequence is odd.

Therefore there are at most two sign-changes in the sequence and, moreover, any such sequence is in fact induced by a topological drawing of K_4. Allowing up to two sign-changes is equivalent to forbidding the two patterns $+-+-$ and $-+-+$.

If a mapping $\chi : [n]_3 \to \{+,-\}$ is *alternating*, i.e., $\chi(i_{\sigma(1)}, i_{\sigma(2)}, i_{\sigma(3)}) = \operatorname{sgn}(\sigma) \cdot \chi(i_1, i_2, i_3)$, and χ avoids the two patterns on sorted indices, i.e., $\chi(ijk), \chi(ijl), \chi(ikl), \chi(jkl)$ has at most two sign-changes for $i < j < k < l$, then it avoids the two patterns on $\chi(abc), \chi(abd), \chi(acd), \chi(bcd)$ for any pairwise different $a, b, c, d \in [n]$. We refer to this as the *symmetry property* of the forbidden patterns.

The symmetry property allows us to define *generalized signotopes* as alternating mappings $\chi : [n]_3 \to \{+,-\}$ with at most two sign-changes on $\chi(abc), \chi(abd), \chi(acd), \chi(bcd)$ for any pairwise different $a, b, c, d \in [n]$. We remark that generalized signotopes can be seen as a proper generalization of topological drawings of K_n – details are deferred to the full version.

3 Kirchberger's Theorem

Two point sets $A, B \subseteq \mathbb{R}^d$ are called *separable* if there exists a hyperplane H separating them. It is well-known that if two sets A, B are separable they can also be separated by a hyperplane H containing one point of A and B. *Kirchberger's Theorem* (see [12] or [5]) asserts that two finite point sets $A, B \subseteq \mathbb{R}^d$ are separable if and only if for every $C \subseteq A \cup B$ with $|C| = d + 2$, $C \cap A$ and $C \cap B$ are separable.

We prove a generalization of the 2-dimensional version of Kirchberger's Theorem in the setting of generalized signotopes, where two sets $A, B \subseteq [n]$ are *separable* if there exist $i, j \in A \cup B$ such that $\chi(i, j, x) = +$ for all $x \in A \setminus \{i, j\}$ and $\chi(i, j, x) = -$ for all $x \in B \setminus \{i, j\}$. In this case we say that ij *separates* A from B and write $\chi(i, j, A) = +$ and $\chi(i, j, B) = -$. Moreover, if we can find $i \in A$ and $j \in B$, we say that A and B are *strongly separable*.

▶ **Theorem 3.1** (Kirchberger's Theorem for Generalized Signotopes). *Let $\chi : [n]_3 \to \{+,-\}$ be a generalized signotope, and $A, B \subseteq [n]$. If for every $C \subseteq A \cup B$ with $|C| = 4$, the sets $A \cap C$ and $B \cap C$ are (strongly) separable, then A and B are (strongly) separable.*

Proof. First note that it is sufficient to prove the theorem for strongly separable since all 4-tuples which are weakly separable are also strongly separable. More details will be given in the full version. Hence in the following we always assume that the 4-tuples are strongly separable.

To prove the claim we consider a counterexample (χ, A, B) minimizing the size of the smaller of the two sets. By symmetry we may assume $|A| \leq |B|$. First we consider the cases $|A| = 1, 2, 3$ individually and then the case $|A| \geq 4$.

Let $A = \{a\}$, we need to find $b \in B$ such that $\chi(a, b, B) = -$. Let B' be a maximal subset of B such that B' can be separated from $\{a\}$, and let $b \in B$ be such that $\chi(a, b, B') = -$. Suppose that $B' \neq B$, then there is a $b^* \in B \setminus B'$ with

$$\chi(a, b, b^*) = +. \tag{1}$$

By maximality of B', the sets $\{a\}$ and $B' \cup \{b^*\}$ cannot be separated. Hence, we have

$$\chi(a, b^*, b') = + \tag{2}$$

for some $b' \in B'$. Since χ is alternating (1) and (2) together imply $b' \neq b$. Since $b' \in B'$ we have $\chi(a, b, b') = -$. From this together with (1) and (2) it follows that the four element set $\{a, b, b', b^*\}$ has no separator. This is a contradiction, whence $B' = B$.

As a consequence we obtain:

- Every one-element set $\{a\}$ with $a \in A$ can be separated from B. Since χ is alternating there can be at most one $b(a) \in B$ such that $\chi(a, b(a), B) = -$.

Now we look at the case where $A = \{a_1, a_2\}$. Let $b_i = b(a_i)$, if $b_1 = b_2$, or $\chi(a_1, b_1, a_2) = +$, or $\chi(a_2, b_2, a_1) = +$ we have a separator for A and B. So assume that $b_1 \neq b_2$, and $\chi(a_1, b_1, a_2) = -$, and $\chi(a_2, b_2, a_1) = -$. Since χ is alternating we also know that $\chi(a_1, b_2, b_1) = +$ and $\chi(a_2, b_1, b_2) = +$. Together these four signs show that $\{a_1, b_1, a_2, b_2\}$ is not separable, a contradiction.

The case $|A| = 3$ works similarly but is more technical. A proof of this case, will be given in the full version.

Now we consider the remaining case where (χ, A, B) is a minimal counterexample with $4 \leq |A| \leq |B|$.

Let $a^* \in A$. By minimality of (χ, A, B), $A \backslash \{a^*\} = A'$ is separable from B. Let $a \in A'$ and $b \in B$ such that $\chi(a, b, A') = +$ and $\chi(a, b, B) = -$. Hence it is

$$\chi(a, b, a^*) = -. \tag{3}$$

Let $b^* = b(a^*)$, i.e., $\chi(a^*, b^*, B) = -$. There is some $a' \in A'$ such that

$$\chi(a^*, b^*, a') = -. \tag{4}$$

If $a' = a$, then $b \neq b^*$ because of (3) and (4). From $\chi(a, b, B) = -$, $\chi(a^*, b^*, B) = -$, (3), and (4) it follows that the 4-element set $\{a, b, a^*, b^*\}$ has no separation. The contradiction shows $a' \neq a$.

Let $b' = b(a')$. If $b = b'$, then $a' \in A'$ implies $\chi(a, b, a') = +$ which yields $\chi(a', b', a) = -$. If $b \neq b'$ we look at the four elements a, b, a', b', and have:

$$\chi(a', b', a) =? \ , \quad \chi(a', b', b) = - \ , \quad \chi(a', a, b) = + \ , \quad \chi(b', a, b) = -$$

To avoid the forbidden pattern for $a'b'ab$ we must have $\chi(a', b', a) = -$.

Hence, regardless whether $b = b'$ or $b \neq b'$ we have

$$\chi(a', b', a) = - \ . \tag{5}$$

Since $|A| \geq 4$, we know by the minimality of (χ, A, B) that the set $\{a, b, a', b', a^*, b^*\}$, which has 3 elements of A and at least 4 elements in total, is separable. It follows from $\chi(a, b, B) = \chi(a', b', B) = \chi(a^*, b^*, B) = +$ that the only possible separators are ab, $a'b'$, and a^*b^*. They, however, do not separate because of (3), (5), and (4) respectively. This is impossible, hence, there is no counterexample. ◀

4 Helly's Theorem

Helly's Theorem asserts that the intersection of n convex sets $S_1, \ldots, S_n$ in $\mathbb{R}^d$ is non-empty if the intersection of every $d+1$ of these sets in non-empty. In other words, the *Helly number* of a family of n convex sets in $\mathbb{R}^d$ is at most $d+1$. The result of Goodman and Pollack [9] (see also [2]) shows that Helly's Theorem holds for pseudoconfigurations of points in two dimensions, and thus for pseudolinear drawings.

In the more general setting of topological drawings, we investigate the intersection properties of triangles. We show that Helly's Theorem does not generalize to topological

drawings by constructing topological drawings with arbitrarily large Helly number. Note that the following theorem does not contradict the topological Helly Theorem [10] (cf. [8]) because the intersections of two triangles are not connected.

▶ **Theorem 4.1.** *For every odd integer n, there exists a topological drawing of K_{3n} with Helly number at least n, i.e., there are n triangles such that any $n-1$ have a common interior, but not all n have a common interior.*

Sketch of the proof. The basic idea of the construction is to draw n triangles on the cylinder with the property that any $n-1$ triangles have a common interior, while there is no common intersection of all n triangles; the left-hand side of Figure 3 gives an illustration. Such a cylindrical drawing can clearly be drawn in the plane as depicted on the right-hand side of Figure 3. The technical part, however, is to complete such a drawing of n triangles to a topological drawing of the complete graph K_{3n}. Technical details are deferred to the full version.

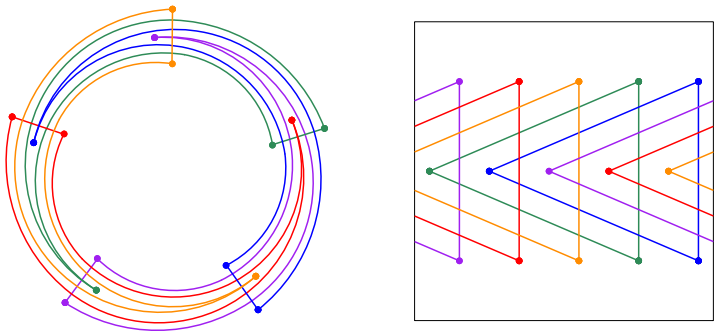


Figure 3 An illustration of a topological drawing of K_{3n} with Helly number n.

For the construction, we identify the cylinder surface with the real plane $\mathbb{R}^2$. As illustrated in Figure 4, we place the vertices of K_{3n} on 3 different layers.

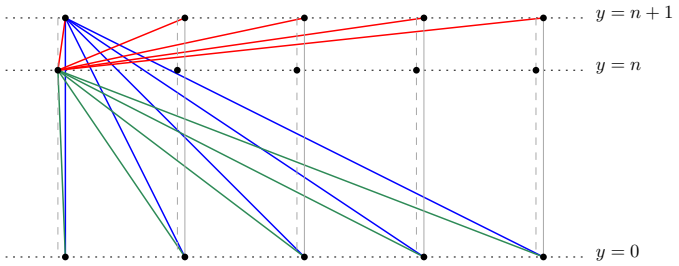


Figure 4 Placement of the vertices and drawing edges between different layers.

The edges between different layers are drawn as straight-line segments (see Figure 4). Edges between two vertices of the same layer are drawn as circular arcs above the top layer, below the bottom layer and in a sufficiently small range above the middle layer (see Figure 5). The obtained drawing is topological (details will be given in the full version), which concludes the proof. ◀

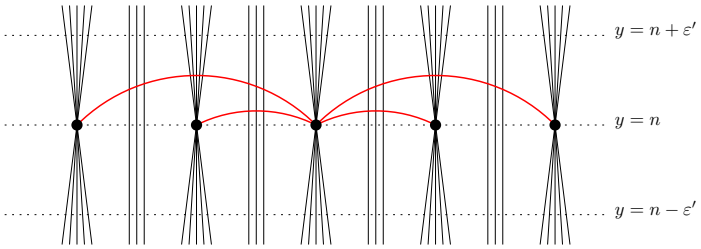


Figure 5 Edges between middle-layer vertices are drawn as very flat circular arcs.

5 Further Results

Carathéodory's Theorem asserts that, if a point x lies in the convex hull of a point set P in $\mathbb{R}^d$, then x lies in the convex hull of at most $d+1$ points of P. A more general version of Carathéodory's Theorem in the plane is by Balko, Fulek, and Kynčl, who provided a generalization to topological drawings [3, Lemma 4.7]. In the full version, we present a new proof for their theorem.

▶ **Theorem 5.1** (Carathéodory for Topological Drawings [3]). *Let D be a topological drawing of K_n and let $x \in \mathbb{R}^2$ be a point in the interior of D. Then there is a triangle in D which contains x in its interior.*

In the full version, we also study colorful variants of the theorems. For example it turned out that Barany's Colorful Carathéodory Theorem [4] holds up to pseudolinear drawings [11] but not for pseudocircular drawings (see Figure 6).

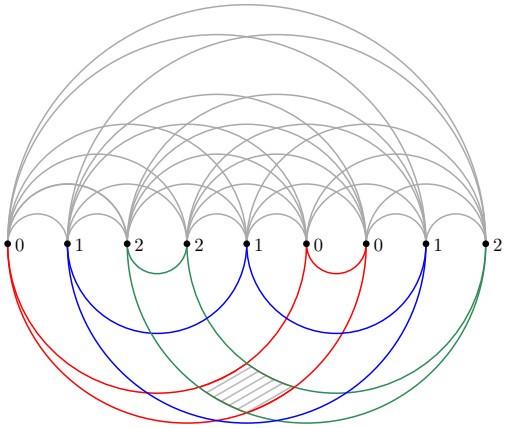


Figure 6 A circular drawing of K_9 violating the condition of Colorful Carathéodory Theorem.

The classical *Erdős–Szekeres Theorem* [6] asserts that every straight-line drawing of K_n contains a crossing-maximal subdrawing of size $k = \Omega(\log n)$, that is, a subdrawing of K_k with $\binom{k}{4}$ crossings. Pach, Solymosi, and Tóth [13] generalized this result by showing that every topological drawing of K_n has a crossing-maximal subdrawing of size $k = \Omega(\log^{1/8} n)$. A simple Ramsey–type argument shows that a variant of the Erdős–Szekeres Theorem even applies to generalized signotopes.

References

1 A. Arroyo, D. McQuillan, R. B. Richter, and G. Salazar. Convex drawings of the complete graph: topology meets geometry. arXiv:1712.06380, 2017.

2 A. Bachem and A. Wanka. Separation theorems for oriented matroids. *Discrete Mathematics*, 70(3):303–310, 1988.

3 M. Balko, R. Fulek, and J. Kynčl. Crossing Numbers and Combinatorial Characterization of Monotone Drawings of K_n. *Discrete & Computational Geometry*, 53(1):107–143, 2015.

4 I. Bárány. A generalization of Carathéodory's Theorem. *Discrete Mathematics*, 40(2):141–152, 1982.

5 A. Barvinok. *A course in convexity*, volume 54 of *Graduate Studies in Mathematics*. American Mathematical Society, 2002.

6 P. Erdős and G. Szekeres. A combinatorial problem in geometry. *Compositio Mathematica*, 2:463–470, 1935.

7 S. Felsner and H. Weil. Sweeps, Arrangements and Signotopes. *Discrete Applied Mathematics*, 109(1):67–94, 2001.

8 X. Goaoc, P. Paták, Z. Patáková, M. Tancer, and U. Wagner. Bounding helly numbers via betti numbers. In *A Journey Through Discrete Mathematics: A Tribute to Jiří Matoušek*, pages 407–447. Springer, 2017.

9 J. E. Goodman and R. Pollack. Helly-type theorems for pseudoline arrangements in $\mathcal{P}^2$. *Journal of Combinatorial Theory, Series A*, 32(1):1–19, 1982.

10 E. Helly. Über Systeme von abgeschlossenen Mengen mit gemeinschaftlichen Punkten. *Monatshefte für Mathematik Band 37*, pages 281–302, 1930.

11 A. F. Holmsen. The intersection of a matroid and an oriented matroid. *Advances in Mathematics*, 290:1–14, 02 2016.

12 P. Kirchberger. Über Tchebychefsche Annäherungsmethoden. *Mathematische Annalen*, 57:509–540, 1903.

13 J. Pach, J. Solymosi, and G. Tóth. Unavoidable configurations in complete topological graphs. *Discrete & Computational Geometry*, 30(2):311–320, 2003.

On the width of the monotone-visibility kernel of a simple polygon

David Orden[1], Leonidas Palios[2], Carlos Seara[3], Jorge Urrutia[4], and Paweł Żyliński[5]

1 **Departamento de Física y Matemáticas, Universidad de Alcalá, Spain,** david.orden@uah.es
2 **Dept. of Computer Science and Engineering, University of Ioannina, Greece,** palios@cs.uoi.gr
3 **Mathematics Department, Universitat Politècnica de Catalunya, Spain,** carlos.seara@upc.edu
4 **Instituto de Matemáticas, Universidad Nacional Autónoma de México, México,** urrutia@matem.unam.mx
5 **Institute of Informatics, University of Gdańsk, Poland,** zylinski@inf.ug.edu.pl

Abstract

Given a simple polygonal region P with n vertices, we present an efficient $O(n \log n)$ time and $O(n)$ space algorithm for computing, over all values of angle θ, the maximum width of the θ-kernel(P), i.e., the locus of points in P from which any point of P can be reached by a $(\theta + \frac{\pi}{2})$-monotone path.

1 Introduction

The computation of the *widest corridor through a set S of n points* in the plane, defined as an open region bounded by two parallel lines that intersect the convex hull of S, has attracted the interest of the computational geometry community since the late 1980s and early 1990s, having applications to robot motion planning [2, 4, 5]. Computing the *width of a simple polygon P with n vertices*, defined as the width of the narrowest corridor containing P, is another well-known problem in computational geometry [8]; it can be computed in linear time using what are known as rotating calipers [10].

The present work deals with the width of a particular corridor through a simple polygonal region P: the θ-kernel of P, denoted as θ-kernel(P). The θ-kernel(P) is defined as the locus of points in P from which any point of P can be reached by a path which has a connected intersection with any line forming an angle θ with the positive x-axis or, in other words, which is monotone with respect to the direction $\theta + \frac{\pi}{2}$. Figure 1 shows the θ-kernel of a polygon for three different values of the angle θ, together with an example of a $\frac{\pi}{2}$-monotone path from a point inside the 0-kernel. For further details see [6]. We present an efficient $O(n \log n)$ time and $O(n)$ space algorithm that, given a simple polygonal region P with n vertices, finds an angle $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2})$ for which the width of the θ-kernel(P) is maximized.

This work has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 734922.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

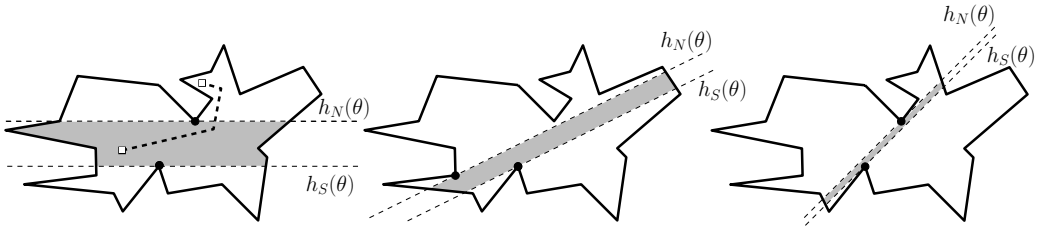


Figure 1 The θ-kernel(P) is shaded for $\theta = 0$ (left), for $\theta = \frac{\pi}{8}$ (middle), and for $\theta = \frac{\pi}{4}$ (right). In addition, a vertically monotone path from a point inside the 0-kernel is shown.

In the following we assume that the vertices of P are labeled $\{p_1, p_2, \ldots, p_n\}$ in counterclockwise order around the boundary of P and that a θ-orientation will be an oriented line forming angle θ with the x-axis.

▶ **Definition 1.1.** A reflex vertex $p_i \in P$ such that p_{i-1} and p_{i+1} are both below (resp. above) p_i with respect to a given θ-orientation is a *reflex maximum* (resp. *reflex minimum*) with respect to the θ-orientation. An edge with orientation θ such that its two neighbors are below (resp. above) with respect to that θ-orientation is also a reflex maximum (resp. minimum) edge with respect to the θ-orientation.

For a given θ-orientation, P can have several reflex minima and reflex maxima, so there is a lowest reflex minimum and a highest reflex maximum with respect to θ. It may also happen that P has no reflex maxima (resp. minima), and then the role of the highest reflex maximum (resp. lowest reflex minimum) will be undertaken by the lowest (resp. highest) vertex of P. See Figure 2, left.

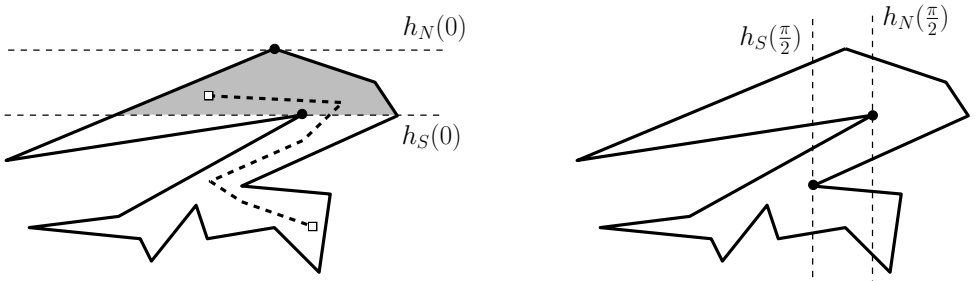


Figure 2 Left: The shaded area is the 0-kernel of a polygonal region P, where the role of the lowest reflex minimum is undertaken by the highest vertex of P. A vertically monotone path from a point in the kernel is also shown. Right: The $\frac{\pi}{2}$-kernel of the same P turns out to be empty.

▶ **Definition 1.2.** A pair (p, q) of vertices of P forms a *pair of antipodal interior vertices* of P for a θ-orientation if p is the lowest reflex minimum vertex and q is the highest reflex maximum vertex with respect to the θ-orientation, or vice versa.

In Figure 1 the pair of antipodal points is the same for $\theta = 0$ (left) and $\theta = \frac{\pi}{4}$ (right), although the pair has changed in between for $\theta = \frac{\pi}{8}$ (middle). Let $h_N(\theta)$ and $h_S(\theta)$ be, respectively, the θ-orientations passing through the lowest reflex minimum and the highest reflex maximum with respect to the current θ-orientation.

▶ **Lemma 1.3** ([9]). *The θ-kernel(P) is empty when $h_N(\theta)$ is below $h_S(\theta)$ with respect to θ. Otherwise, it is given by the intersection of P and the strip determined by $h_N(\theta)$ and $h_S(\theta)$.*

Figure 2, right, shows an example where the $\frac{\pi}{2}$-kernel(P) is empty because, with respect to the $\frac{\pi}{2}$-orientation, $h_N(\frac{\pi}{2})$ is below (to the right of) $h_S(\frac{\pi}{2})$. That is, there is no point from which all other points could be reached by a horizontally monotone path.

In order to compute the maximum width of the θ-kernel(P) we maintain the lines $h_N(\theta)$ and $h_S(\theta)$ enclosing the θ-kernel(P) for $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2})$. As the pairs of antipodal interior vertices of P can change during the process, we subdivide $[-\frac{\pi}{2}, \frac{\pi}{2})$ into subintervals $[\theta_i, \theta_{i+1})$ where the pair of antipodal interior vertices does not change, so that it is enough to compute the maximum value of the width of the θ-kernel(P) for θ in each of these subintervals $[\theta_i, \theta_{i+1})$.

2 Computing the maximum width of the θ-kernel(P)

Next, we sketch the algorithm to compute the intervals of the values of θ within $[-\frac{\pi}{2}, \frac{\pi}{2})$ such that θ-kernel(P) $\neq \emptyset$; i.e., the width of the θ-kernel(P), is not zero. Then we calculate the maximum width, and maintain its maximum value over all the intervals.

ALGORITHM FOR COMPUTING INTERVALS SUCH THAT θ-kernel(P) $\neq \emptyset$

1. For each vertex $p_i \in P$, check whether p_i is reflex. If it is, compute the angular intervals $[\theta_1^i, \theta_2^i)$ and $[\theta_1^i + \pi, \theta_2^i + \pi)$ of orientations θ for which p_i is reflex, and the corresponding *reflex slope intervals* defined when rotating with pivot p_i the line containing the edge $p_{i-1}p_i$ up to the line containing the edge p_ip_{i+1} (see Figure 3, left). Then the vertex p_i is a candidate to be a reflex maximum and a reflex minimum only for the θ-orientations in those reflex slope intervals. Note that, since $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2})$, a reflex slope interval may be split into two if it contains the orientation $\pi/2$.
2. Compute the sequence of *event intervals*, $[\theta_i, \theta_{i+1}) \subset [-\frac{\pi}{2}, \frac{\pi}{2})$, each one defined by a pair of θ-orientations such that for any value $\theta \in [\theta_i, \theta_{i+1})$, the strip θ-kernel(P) is supported by the same pair of reflex vertices; i.e., the same lowest reflex minimum and highest reflex maximum for any $\theta \in [\theta_i, \theta_{i+1})$. See Figure 1. The strip θ-kernel(P) with θ-orientation is empty if the lowest reflex minimum is below the highest reflex maximum. In order to compute the sequence of event intervals do the following:

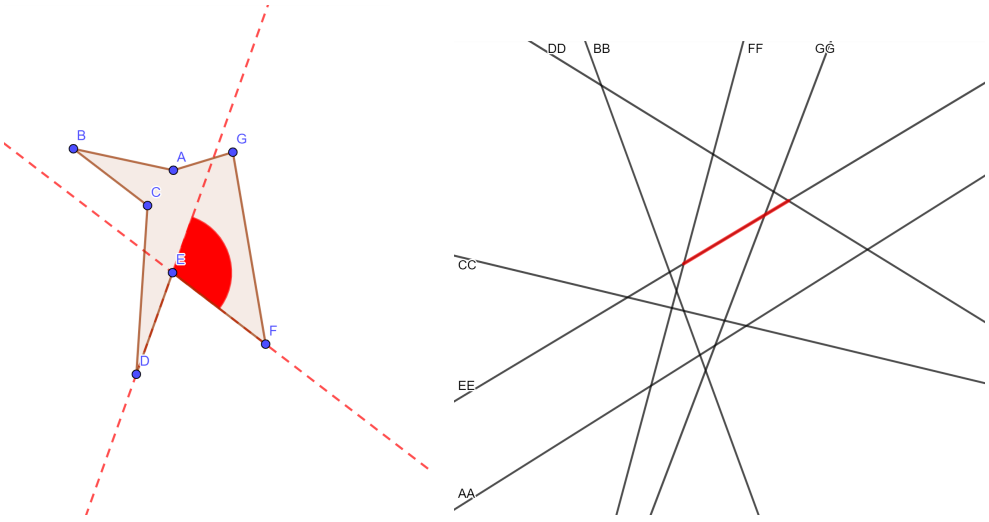


Figure 3 Dualization of vertices into lines and a reflex slope interval into a segment.

a. Dualize the vertices $p_i \in P$ into lines $D(p_i)$. On each of these lines, mark the segment corresponding to the reflex slope interval of its primal point; i.e., dualize the supporting lines on p_i with slopes in those intervals. See Figure 3.
Color the segment blue if it corresponds to a reflex minimum vertex or red if it corresponds to a reflex maximum vertex. Note that since a reflex vertex can become reflex minimum for a θ-orientation and reflex maximum for the $(\theta + \pi)$-orientation, corresponding blue and red segments can lie on the same dual line corresponding to that vertex.
It may be the case that for some θ-orientations there are no reflex minimum or no reflex maximum vertices because there are no reflex vertices at all; i.e., at those θ-orientations P has a convex chain part, and then the lowest or the highest convex vertices of $CH(P)$ play the role of the lowest reflex minimum or the highest reflex maximum for those θ-orientations. Then we mark the line segments in the dual accordingly, corresponding to those convex chains of $CH(P)$; i.e., red if there is no reflex minimum or blue if there is no reflex maximum.

b. Let $\mathcal{D}_R$ and $\mathcal{D}_B$ be the arrangements containing the blue and red segments defined above. We then do a line-sweep with a vertical line (corresponding to a value θ) from left to right such that the vertical line intersects some segments of the arrangement, which correspond to the reflex vertices in the primal. Since dualization preserves the above–below relationships between lines and/or points, the lowest blue segment and the highest red segment intersected by the vertical line in the dual correspond to the lowest reflex minimum and to the highest reflex maximum in the primal.

3. To do the line-sweep of step 2 in $O(n \log n)$ time, we compute the lower envelope of $\mathcal{D}_\mathcal{B}$, $\mathcal{L}_{\mathcal{D}_\mathcal{B}}$ and the upper envelope of $\mathcal{D}_\mathcal{R}$, $\mathcal{U}_{\mathcal{D}_\mathcal{R}}$ [3]. After merging the two envelopes in linear time, we do a line-sweep of $\mathcal{L}_{\mathcal{D}_\mathcal{B}} \cup \mathcal{U}_{\mathcal{D}_\mathcal{R}}$, obtaining the sequence of pairs of antipodal interior points for all the *event intervals* $[\theta_i, \theta_{i+1}]$ as θ varies in $[-\frac{\pi}{2}, \frac{\pi}{2})$.
In general, the lower envelope and the upper envelope of a set of n (possibly intersecting) line segments in the plane have worst-case size $O(n\alpha(n))$, where $\alpha(n)$ is the extremely-slowly–growing inverse of Ackermann's function [1], which would consequently give that the number of pairs of antipodal interior vertices is $O(n\alpha(n))$; see [3]. However, for this particular arrangement of segments, we can prove that the complexity of the lower envelope $\mathcal{L}_{\mathcal{D}_\mathcal{B}}$ and of the upper envelope $\mathcal{U}_{\mathcal{D}_\mathcal{R}}$ are in $O(n)$. For the sake of easier reading, this claim is proved as Lemma 2.1 below. Thus, merging the two envelopes has $O(n)$ complexity and we can line-sweep them in linear time.
4. Update an *event interval* $[\theta_i, \theta_{i+1}) \subseteq [-\frac{\pi}{2}, \frac{\pi}{2})$ if, for the corresponding pair, the lowest reflex minimum is above the highest reflex maximum in a θ-orientation.

▶ **Lemma 2.1.** *The complexity of the lower envelope $\mathcal{L}_{\mathcal{D}_\mathcal{B}}$ and the complexity of the upper envelope $\mathcal{U}_{\mathcal{D}_\mathcal{R}}$ are in $O(n)$.*

Proof. Clearly, the proof for the complexity of the lower envelope $\mathcal{L}_{\mathcal{D}_\mathcal{B}}$ is analogous to the proof for the complexity of the upper envelope $\mathcal{U}_{\mathcal{D}_\mathcal{R}}$. Thus, we concentrate on the former. Figure 4 shows an example where the lowest reflex minimum for $\theta_1 = 0$ is d, and increasing θ gives rise to a counterclockwise sliding rotation [7]; rotating with pivot at d, we hit b and change the pivot to it, then hit c and change the pivot to it. When θ reaches the orientation $\bar{a}$ aligned to an edge incident to a, a becomes a reflex minimum, so we slide to a, then pivot around a, until we hit c, and so on. Figure 5 shows the primal on the left, including the full sequence of pivots, and the dual on the right, including the lower envelope $\mathcal{L}_{\mathcal{D}_\mathcal{B}}$.

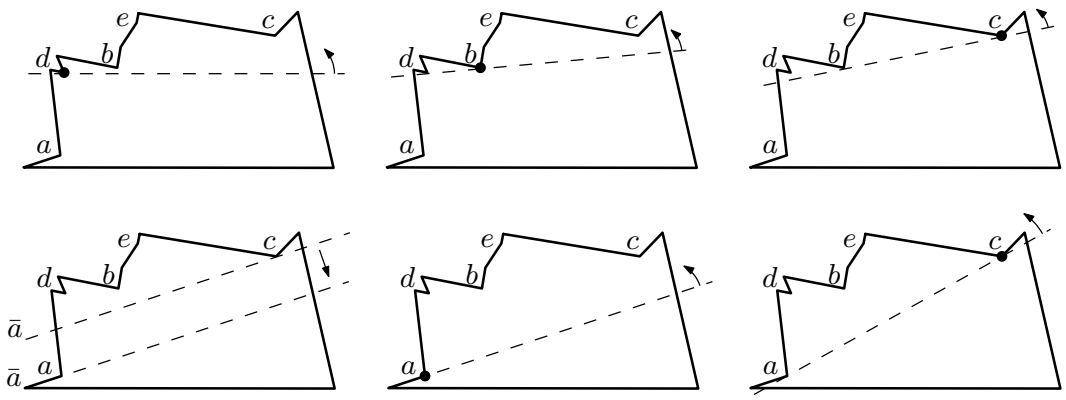


Figure 4 From left to right and from top to bottom, first steps of a sliding rotation. The pivot point at each step is marked.

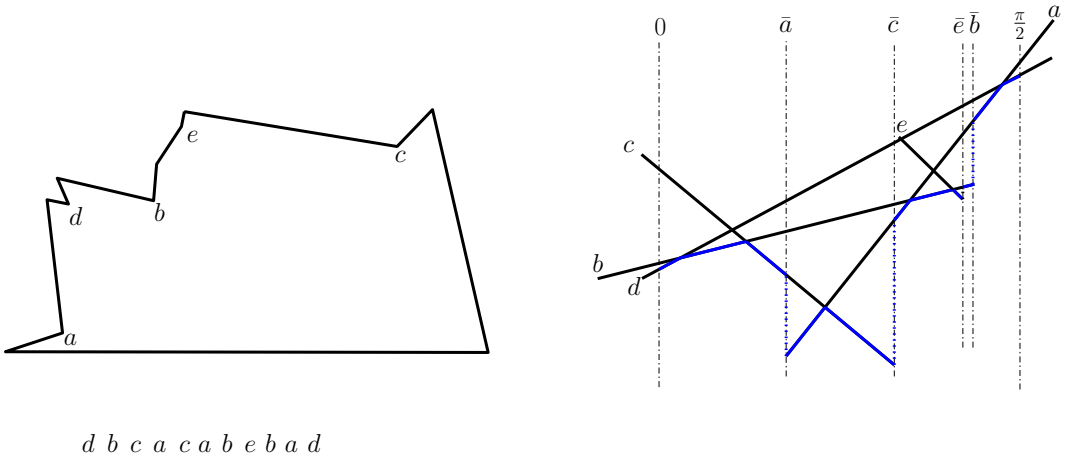


Figure 5 Left: Full sequence of pivots for the polygonal region in Figure 4 (where only the first five pivots were included). Right: Illustration of the dual and the lower envelope $\mathcal{L}_{\mathcal{D}_\mathcal{B}}$.

Next, for the rotation between 0 and $\frac{\pi}{2}$, we describe a charging scheme proving that the complexity of $\mathcal{L}_{\mathcal{D}_\mathcal{B}}$ is in $O(n)$ (an analogous argument holds for the rest of the rotation and for $\mathcal{U}_{\mathcal{D}_\mathcal{R}}$). Figure 6, left, shows in purple the *interior tangents*, when a vertex is hit by the rotation motion, together with an auxiliary dashed horizontal tangent arriving at the starting vertex d. Figure 6, right, shows the dual, the lower envelope $\mathcal{L}_{\mathcal{D}_\mathcal{B}}$, and the charging labels in blue.

1. When a vertex z is reached by a sliding motion, this is because the orientation has aligned with a side of the polygon incident to the vertex y. In this case, the vertex z receives the label $\bar{y}$ (the bar indicating "side incident to"). Note that this may happen either (i) because z was *inactive* (not reflex minimum) and it becomes *active* (reflex minimum), like the first appearance of a in Figure 6 where $y = a$ and hence the label is $\bar{a}$); or (ii) because an active vertex becomes inactive, like the second appearance of c in Figure 6, and the sliding motion hits z (in the figure, this z is the second appearance of a, which is labeled as $\bar{c}$ because in this case $y = c$).
2. When a vertex z is hit by a rotating motion, an interior tangent yz arises.
   a. If this is the first appearance of z, it receives the label z.

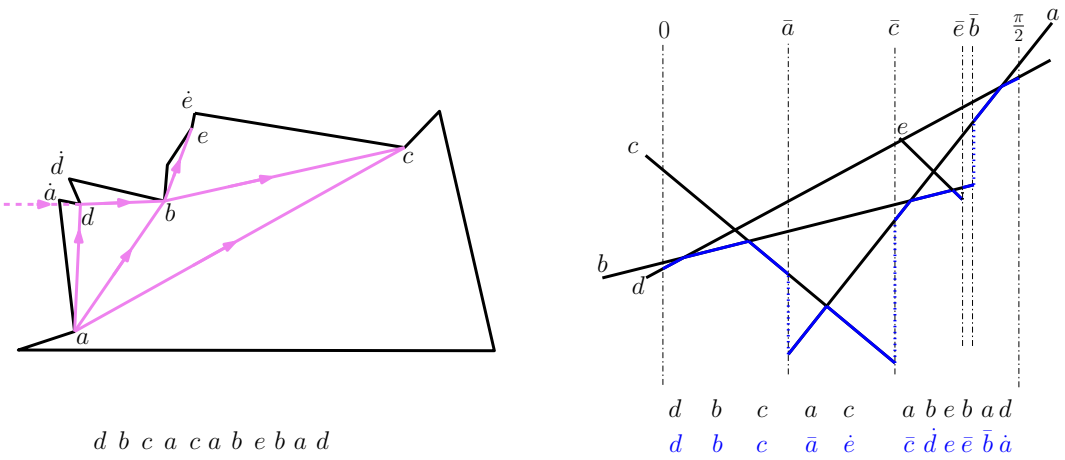


Figure 6 Left: Polygon, sequence of vertices, and interior tangents. Right: Dual and charging labels.

b. If this z has already been hit, then it is charged to the convex vertex $\dot{r}$ closest to z in CCW order that has not already been used (in the figure, the second appearance of c is charged to $\dot{e}$, with the dot indicating "convex vertex incident to").

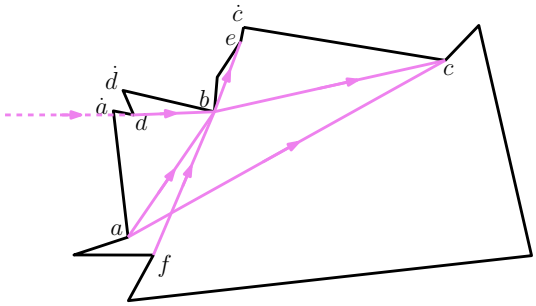
It is clear that a label $\bar{z}$ can appear at most two times and that a label z can appear at most once. In order to see that a label $\dot{r}$ always exists, observe that for z to have been hit before yz, a tangent xz with $\text{slope}(xz) < \text{slope}(yz)$ must exist. Hence, z being reflex at orientation yz and y being reflex at orientation yz imply that there is a convex vertex $\dot{r}$ above yz, where r is the closest reflex vertex to $\dot{r}$ in CCW order. For an example, take $z = b$, $y = a$, $x = d$, $r = d$ in Figure 6, where the second appearance of b happens at a tangent ab and there is a tangent db with $\text{slope}(db) < \text{slope}(ab)$, so that a convex vertex $\dot{d}$ exists above db. The same happens for $z = c$, $y = a$, $x = b$, $r = e$ and for $z = d$, $y = a$, $x = \textit{horizontal}$, and $r = a$, using the auxiliary dashed horizontal arriving at d in order to label the last d in the sequence as $\dot{a}$. (See Figure 7 for a second example.)

It just remains to show that for each y, a $\dot{y}$ is used only once. To this end, observe that a quadruple of internal tangents xz, yz, xt, and yt cannot appear: Without loss of generality, assume that the extensions of xt and yz do cross when extending from x and y as in the figure. Then, xt is not a valid interior tangent, since f is a lower reflex vertex. See Figure 8, where $z = b$, $y = f$, $x = a$, and $t = u$. ◀

By Lemma 2.1, the sizes of both the lower envelope and the upper envelope are linear, so all the steps of the algorithm above can be done within $O(n \log n)$ time and $O(n)$ space. Therefore, we get the following result.

▶ **Theorem 2.2.** *For a simple polygon P with n vertices, there are $O(n)$ angular intervals $[\theta_i, \theta_{i+1}) \subset [-\frac{\pi}{2}, \frac{\pi}{2})$ such that θ-kernel$(P) \neq \emptyset$ for all the values $\theta \in (\theta_i, \theta_{i+1})$, and the set of such intervals together with the maximum value of the width of the θ-kernel(P) can be computed and maintained in $O(n \log n)$ time and $O(n)$ space.*

Proof. For each of the $O(n)$ angular intervals, compute the maximum value of the width of the θ-kernel(P) for $\theta \in [\theta_i, \theta_{i+1})$ in constant time since we know the pair of supporting points, the range of the value θ, and that the width of the θ-kernel(P) is an unimodal function. ◀



$d\ b\ c\ a\ c\ a\ b\ f\ b\ e\ b\ f$

$d\ b\ c\ \bar{a}\ \dot{c}\ \bar{c}\ \dot{d}\ \bar{f}\ \dot{a}\ e\ \bar{e}\ \bar{b}$

Figure 7 Another example of the charging scheme.

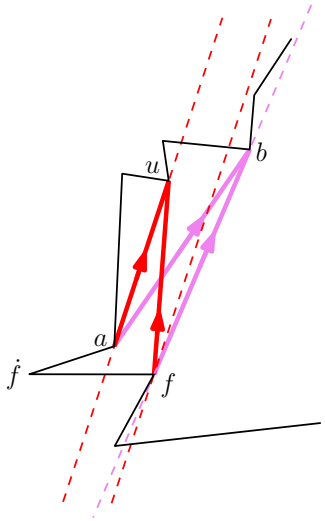


Figure 8 Illustration of why label $\dot{f}$ is not used a second time.

▶ **Corollary 2.3.** *The maximum width of the* θ-kernel(P) *simple polygon* P *with* n *vertices can be computed in* $O(n \log n)$ *time and* $O(n)$ *space.*

Acknowledgements

David Orden was supported by project MTM2017-83750-P of the Spanish Ministry of Science (AEI/FEDER, UE). Carlos Seara was supported by projects MTM2015-63791-R MINECO/FEDER and Gen. Cat. DGR 2017SGR1640. Jorge Urrutia was supported in part by SEP-CONACYT of Mexico, Proyecto 80268, and by PAPIIT IN102117 Programa de Apoyo a la Investigación e Innovación Tecnológica, UNAM. Paweł Żyliński was supported by the grant 2015/17/B/ST6/01887 (National Science Centre, Poland).

References

1 W. Ackermann. Zum Hilbertschen Aufbau der reellen Zahlen. *Mathematische Annalen*, 99, (1928), 118–133.

Canadian Conference on Computational Geometry, (2001).

2 S. Chattopadhyay and P. Das. The k-dense corridor problems. *Pattern Recognition Letters*, 11, (1990), 463–469.

3 J. Hershberger. Finding the upper envelope of n line segments in $O(n \log n)$ time. *Inf. Process. Lett.*, 33(4), (1989), 169–174.

4 M. Houle and A. Maciel. Finding the widest empty corridor through a set of points. *In G.T. Toussaint, editor, Snapshots of Computational and Discrete Geometry*, 201–213. TR SOCS-88.11, Dept. of Computer Science, McGill University, Montreal, Canada, (1988).

5 R. Janardan and F. P. Preparata. Widest-corridor problems. *Nordic Journal of Computing*, 1, (1994), 231–245.

6 D. Orden, L. Palios, C. Seara, and P. Żyliński. Generalized kernels of polygons under rotation. In *Proceedings of EuroCG 2018*, paper 74.

7 D. Orden, P. Ramos, and G. Salazar. The number of generalized balanced lines. *Discrete & Computational Geometry*, 44(4), 2010, 805–811.

8 F. P. Preparata and M. I. Shamos. *Computational Geometry: An introduction*, Springer-Verlag, (1985).

9 S. Schuierer, G. J. E. Rawlins, and D. Wood. A generalization of staircase visibility. *3rd Canadian Conference on Computational Geometry*, Vancouver, 1991, 96–99.

10 G. Toussaint. Solving geometric problems with the rotating calipers. In *Proceedings of IEEE MELECON'83*, Athens, Greece, May 1983.

On Implementing Multiplicatively Weighted Voronoi Diagrams*

Martin Held[1] and Stefan de Lorenzo[1]

1 **Universität Salzburg, FB Computerwissenschaften, Salzburg, Austria**
{held,slorenzo}@cs.sbg.ac.at

Abstract

We present a simple wavefront-like approach for computing multiplicatively weighted Voronoi diagrams of points and straight-line segments in the Euclidean plane. If the input sites may be assumed to be randomly weighted points then the use of a so-called overlay arrangement [Har-Peled&Raichel, Discrete Comput. Geom., 2015] allows to achieve an expected runtime complexity of $\mathcal{O}(n \log^4 n)$, while still maintaining the simplicity of our approach. We implemented the full algorithm for weighted points as input sites, based on CGAL. The results of an experimental evaluation of our implementation suggest $\mathcal{O}(n \log^2 n)$ as a practical bound on the runtime. Our algorithm can be extended to handle also additive weights in addition to multiplicative weights.

1 Introduction and Preliminaries

Aurenhammer and Edelsbrunner [1] present a worst-case optimal algorithm for constructing the multiplicatively weighted Voronoi diagram (MWVD) of a set of n points in $\mathcal{O}(n^2)$ time and space. Har-Peled and Raichel [2] show that a bound of $\mathcal{O}(n \log^2 n)$ holds on the expected combinatorial complexity if the weights of all points are chosen randomly. They also sketch how to compute MWVDs in expected time $\mathcal{O}(n \log^3 n)$, where linear-time triangulation and the algorithm by Aurenhammer and Edelsbrunner [1] are used as subroutines.

Let $S := \{s_1, s_2, \ldots, s_n\}$ denote a set of n distinct weighted points in $\mathbb{R}^2$ that are indexed such that $w(s_i) \leq w(s_j)$ for $1 \leq i < j \leq n$, where $w(s_i) \in \mathbb{R}^+$ is the weight associated with s_i. It is common to regard the weighted distance $d_w(p, s_i)$ from an arbitrary point p in $\mathbb{R}^2$ to s_i as the standard Euclidean distance $d(p, s_i)$ from p to s_i divided by the weight of s_i. The *(weighted) Voronoi region* $\mathcal{VR}_w(s_i, S)$ of s_i relative to S is the set of all points of the plane that are not farther to s_i than to any other site s_j in S. A connected component of a Voronoi region is called a *face*. For two distinct sites s_i and s_j of S, the *bisector* $b_{i,j}$ of s_i and s_j models the set of points of the plane that are at the same weighted distance from s_i and s_j. The MWVD $\mathcal{VD}_w(S)$ of S is the union of the boundaries of the individual Voronoi regions; see Figure 1. Following common terminology, a connected component of such a set is called a *(Voronoi) edge* of $\mathcal{VD}_w(S)$. An end-point of an edge is called a *(Voronoi) node*. It is known that the bisector between two unequally weighted sites forms a circle.

The wavefront $\mathcal{WF}(S, t)$ emanated by S at time $t \geq 0$ is the set of all points p of the plane whose minimal weighted distance from S equals t. The wavefront itself consists of several circular arcs which we call *wavefront arcs*. A common end-point of two consecutive wavefront arcs is called a *wavefront vertex*; see the blue dots in Figure 2. For $t \geq 0$, the *offset circle* $c_i(t)$ of the i-th site s_i is given by a circle centered at s_i with radius $t \cdot w(s_i)$. We specify a point of $c_i(t)$ relative to s_i by its polar angle α and its (weighted) polar radius t and denote it by $p_i(\alpha, t)$. Every pair of offset circles defines exactly two *(moving) vertices* $v^l_{i,j}$

* Work supported by Austrian Science Fund (FWF): Grant P31013-N31.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

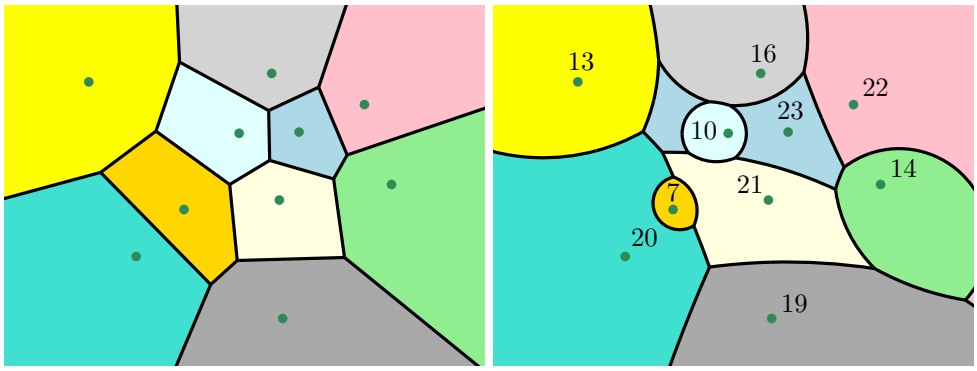


Figure 1 Left: Standard Voronoi diagram of a set of points (depicted by dark-green dots). Right: The numbers next to the points indicate their weights and the corresponding MWVD is shown.

and $v^r_{i,j}$ which can be interpreted as the traces of the intersections of the two offset circles over time. We refer to $v^l_{i,j}(t)$ as the *vertex married to* $v^r_{i,j}(t)$, and vice versa; see Figure 3.

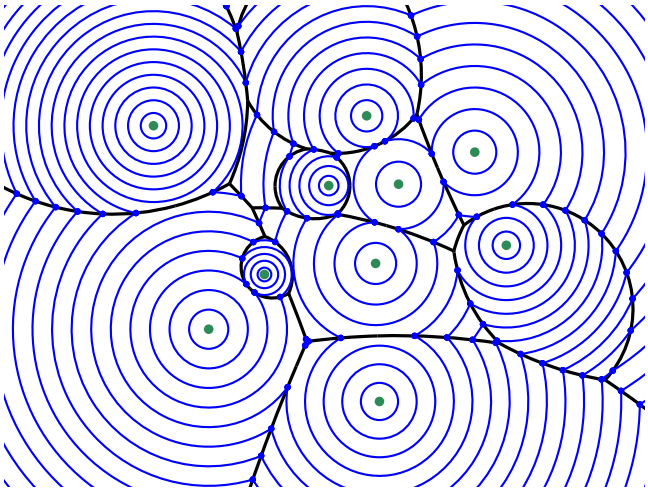


Figure 2 Wavefronts (in blue) for equally-spaced points in time for the input shown in Figure 1.

2 A Simple Event-Based Construction Scheme

In prior work [3], we introduced a wavefront-based strategy to compute $\mathcal{VD}_w(S)$ on the basis of which we made our improvements. Thus, we want to review its main ideas. For the sake of descriptional simplicity, we assume that no point in the plane has the same weighted distance to more than three elements of S. (This restriction can be waived.)

▶ **Definition 2.1** (Active point)**.** A point p on the offset circle $c_i(t)$ is called *inactive* at time t (relative to S) if there exists $j > i$, with $1 \leq i < j \leq n$, such that p lies strictly inside of $c_j(t)$. Otherwise, p is *active* (relative to S) at time t. A vertex $v_{i,j}(t)$ is an *active vertex* if it is an active point on both $c_i(t)$ and $c_j(t)$ at time t; otherwise, it is an *inactive vertex*.

▶ **Lemma 2.2.** *If $p_i(\alpha, t)$ is inactive at time t then $p_i(\alpha, t')$ will be inactive for all $t' \geq t$.*

An inactive point $p_i(\alpha, t)$ cannot be part of the wavefront $\mathcal{WF}(S, t)$. Lemma 2.2 ensures that none of its future incarnations $p_i(\alpha, t')$ can become part of the wavefront $\mathcal{WF}(S, t')$.

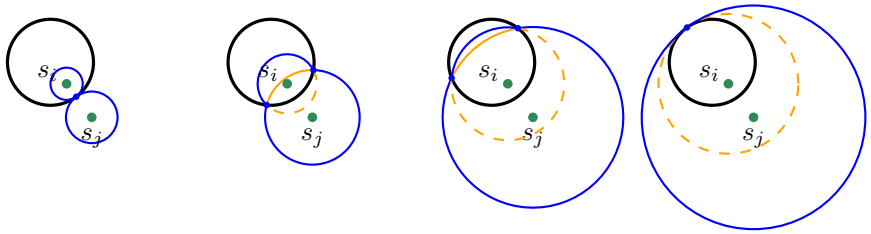


Figure 3 Two married vertices (highlighted by the blue dots) trace out the bisector (in black).

▶ **Definition 2.3** (Active arc)**.** For $1 \leq i \leq n$ and $t \geq 0$, an *active arc* of the offset circle $c_i(t)$ at time t is a maximal connected set of points on $c_i(t)$ that are active at time t. The closure of a maximal connected set of inactive points of $c_i(t)$ forms an *inactive arc* of $c_i(t)$ at time t.

Every end-point of an active arc of $c_i(t)$ is given by the intersection of $c_i(t)$ with some other offset circle $c_j(t)$, i.e., by a moving vertex $v_{i,j}(t)$. This vertex is active, too. If $i < j$ then one active arc of $c_i(t)$ and two active arcs of $c_j(t)$ are incident to $v_{i,j}(t)$. The *arc arrangement (AA)* of S at time t, $\mathcal{A}(S,t)$, is the arrangement induced by all active arcs of all offset circles of S at time t; see Figure 4. As time t increases, the offset circles expand. This causes the vertices of $\mathcal{A}(S,t)$ to move, but it will also result in topological changes of the arc arrangement. Every such topological change can be classified as one of three event types.

▶ **Definition 2.4** (Collision event)**.** Let $p_i(\alpha, t_{ij}^{min}) = p_j(\alpha+\pi, t_{ij}^{min})$ be the point of intersection of the offset circles of s_i and s_j at the collision time t_{ij}^{min}, for some fixed angle α. A *collision event* occurs between these two offset circles at time t_{ij}^{min} if the points $p_i(\alpha, t)$ and $p_j(\alpha+\pi, t)$ have been active for all times $0 \leq t \leq t_{ij}^{min}$.

▶ **Definition 2.5** (Domination event)**.** Let $p_i(\alpha, t_{ij}^{max}) = p_j(\alpha, t_{ij}^{max})$ be the point of intersection of the offset circles of s_i and s_j at the domination time t_{ij}^{max}, for some fixed angle α. A *domination event* occurs between these two offset circles at time t_{ij}^{max} if the points $p_i(\alpha, t)$ and $p_j(\alpha, t)$ have been active for all times $0 \leq t \leq t_{ij}^{max}$.

▶ **Definition 2.6** (Arc event)**.** An *arc event* e occurs at time t_e when an active arc shrinks to zero length as two unmarried vertices $v_{i,j}(t_e)$ and $v_{i,k}(t_e)$ meet in a point p_e on $c_i(t_e)$.

Domination events and arc events are easy to detect. The point and time of a collision is trivial to compute for any pair of offset circles, too. For the rest of this section we assume that all collisions among all pairs of offset circles are computed prior to the actual arc expansion. All events are stored in a priority queue $\mathcal{Q}$. If the maximum weight of all sites is associated with only one site then there will be a time t when the offset circle of this site dominates all other offset circles, i.e., when $\mathcal{WF}(S,t)$ contains only this offset circle as one active arc. Obviously, at this time no further event can occur and the arc expansion stops. If multiple sites have the same maximum weight then $\mathcal{Q}$ can only be empty once $\mathcal{WF}(S,t)$ contains only one loop of active arcs which all lie on offset circles of these sites and if all wavefront vertices move along rays to infinity.

During the arc expansion $\mathcal{O}(n^2)$ collision and domination events are computed. We know that collision events create and domination events remove active vertices (and make them inactive for good). A collapse of an entire active-arc triangle causes two vertices to become inactive. During every other arc event at least one active vertex becomes inactive, but at the same time one inactive vertex may become active again. In order to bound the number of arc events it is essential to determine how many vertices can be active and how often a vertex can undergo a *reactivation*, i.e., change its status from inactive to active.

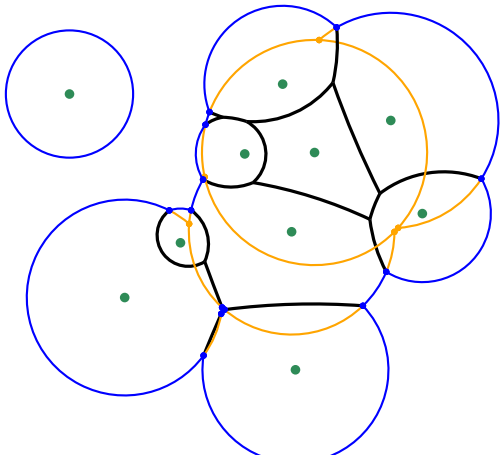


Figure 4 A snapshot of the arc expansion for the input shown in Figure 1. Active arcs that are currently not part of the wavefront are drawn in orange.

▶ **Lemma 2.7.** *Every reactivation of a moving vertex during an arc event forces another moving vertex to become inactive and remain inactive for the rest of the arc expansion.*

▶ **Lemma 2.8.** *Let h be the number of different vertices that ever were active during the arc expansion. Then $\mathcal{O}(h)$ arc events can take place during the arc expansion.*

Our naïve approach computes all potential collision events between all pairs of input sites as preprocessing. Thus, $h \in \Theta(n^2)$, and we get an overall runtime of $\mathcal{O}(n^2 \log n)$.

3 Reducing the Number of Collisions Computed

Experiments quickly indicate that the vast majority of pairwise collisions computed a priori does never end up on pairs of active arcs. Furthermore, the resulting Voronoi diagrams show a quadratic combinatorial complexity only for contrived input data.

This observation is backed by a result by Har-Peled and Raichel [2]: They show that the expected combinatorial complexity of $\mathcal{VD}_w(S)$ for a set S of n randomly weighted point sites is $\mathcal{O}(n \log^2 n)$. In order to keep our paper self-contained, we summarize their key principles.

- *Candidate set*: Consider an arbitrary (but fixed) point $q \in \mathbb{R}^2$, and let s be its nearest neighbor in S under the weighted distance. Let $s' \in S \setminus \{s\}$ be another site. Since s is the nearest neighbor of q we know that either s has a higher weight than s' or a smaller Euclidean distance to q than s'. Thus, one can define a *candidate set* for a weighted nearest neighbor of q which consists of all sites $s \in S$ such that all other sites in S either have a smaller weight or a larger Euclidean distance to q. Now assume that the sites are weighted randomly. Then Har-Peled and Raichel [2] show that this candidate set for q has a cardinality of $\mathcal{O}(\log n)$ with high probability.
- *Gerrymandering the plane*: The plane $\mathbb{R}^2$ is partitioned into a small number of regions such that the candidate set stays the same for all points within a region.
- *Randomized incremental campaigning*: Consider inserting the sites in order of decreasing weight: Then the i-th site is in the candidate set of a point $q \in \mathbb{R}^2$ if and only if it is the (unweighted) nearest neighbor of q among the first i sites. That is, if and only if q lies in the Voronoi region of the i-th site within the Voronoi diagram of the first i sites.

In more formal terms, we make use of the following results in order to determine all collision events among elements of S in near-linear expected time.

▶ **Lemma 3.1** (Har-Peled and Raichel [2]). *For all points $q \in \mathbb{R}^2$, the candidate set for q among S is of size $\mathcal{O}(\log n)$ with high probability.*

▶ **Lemma 3.2** (Har-Peled and Raichel [2]). *Let K_i denote the Voronoi cell of s_i in the unweighted Voronoi diagram of the i-th suffix $S_i := \{s_i, \ldots, s_n\}$. Let $\mathcal{OA}$ denote the arrangement formed by the overlay of the regions $K_1, \ldots, K_n$. Then, for every face f of $\mathcal{OA}$, the candidate set is the same for all points in f.*

Therefore, the overlay arrangement can be generated by incrementally constructing the unweighted Voronoi diagram of S in which the sites are inserted ordered by decreasing weights; see Figure 5. Kaplan et al. [4] prove that this overlay arrangement has an expected complexity of $\mathcal{O}(n \log n)$. Note that their result is applicable since inserting the points in sorted order of their randomly chosen weights corresponds to a randomized insertion. These results allow us to derive better complexity bounds.

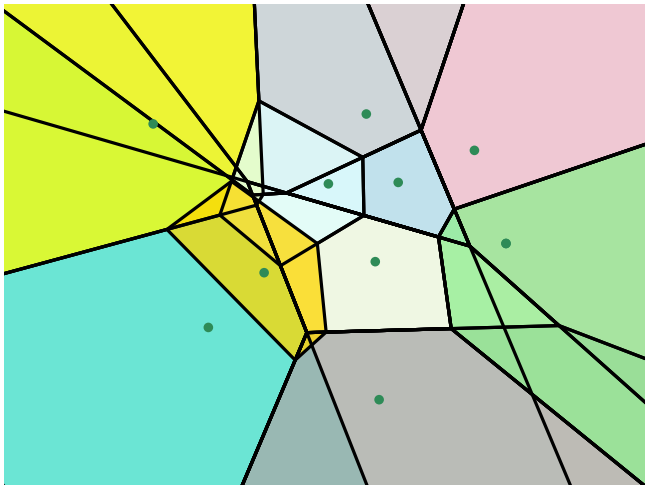


Figure 5 We insert the sites ordered by decreasing weights to generate $\mathcal{OA}$.

▶ **Lemma 3.3.** *If a collision event occurs between the offset circles of two sites $s_i, s_j \in S$ then there exists at least one candidate set which includes both s_i and s_j.*

▶ **Theorem 3.4.** *All collision events can be determined in $\mathcal{O}(n \log^3 n)$ expected time by computing the overlay arrangement $\mathcal{OA}$ of a set S of n input sites.*

Thus, the number h of vertices created during the arc expansion can be expected to be bounded by $\mathcal{O}(n \log^3 n)$. Theorem 2.8 tells us that the number of arc events is in $\mathcal{O}(h)$. Therefore, $\mathcal{O}(n \log^3 n)$ events happen in total.

▶ **Theorem 3.5.** *A wavefront-based approach allows to compute the multiplicatively weighted Voronoi diagram $\mathcal{VD}_w(S)$ of a set S of n (randomly) weighted point sites in expected $\mathcal{O}(n \log^4 n)$ time and expected $\mathcal{O}(n \log^3 n)$ space.*

4 Extensions

Consider a set S' of n disjoint weighted straight-line segments in $\mathbb{R}^2$. A wavefront propagation among weighted line segments requires us to refine our notion of "collision". We call an

intersection of two offset circles a *non-piercing collision event* if it marks the initial contact of the two offset circles. That is, it occurs when the first pair of moving vertices appear. We call an intersection of two offset circles a *piercing collision event* if it takes place when two already intersecting offset circles intersect in a third point for the first time; see Figure 6. In this case, a second pair of moving vertices appear.

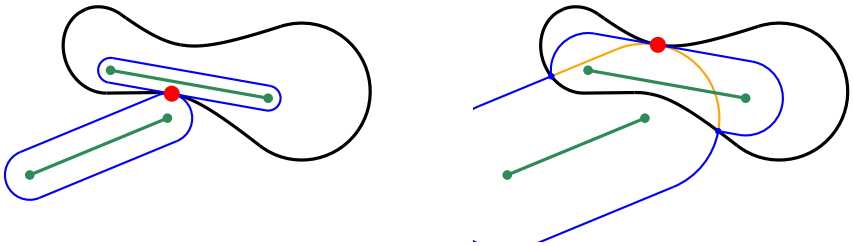


Figure 6 An example of a non-piercing (left) as well as a piercing collision event (right).

Hence, a minor modification of our event-based construction scheme is sufficient to extend it to weighted straight-line segments: We only need to check whether a piercing collision event that happens at a point p_e at time t_e currently is part of $\mathcal{WF}(S', t_e)$. In such a case the two new vertices as well as the corresponding active arc between them need to be flagged as part of $\mathcal{WF}(S', t_e)$. See Figure 7.

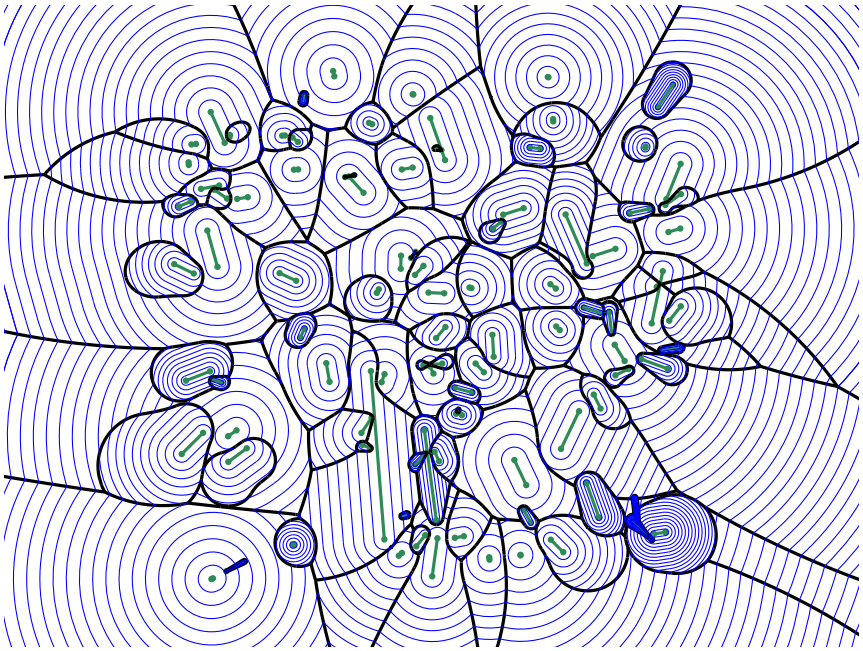
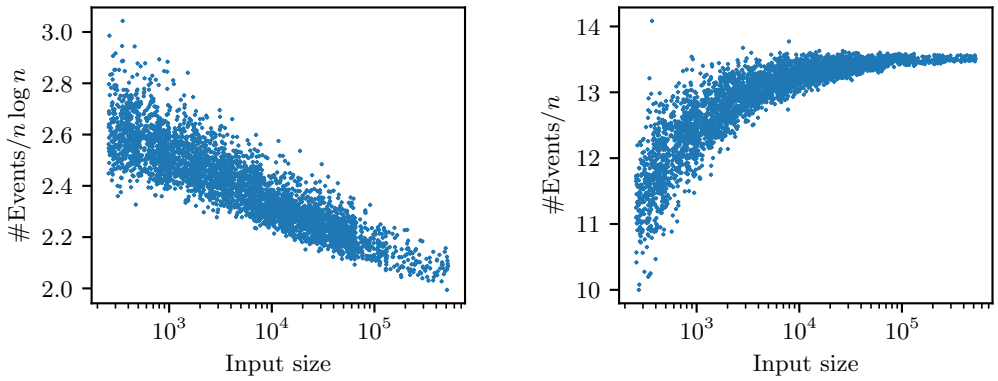


Figure 7 The MWVD of a set of weighted points and weighted straight-line segments together with a family of wavefronts for equally-spaced points in time.

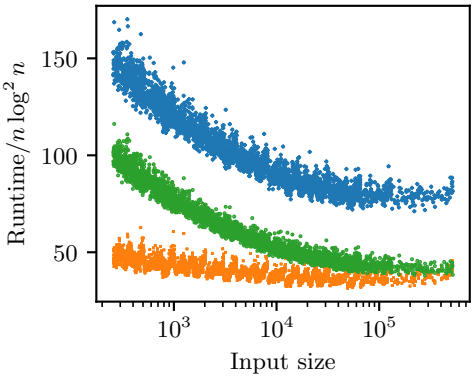
An extension to additive weights can be integrated easily into our scheme by simply giving every offset circle a head-start of $w_a(s_i)$ at time $t = 0$, where $w_a(s_i) \geq 0$ denotes the real-valued additive weight that is associated with s_i.

5 Experimental Evaluation

We implemented our full algorithm for multiplicatively weighted points as input sites[1], based on the Computational Geometry Algorithms Library (CGAL) and exact arithmetic. In particular, we use CGAL's `Arrangement_2` package for computing the overlay arrangement and its `Voronoi_diagram_2` package for computing unweighted Voronoi diagrams. The computation of the MWVD itself utilizes CGAL's `Exact_circular_kernel_2` package.



(a) The left plot shows the total number of (valid and invalid) collision events (divided by $n \log n$); the right plot shows the number of arc events (divided by n) processed during the arc expansion.



(b) The orange plot shows the runtime consumed by the computation of the overlay arrangement. The green plot shows the time which it took to process all events, and the blue plot shows the overall runtime. All runtimes were divided by $n \log^2 n$.

Figure 8 Each marker on the x-axes indicates the number n of input sites for one out of over 3800 test cases. All weights and all point coordinates were chosen randomly.

We used our implementation for an experimental evaluation and ran our code on over 3800 inputs ranging from 256 vertices to 524 288 vertices. For all inputs all weights were chosen uniformly at random, and all point coordinates were chosen according to either a

[1] Our code is publicly available on GitHub under `https://github.com/cgalab/wevo`. We do also have a prototype implementation that handles both weighted points and weighted straight-line segments. It was used to generate the diagram shown in Figure 7.

uniform or a normal distribution. All tests were carried out with CGAL 4.11 on an Intel Xeon E5-2687W v4 processor clocked at 3.0 GHz. (We carried out our tests before CGAL 5.0 was released. Sample runs obtained with CGAL 5.0 indicate that all results would be the same for CGAL 5.0.) The numbers of collision events and arc events that occurred during the arc expansion are plotted in Figure 8a. Our tests suggest that we can expect at most $c \cdot n \log n$ collision events to occur, for some small constant c. (We had $c \leq 3$ in our tests.) Furthermore, we observed at most $14n$ arc events.

In any case, the number of events is smaller than predicted by the theoretical analysis. This is also reflected by our runtime statistics: In Figure 8b the runtime of the generation of the overlay arrangement, the time that was consumed by the computation of the MWVD, and the overall runtime are plotted. Summarizing, our tests suggest an average overall runtime of $\mathcal{O}(n \log^2 n)$ if all weights are chosen randomly.

References

1 Franz Aurenhammer and Herbert Edelsbrunner. An Optimal Algorithm for Constructing the Weighted Voronoi Diagram in the Plane. *Pattern Recogn.*, 17(2):251–257, 1984. doi:10.1016/0031-3203(84)90064-5.

2 Sariel Har-Peled and Benjamin Raichel. On the Complexity of Randomly Weighted Multiplicative Voronoi Diagrams. *Discrete Comput. Geom.*, 53(3):547–568, 2015. doi:10.1007/s00454-015-9675-0.

3 Martin Held and Stefan de Lorenzo. A Wavefront-Like Strategy for Computing Multiplicatively Weighted Voronoi Diagrams. In *Proceedings of the 35th European Workshop on Computational Geometry*, Utrecht, Netherlands, 2019.

4 Haim Kaplan, Edgar Ramos, and Micha Sharir. The Overlay of Minimization Diagrams in a Randomized Incremental Construction. *Discrete Comput. Geom.*, 45(3):371–382, 2011. doi:10.1007/s00454-010-9324-6.

Sometimes Reliable Spanners of Almost Linear Size

Kevin Buchin[1], Sariel Har-Peled[2], and Dániel Oláh[1]

1 **Department of Mathematics and Computing Science, TU Eindhoven, The Netherlands**
k.a.buchin@tue.nl | d.olah@tue.nl

2 **Department of Computer Science, University of Illinois at Urbana-Champaign, USA**
sariel@illinois.edu

Abstract

Reliable spanners can withstand huge failures, even when a linear number of vertices are deleted from the network. In case of failures, some of the remaining vertices of a reliable spanner may no longer admit the spanner property, but this collateral damage is bounded by a fraction of the size of the attack. It is known that $\Omega(n \log n)$ edges are needed to achieve this strong property, where n is the number of vertices in the network, even in one dimension. Constructions of reliable geometric $(1+\varepsilon)$-spanners, for n points in $\mathbb{R}^d$, are known, where the resulting graph has $O(n \log n \log\log^6 n)$ edges.

Here, we show randomized constructions of smaller size spanners that have the desired reliability property in expectation or with good probability. The new construction is simple, and potentially practical – replacing a hierarchical usage of expanders (which renders the previous constructions impractical) by a simple skip-list like construction. This results in a 1-spanner, on the line, that has linear number of edges. Using this, we present a construction of a reliable spanner in $\mathbb{R}^d$ with $O(n \log\log^2 n \log\log\log n)$ edges.

1. Introduction

Geometric graphs are such that their vertices are points in the d-dimensional Euclidean space $\mathbb{R}^d$ and edges are straight line segments. Let $G = (P, E)$ be a geometric graph, where $P \subset \mathbb{R}^d$ is a set of n points and E is the set of edges. The shortest path distance between two points $p, q \in P$ in the graph G is denoted by $\mathsf{d}_G(p, q)$ (or just $\mathsf{d}(p, q)$). The graph G is a t-spanner for some constant $t \geq 1$, if $\mathsf{d}(p, q) \leq t \cdot \|p - q\|$ holds for all pairs of points $p, q \in P$, where $\|p - q\|$ stands for the Euclidean distance of p and q. The spanning ratio, stretch factor, or dilation of a graph G is the minimum number $t \geq 1$ for which G is a t-spanner. A path between p and q is a *t-path* if its length is at most $t \cdot \|p - q\|$.

We focus our attention to construct spanners that can survive massive failures of vertices. The most studied notion is fault tolerance [6, 7, 8], which provides a properly functioning residual graph if there are no more failures than a predefined parameter k. It is clear, that a k-fault tolerant spanner must have $\Omega(kn)$ edges to avoid small degree nodes. Therefore, fault tolerant spanners must have quadratic size to be able to survive a failure of a constant fraction of vertices. Another notion is robustness [2], which gives more flexibility by allowing the loss of some additional nodes by not guaranteeing t-paths for them. For a function $f : \mathbb{N} \to \mathbb{R}^+$ a t-spanner G is f-robust, if for any set of failed points B there is an extended set B^+ with size at most $f(|B|)$ such that the residual graph $G \setminus B$ has a t-path for any pair of points $p, q \in P \setminus B^+$. The function f controls the robustness of the graph - the slower the function grows the more robust the graph is. The benefit of robustness is that a near linear number of edges are enough to achieve it, even for the case when f is linear, there

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

are constructions with nearly $\mathcal{O}(n \log n)$ edges. For $\vartheta \in (0,1)$, a spanner that is f-robust with $f(k) = (1+\vartheta)k$ is a *ϑ-reliable* spanner [4]. This is the strongest form of robustness, since the dilation can increase for only a tiny additional fraction of points beyond t. The fraction is relative to the number of failed vertices and controlled by the parameter ϑ.

Recently, the authors [4] showed a construction of reliable 1-spanners of size $\mathcal{O}(n \log n)$ in one dimension, and of reliable $(1+\varepsilon)$-spanners of size $\mathcal{O}\big(n \log n \log \log^6 n\big)$ in higher dimensions (the constant in the $\mathcal{O}$ depends both on the dimension, ε, and the reliability parameter). An alternative construction, with slightly worse bounds, was given by Bose *et al.* [1].

Limitations of previous constructions. The construction of Buchin *et al.* [4] (and also the construction of Bose *et al.* [1]) relies on using expanders to get a monotone spanner for points on the line, and then extending it to higher dimensions. The spanner (in one dimension) has $\mathcal{O}(n \log n)$ edges. Unfortunately, even in one dimension, such a reliable spanner requires $\Omega(n \log n)$ edges, as shown by Bose *et al.* [2].

The problem. As such, the question is whether one can come up with simple and practical constructions of spanners that have linear or near linear size, while still possessing some reliability guarantee – either in expectation or with good probability.

Some definitions. Given a graph G, an *attack* $B \subseteq V(G)$ is a set of vertices that are being removed. The *damaged set* B^+, is the set of all the vertices which are no longer connected to the rest the graph, or are badly connected to the rest of the graph – that is, these vertices no longer have the desired spanning property. The *loss* caused by B, is the quantity $|B^+ \setminus B|$, where we take the minimal damaged set. Note, that B^+ is not necessarily unique. The *loss rate* of B is $\lambda(G,B) = |B^+ \setminus B| \,/\, |B|$. A graph G is *ϑ-reliable* if for any attack B, the loss rate $\lambda(G,B)$ is at most ϑ.

Randomness and obliviousness. As mentioned above, reliable spanners must have size $\Omega(n \log n)$. A natural way to get a smaller spanner, is to consider randomized constructions, and require that the reliability holds in expectation (or with good probability). Randomized constructions are (usually) still sensitive to adversarial attacks, if the adversary is allowed to pick the attack set after the construction is completed (and it is allowed to inspect it). A natural way to deal with this issue is to restrict the attacks to be *oblivious* – that is, the attack set is chosen before the graph is constructed (or without any knowledge of the edges).

In such an oblivious model, the loss rate is a random variable (for a fixed attack B). It is thus natural to construct the graph G randomly, in such a way that $\mathbb{E}[\lambda(G,B)] \leq \vartheta$, or alternatively, that the probability $\mathbb{P}[\lambda(G,B) \geq \vartheta]$ is small.

Our results. We give a randomized construction of a 1-spanner in one dimension, that is ϑ-reliable in expectation, and has size $\mathcal{O}(n)$. Formally, the construction has the property that $\mathbb{E}[\lambda(G,B)] \leq \vartheta$. This construction can also be modified so that $\lambda(G,B) \leq \vartheta$ holds with some desired probability. This is the main technical contribution of this work.

Next, following in the footsteps of the construction of reliable spanners, we use the one-dimensional construction to get $(1+\varepsilon)$-spanners that are ϑ-reliable either in expectation or with good probability. The new constructions have size roughly $O\big(n \log \log^2 n\big)$.

In this abstract, we only present the one-dimensional construction of reliable spanners in expectation. For the missing proofs and further results, we refer to the full version [3].

2. Preliminaries

▶ **Definition 2.1** (Reliable spanner)**.** Let $G = (P, E)$ be a t-spanner for some $t \geq 1$ constructed by a (possibly) randomized algorithm. Given an oblivious attack B, its ***damaged set*** B^+ is the smallest set, such that for any pair of vertices $u, v \in P \setminus B^+$, we have $\mathsf{d}_{G \setminus B}(u, v) \leq t \cdot \|u - v\|$, that is, t-paths are preserved for all pairs of points not contained in B^+. The quantity $|B^+ \setminus B|$ is the *loss* of G under the attack B. The ***loss rate*** of G is $\lambda(G, B) = |B^+ \setminus B| / |B|$. For $\vartheta \in (0, 1)$, the graph G is ***ϑ-reliable*** if $\lambda(G, B) \leq \vartheta$ holds for any attack $B \subseteq P$. Further, we say that the graph G is ***ϑ-reliable in expectation*** if $\mathbb{E}[\lambda(G, B)] \leq \vartheta$ holds for any oblivious attack $B \subseteq P$. For $\vartheta, \rho \in (0, 1)$, we say that the graph G is ***ϑ-reliable with probability*** $1 - \rho$ if $\mathbb{P}[\lambda(G, B) \leq \vartheta] \geq 1 - \rho$ holds for any oblivious attack $B \subseteq P$.

Let $[n]$ denote the *interval* $\{1, \ldots, n\}$. Similarly, for x and y, let $[x \ldots y]$ denote the interval $\{x, x + 1, \ldots, y\}$. We borrow the notion of shadow from our previous work [4]. A point p is in the α-shadow if there is a neighborhood of p, such that an α-fraction of it belongs to the attack set. One can think about the maximum α such that p is in the α-shadow of B as the *depth* of p (here, the depth is in the range $[0, 1]$). A point with depth close to one, are intuitively surrounded by failed points, and have little hope of remaining well connected. Fortunately, only a few points have depth truly close to one 1.

▶ **Definition 2.2.** Consider an arbitrary set $B \subseteq [n]$ and a parameter $\alpha \in (0, 1)$. A number i is in the ***left α-shadow*** of B, if and only if there exists an integer $j \geq i$, such that $\big|[i \ldots j] \cap B\big| \geq \alpha \big|[i \ldots j]\big|$. Similarly, i is in the ***right α-shadow*** of B, if and only if there exists an integer h, such that $h \leq i$ and $|[h \ldots i] \cap B| \geq \alpha |[h \ldots i]|$. The left and right α-shadow of B is denoted by $\mathcal{S}_{\rightarrow}(B)$ and $\mathcal{S}_{\leftarrow}(B)$, respectively. The combined shadow is denoted by $\mathcal{S}(\alpha, B) = \mathcal{S}_{\rightarrow}(B) \cup \mathcal{S}_{\leftarrow}(B)$.

▶ **Lemma 2.3** ([4])**.** *For any set $B \subseteq [n]$, and $\alpha \in (0, 1)$, we have that $|\mathcal{S}(\alpha, B)| \leq \big(1 + 2\lceil 1/\alpha \rceil\big) |B|$. Further, if $\alpha \in (2/3, 1)$, we have that $|\mathcal{S}(\alpha, B)| \leq |B| / (2\alpha - 1)$.*

▶ **Definition 2.4.** Given a graph G over $[n]$, a ***monotone path*** between $i, j \in [n]$, such that $i < j$, is a sequence of vertices $i = i_1 < i_2 < \cdots < i_k = j$, such that $i_{\ell-1} i_\ell \in E(G)$, for $\ell = 2, \ldots, k$.

A monotone path between i and j has length $|j - i|$. We use $\log x$ and $\ln x$ to denote the base 2 and natural base logarithm of x, respectively. For any set $A \subseteq P$, let $A^c = P \setminus A$ denote the complement of A. For two integer numbers $x, y > 0$, let $x_{\uparrow y} = \lceil x/y \rceil\, y$.

3. Construction of reliable spanners on the line

The input consists of a parameter $\vartheta > 0$ and the point set $P = [n] = \{1, \ldots, n\}$. The backbone of the construction is a random elimination tournament, see Figure 3.1 as an example. We assume that n is a power of 2 as otherwise one can construct the graph for the next power of two, and then throw away the unneeded vertices.

The tournament is a full binary tree, with the leafs storing the values from 1 to n, say from left to right. The value of a node is computed randomly and recursively. For a node, once the values of the nodes were computed for both children, it randomly copies the value of one of its children, with equal probability to choose either child. Let P_i be the values stored in the ith bottom level of the tree. As such, $P_0 = P$, and $P_{\log n}$ is a singleton. Each set P_i can be interpreted as an ordered set (from left to right, or equivalently, by value). Let

$$\alpha = 1 - \frac{\vartheta}{8} \qquad \text{and} \qquad \varepsilon = \frac{8(1 - \alpha)}{c \ln \vartheta^{-1}} = \frac{\vartheta}{c \ln \vartheta^{-1}}, \tag{3.1}$$

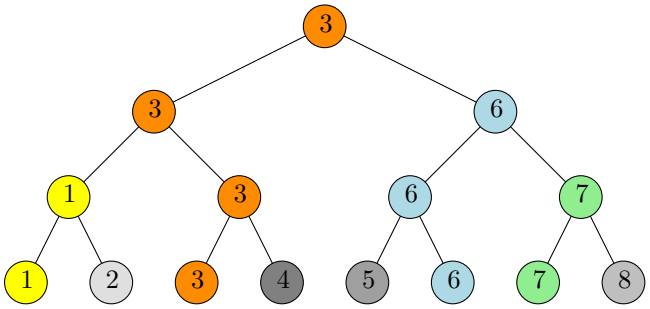


Figure 3.1 An example of a tournament tree with $n = 8$.

where $c > 1$ is a sufficiently large constant. Let M be the smallest integer for which $|P_M| \leq 2^{M/2}/\varepsilon$ holds (i.e., $M = \lceil (2/3)\log(\varepsilon n) \rceil$). For $i = 0, 1, \ldots, M$, and for all $p \in P_i$ connect p with the first

$$\ell(i) = \left\lceil \frac{2^{i/2}}{\varepsilon} \right\rceil \tag{3.2}$$

successors (and hence predecessors) of p in P_i. Let E_i be the set of all edges in level i. The graph G on P is defined as the union of all edges over all levels – that is, $E(G) = \cup_{i=0}^{M} E_i$.

4. Analysis

▶ **Lemma 4.1.** *The graph G has $\mathcal{O}\left(n\vartheta^{-1}\log\vartheta^{-1}\right)$ edges.*

Proof. The number of edges contributed by a point in P_i is at most $\ell(i)$ at level i, and $|P_i| = n/2^i$. Thus, we have

$$|E(G)| \leq \sum_{i=0}^{M} |P_i| \cdot \ell(i) \leq \sum_{i=0}^{M} \frac{n}{2^i} \cdot \left\lceil \frac{2^{i/2}}{\varepsilon} \right\rceil \leq \sum_{i=0}^{M} \frac{n}{2^i} \cdot \frac{2 \cdot 2^{i/2}}{\varepsilon} \leq \frac{n}{\varepsilon} \cdot \sum_{i=0}^{\infty} \frac{2}{2^{i/2}} = \mathcal{O}\left(\frac{n}{\varepsilon}\right). \quad \blacktriangleleft$$

Fix an attack $B \subseteq P$. The high-level idea is to show that if a point $p \in P \setminus B$ is far enough from the faulty set, then, with high probability, there exist monotone paths reaching far from p in both directions.

▶ **Definition 4.2** (Stairway)**.** Let $p \in P$ be an arbitrary point. The path $p = p_0, p_1, \ldots, p_j$ is a right (resp., left) ***stairway*** of p to level j, if

(i) $p = p_0 \leq p_1 \leq \cdots \leq p_j$ (resp., $p \geq p_1 \geq \cdots \geq p_j$),
(ii) if $p_i \neq p_{i+1}$, then $p_i p_{i+1} \in E$, for $i = 0, 1, \ldots, j-1$,
(iii) $p_i \in P_i$, for $i = 1, \ldots, j$.

Furthermore, a stairway is ***safe*** if none of its points are in the attack set B. A right (resp., left) stairway is ***usable***, if $[p_j \ldots n] \cap P_j$ (resp., $[1 \ldots p_j] \cap P_j$) forms a clique in G. Let $T \subseteq P$ denote the set of points that have a safe and usable stairway to both directions. Finally, a point p is ***bad*** if it belongs to B, or it does not have safe and usable stairways to both directions, that is, $p \in P \setminus T$.

Let $p, q \in T$ be two points such that $p < q$. Intuitively, it is clear that the right stairway of p and the left stairway of q must cross each other at some level. Combining these stairways, with some care at the point where they cross, we obtain a monotone path between p and q.

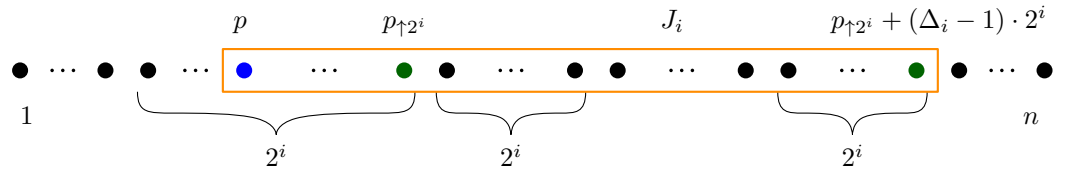


Figure 4.1 The interval $J_i = \left[p \dots p_{\uparrow 2^i} + (\Delta_i - 1) \cdot 2^i\right]$.

▶ **Lemma 4.3.** *For any two points $p, q \in T$ that are not bad, there is a monotone path connecting p and q in the residual graph $G \setminus B$.*

Let $\alpha_k = \alpha/2^k$, for $k = 0, 1, \dots, \log n$. Let $\mathcal{S}_k = \mathcal{S}(\alpha_k, B)$ be the α_k-shadow of B, for $k = 0, 1, \dots, \log n$. Observe that $\mathcal{S}_0 \subseteq \mathcal{S}_1 \subseteq \cdots \subseteq \mathcal{S}_{\log n}$, and there is an index j such that $\mathcal{S}_j = P$, if $B \neq \emptyset$. A point is classified according to when it gets "buried" in the shadow. A point p, for $k \geq 1$, is a ***k*th round** point, if $p \in \mathcal{S}_k \setminus \mathcal{S}_{k-1}$. Intuitively, a kth round point is more likely to have a safe stairway the larger the value of k is.

▶ **Lemma 4.4.** *Assume that $\vartheta \in (0, 1/2)$ and let $p \in \mathcal{S}_k \setminus \mathcal{S}_{k-1}$ be a kth round point for some $k \geq 1$. The probability that p is bad is at most $(\vartheta/2)^k/32$.*

Proof (idea). By symmetry, it is enough to consider right stairways. We define a sequence of intervals $J_1, J_2, \dots$, see Figure 4.1, such that each interval starts at p, their length increases exponentially, and $J_i \cap P_i$ contains exactly Δ_i or $\Delta_i - 1$ points. We set Δ_i to ensure that any pair of points $p_i \in J_i \cap P_i$ and $p_{i+1} \in J_{i+1} \cap P_{i+1}$ are connected. Thus, if the sets $J_i \cap P_i$, for all $i \geq 1$, contain at least one point outside of B, then we have a possible candidate for a safe and usable stairway. It is not hard to see that, for example, by choosing the leftmost available point in each set, we obtain a monotone path. ◀

We obtain bounds on the expected number of bad kth round points by using Lemma 2.3 and Lemma 4.4 to bound the number of such points and the probability of a kth round point being bad, respectively. Then, we sum up for all rounds to obtain the desired bound.

▶ **Lemma 4.5.** *Let $\vartheta \in (0, 1/2)$ and $B \subseteq P$ be an oblivious attack. Recall, that T^c is the set of bad points. Then, we have $\mathbb{E}[|T^c|] \leq (1 + \vartheta)\,|B|$.*

▶ **Theorem 4.6.** *Let $\vartheta \in (0, 1/2)$ and $P = [n]$ be fixed. The graph G, constructed in Section 3, has $\mathcal{O}\big(n\vartheta^{-1} \log \vartheta^{-1}\big)$ edges, and it is a ϑ-reliable 1-spanner of P in expectation. Formally, for any oblivious attack B, we have $\mathbb{E}[\lambda(G, B)] \leq \vartheta$.*

Proof. By Lemma 4.1, the size of G is $\mathcal{O}\big(n\vartheta^{-1} \log \vartheta^{-1}\big)$. Let $B \subseteq P$ be an oblivious attack and consider the bad set $P \setminus T$. By Lemma 4.3, for any two points outside the bad set, there is a monotone path connecting them. Further, by Lemma 4.5, we have $\mathbb{E}[|P \setminus T|] \leq (1 + \vartheta)\,|B|$ for any oblivious attack. Thus, we obtain $\mathbb{E}[\lambda(G, B)] \leq \mathbb{E}[|T^c \setminus B| \,/\, |B|] \leq \vartheta$. ◀

Using Theorem 4.6 and a result of Chan *et al.* [5] on orderings of a set of points in $\mathbb{R}^d$, we can construct spanners for higher dimensional point sets that are reliable in expectation.

▶ **Theorem 4.7.** *Let $\vartheta, \varepsilon \in (0, 1)$ be fixed and $P \subseteq \mathbb{R}^d$ be a set of n points. We can construct a $(1 + \varepsilon)$-spanner of P that is ϑ-reliable in expectation and has size $\mathcal{O}\big(n \log \log^2 n \log \log \log n\big)$.*

References

1 P. Bose, P. Carmi, V. Dujmović, and P. Morin. Near-optimal O(k)-robust geometric spanners. *CoRR*, abs/1812.09913, 2018. URL: http://arxiv.org/abs/1812.09913, arXiv:1812.09913.

2 P. Bose, V. Dujmović, P. Morin, and M. Smid. Robust geometric spanners. *SIAM Journal on Computing*, 42(4):1720–1736, 2013. URL: https://doi.org/10.1137/120874473, doi:10.1137/120874473.

3 K. Buchin, S. Har-Peled, and D. Oláh. Sometimes reliable spanners of almost linear size, 2019. arXiv:1912.01547.

4 K. Buchin, S. Har-Peled, and D. Oláh. A spanner for the day after. In *Proc. 35th Int. Annu. Sympos. Comput. Geom.* (SoCG), pages 19:1–19:15, 2019. URL: https://doi.org/10.4230/LIPIcs.SoCG.2019.19, doi:10.4230/LIPIcs.SoCG.2019.19.

5 T. M. Chan, S. Har-Peled, and M. Jones. On Locality-Sensitive Orderings and Their Applications. In *Proc. 34th Int. Annu. Sympos. Comput. Geom.* (SoCG), pages 21:1–21:17, 2018. URL: http://drops.dagstuhl.de/opus/volltexte/2018/10114, doi:10.4230/LIPIcs.ITCS.2019.21.

6 C. Levcopoulos, G. Narasimhan, and M. Smid. Efficient algorithms for constructing fault-tolerant geometric spanners. In *Proc. 30th Annu. ACM Sympos. Theory Comput.* (STOC), pages 186–195, 1998. URL: http://doi.acm.org/10.1145/276698.276734, doi:10.1145/276698.276734.

7 C. Levcopoulos, G. Narasimhan, and M. Smid. Improved algorithms for constructing fault-tolerant spanners. *Algorithmica*, 32(1):144–156, 2002. URL: https://doi.org/10.1007/s00453-001-0075-x, doi:10.1007/s00453-001-0075-x.

8 T. Lukovszki. New results of fault tolerant geometric spanners. In *Proc. 6th Workshop Algorithms Data Struct.* (WADS), pages 193–204, 1999. URL: https://doi.org/10.1007/3-540-48447-7_20, doi:10.1007/3-540-48447-7_20.

A polynomial-time partitioning algorithm for weighted cactus graphs

Maike Buchin and Leonie Selbach

Ruhr-University Bochum
maike.buchin@rub.de, leonie.selbach@rub.de

Abstract

Some graph partitioning problems are known to be NP-hard for certain graph classes and polynomially solvable for others. We study the problem of partitioning the vertex set of a weighted cactus graph into clusters with bounded weights and present a polynomial-time algorithm that solves the problem with the optimal as well as some fixed number of clusters.

1 Introduction

Let $G = (V, E)$ be a cactus graph, that is a graph where every two simple cycles have at most one vertex in common. Every vertex v in G is assigned a non-negative integer weight $w(v)$. A *(vertex) partition* of G is a partition of the vertex set V into pairwise disjoint clusters such that the induced subgraphs are connected. Given two non-negative integers l and u (with $l \leq u$), we call a partition (l, u)-*partition* if the weight of each cluster is at least l and at most u. We consider different variants of this problem: For the *minimum/maximum (l, u)-partition problem* we want to find a (l, u)-partition of a graph with the minimal resp. maximal number of clusters. And for the *p-(l, u)-partition problem* the goal is to find a partition with exactly p clusters (see Fig. 1).

Dyer and Frieze studied the complexity of different graph partitioning problems [3]. They showed that partitioning a general graph into connected components (or trees) of a given size is NP-complete, even for planar bipartite graphs – but polynomially solvable for both series-parallel graphs and trees. Cordone and Maffioli consider partitioning an edge-weighted graph into trees with different optimization goals [2]. They proved that this problem is NP-hard for all cases if the graph has additional vertex weights and a weight constraint – as a lower and upper weight bound for each tree – is added. The related tree-equipartition problem, where the partition should consist of clusters with weight as equal as possible, proved to be NP-complete for trees as well as 2-spiders, but can be solved in polynomial time for other graph classes, such as stars, worms and caterpillars [6]. Ito et al. showed that the (l, u)-partition problems are NP-hard for series-parallel graphs, but pseudo-polynomial algorithms exist [5]. Ito et al. proved that the decision variant of those problems can be solved in polynomial time if the graph is a weighted tree [4]. In this paper we present a (l, u)-partitioning algorithm for weighted cactus graphs whose runtime can be reduced to solve the given problems in polynomial time. Because of length constraints we excluded the proofs, these can be found in the full version of this paper [1].

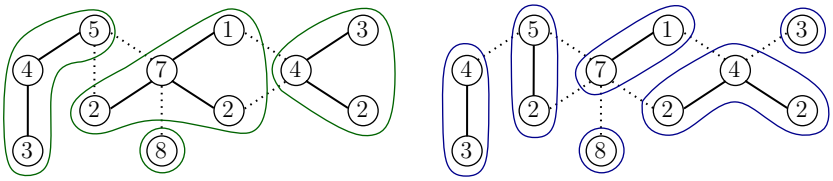


Figure 1 Left: Minimum $(3, 12)$-partition. Right: 6-$(3, 12)$-partition.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

2 Preliminaries

Let $G = (V, E, w)$ be a weighted cactus graph. We want to find a partition of G into clusters V_i such that the weight of each cluster $w(V_i) = \sum_{v \in V_i} w(v)$ lies between the two given bounds: $l \leq w(V_i) \leq u$. For our algorithm we represent the graph G as a tree T_G using depth first search (DFS) on an arbitrary vertex. This vertex becomes the root of the tree (see. Fig. 2). Every vertex of G is a node in T_G and every cycle degenerates to a path. We store a cycle by its path $\langle x, y \rangle$ in T_G and define x as the *start node* and y as the *end node* of this cycle. We denote the cycle by $C(x, y)$. If two cycles are adjacent, their common node is part of both cycles. When building the tree T_G, this node will become the start node of at least one of them.

▶ **Lemma 2.1.** *Let v be a node in T_G. If $v \in C(x, y)$ and $v \in C(x', y')$ for $(x', y') \neq (x, y)$, then v equals either x or x' (or both).*

▶ **Corollary 2.2.** *For each node v there exists at most one cycle in T_G which contains v but where v is not the start node.*

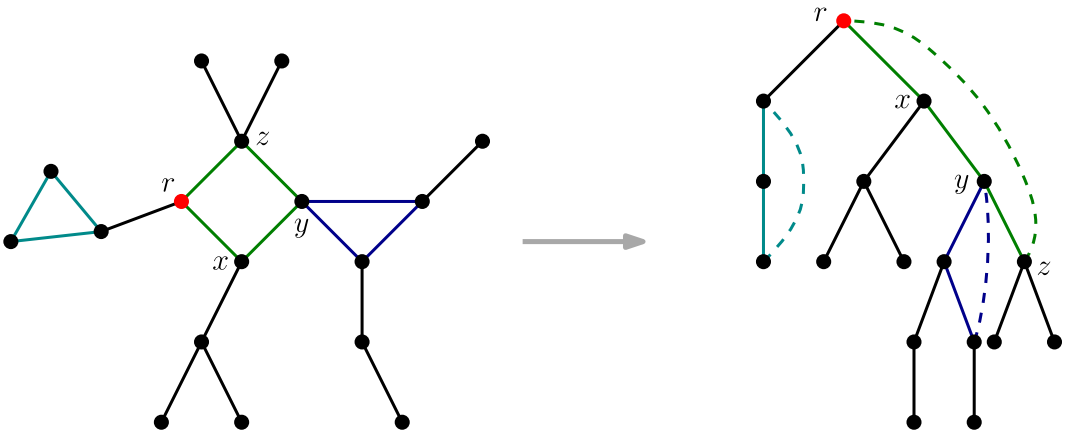


Figure 2 Turning a cactus graph G into a tree T_G by starting the DFS with the red vertex r. The cycle containing the node r is represented as a path $\langle r, x, y, z \rangle = C(r, z)$ with start node r and end node z.

▶ Remark. Let v be a node with s children $v_1, v_2, \ldots, v_s$ and let v be part of a cycle $C(x, y)$ with $v \neq x$. Because of Cor. 2.2 there is at most one such cycle. If $v \neq y$, there exists a child v' of v with $v' \in C(x, y)$ as well. W.l.o.g. we sort the children of v such that $v' = v_s$. This allows easier definitions for the algorithm since these nodes have to be treated differently.

Let us define some further notations. Let T_v be the subtree rooted in the node v. We have $T = T_r$ for r being the root of T. We define T_v^i as the subtree with root v and its first i children (see Fig. 3). If v has s children, we have $T_v^s = T_v$. Consider a partition P of the graph. Let $|P|$ be the size of the partition, meaning the number of clusters in P. For a node v we denote the cluster of P that contains v by C_v.

Let P be a (l, u)-partition of a general tree. If the lower weight bound l equals zero, the partition P induces a feasible $(0, u)$-partition of any subtree T_v. This is not necessarily the case for a general (l, u)-partition since the weight of the cluster containing the node v might not exceed the lower weight bound l. We call such a partition of a subtree T_v– where $w(C_v) \leq u$ and $l \leq w(C') \leq u$ for every cluster $C' \neq C_v$ – an *extendable* (l, u)-partition.

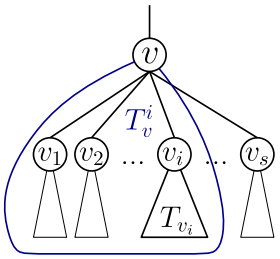


Figure 3 Definition of a subtree T_v^i of a node v.

By adding further nodes to the cluster C_v we can extend such a partition to be a feasible (l, u)-partition. For a subtree T_v (or resp. T_v^i) we define a set $S(T_v)$ as follows:

$$S(T_v) = \{(x, k) \mid \exists \text{ extendable } (l, u)\text{-partition } P \text{ of } T_v \text{ such that } |P| = k \ \wedge \ w(C_v) = x\}$$

Thus, a tuple (x, k) corresponds to a possible partition of T_v, where x is the weight of the cluster containing the node v and k is the number of clusters.

▶ **Lemma 2.3.** *A rooted tree T has a p-(l, u)-partition if and only if there exists a tuple $(x, p) \in S(T)$ such that $l \leq x \leq u$.*

Ito et al. described a method for solving the partition problems for weighted trees by computing the sets S for every subtree T_v^i bottom-up. With an adjustment they can decide the partition problem in time $\mathcal{O}(p^4 n)$. Cactus graphs, however, present the challenge of having cycles and therefore considerably more partition possibilities. We show how to extend their algorithm to cactus graphs while retaining polynomial time.

3 Partitioning Algorithms

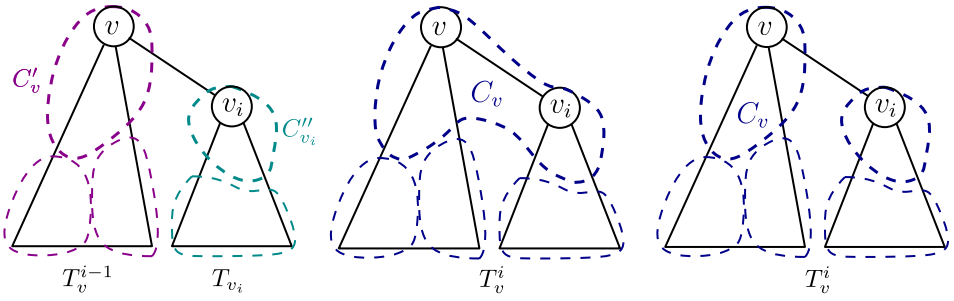
First, let us consider the decision of the p-(l, u)-partition problem, that is deciding if there exists a (l, u)-partition of a given cactus graph G into p clusters. Using the tree representation of G described in Sec. 2, we follow the general idea of Ito et al. but include a procedure that deals efficiently with cycles. Similary, we present a general algorithm which we adjust to solve the problems in polynomial time.

For each subtree T_v^i we compute all extendable (l, u)-partitions with at most p clusters. This can be done by combining the computed partitions of T_v^{i-1} and T_{v_i} as follows. First, let us consider the case that v and v_i are not part of a cycle. For each combination of partitions for the given subtrees we have two options: Either we merge the clusters containing v resp. v_i or we do not (see Fig. 4). Obviously, we keep only the generated partitions that are extendable (l, u)-partitions with size less than p. We have to make sure that in option 1 (Fig. 4b) the weight of the new cluster C_v does not exceed the upper bound u. In option 2 (Fig. 4c) P'' has to be a feasible (l, u)-partition – not just an extendable one. Therefore, we have to make sure that the weight of the cluster C''_{v_i} fulfills the lower weight constraint.

We compute those partitions using the sets S as defined in the previous section. We use a specific set-operation denoted by $\oplus$. For two sets $A = S(T_v^{i-1})$ and $B = S(T_{v_i})$ we define:

$$\begin{aligned} A \oplus B = \{&(x_1, k_1 + k_2) \mid l \leq x_2, k_1 + k_2 \leq p, (x_1, k_1) \in A, (x_2, k_2) \in B\} \\ \cup \{&(x_1 + x_2, k_1 + k_2 - 1) \mid x_1 + x_2 \leq u, k_1 + k_2 - 1 \leq p, (x_1, k_1) \in A, (x_2, k_2) \in B\}. \end{aligned} \tag{1}$$

The second line of this definition corresponds to all feasible partitions we obtain by merging the clusters and the first line corresponds to those we get without merging. We assume



(a) Partitions P' and P''. **(b)** Option 1: $v_i \in C_v$. **(c)** Option 2: $v_i \notin C_v$.

Figure 4 Combining the two partitions P' of T_v^{i-1} and P'' of T_{v_i} into a partition P of T_v^i.

that $w(v) \leq u$ for all nodes v – otherwise a (l, u)-partition of G would not exist. We set $S(T_v^0) = \{(w(v), 1)\}$ and compute $S(T_v^i) = S(T_v^{i-1}) \oplus S(T_{v_i})$ iteratively for all $1 \leq i \leq c_v$ where c_v is the number of children of the node v.

Now, let us consider cycles. Deleting one edge of a cycle leads to a path which we can partition. For a cycle of length m there are m different paths. If we take the union over all partitions of these paths, we obtain all possible partitions of the cycle. This remains true even if we declare one node of the cycle as the "root" and consider the paths as trees (see. Fig. 5). We call these the different *configurations* of a cycle which we number from 1 to m. To find all partitions the m-th configuration is not necessary.

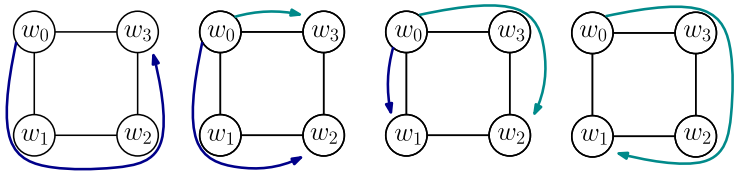


Figure 5 Configurations in a cycle of length four.

Let C be a cycle in G of length m and $C(w_0, w_{m-1}) = \langle w_0, w_1, \ldots, w_{m-1} \rangle$ be the corresponding cycle in T_G. Let each node w_i have c_i children and let w_1 be the k-th child of w_0. Remember that for all other nodes w_{i+1} is always the c_i-th, i.e. the last, child of w_i (Rem. 2). Similarly, we consider different configurations of a cycle in our tree representation (see. Fig 6). The first one is the original subtree as in T_G. For the second configuration we introduce the node w_{m-1} with its corresponding subtree as an additional child of w_0 and remove it from w_{m-2}. We continue this procedure of removing and adding nodes and their corresponding subtrees. Thus, in configuration j the node w_{m-j+1} is no longer the child of w_{m-j} but of w_{m-j+2}.

▶ **Lemma 3.1.** *Let C be a cycle in G and $C(w_0, w_{m-1})$ the corresponding cycle in T_G. A partition P of C can be found in one of the $m-1$ configurations of $C(w_0, w_{m-1})$.*

Let us transfer this to the sets $S(T_v^i)$ as computed by the algorithm. We introduce additional sets $S_j(T_{w_i})$ for each node w that is part of a cycle. Each set $S_j(T_w)$ contains tuples (x, k) corresponding to an extendable (l, u)-partition of T_w in its configuration j. As before, such a tuple corresponds to a partition of size k where the cluster containing the node w has weight x. We can define the different computations for $S(T_{w_i})$ using the $\oplus$-operation as above. First, let us consider the case where $i = m - 1$. Until the third configuration its subtree (and

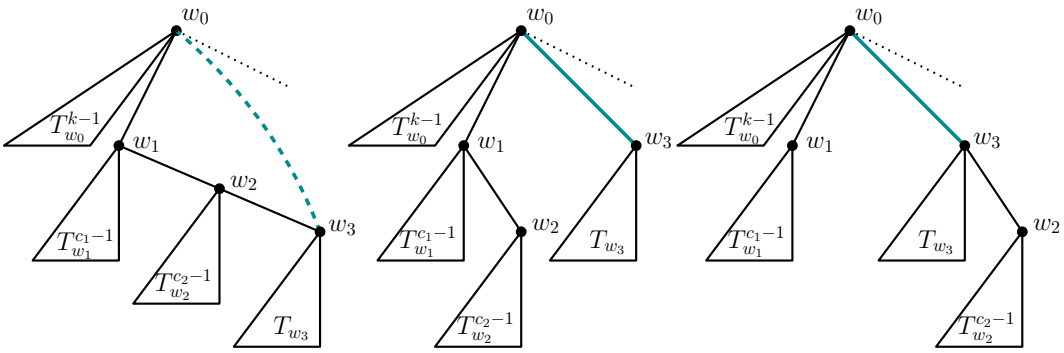


Figure 6 The different considered configurations for a cycle of length four in the subtree of w_0.

therefore the computation) stays the same. Then, the node obtains a additional child w_{m-2} whose subtree (and hence partition set) changes with each configuration.

$$S_j(T_{w_{m-1}}) = \begin{cases} S(T_{w_{m-1}}) \oplus S_j(T_{w_{m-2}}) & \text{for } j > 2, \\ S(T_{w_{m-1}}) & \text{otherwise.} \end{cases} \tag{2}$$

When $0 < i < m-1$, that is w_i is neither the start nor end node of the given cycle, we have to consider three cases: In configuration $m-i$ and $m-i+1$ the subtree T_{w_i} is equal to $T_{w_i}^{c_i-1}$ because the last (c_i-th) child is removed. If $j < m-i$, w_i still has w_{i+1} as a child but with a different subtree. If $j > m-i+1$, w_i does not have w_{i+1} as its last child but w_{i-1} instead. This can be formalized in the following way:

$$S_j(T_{w_i}) = S_j(T_{w_i}^{c_i}) = \begin{cases} S(T_{w_i}^{c_i-1}) \oplus S_j(T_{w_{i+1}}) & \text{for } j < m-i, \\ S(T_{w_i}^{c_i-1}) \oplus S_j(T_{w_{i-1}}) & \text{for } j > m-i+1, \\ S(T_{w_i}^{c_i-1}) & \text{otherwise.} \end{cases} \tag{3}$$

Now let us consider the start node w_0 of the cycle. For the first configuration we have to combine the partitions of the subtree $T_{w_0}^{k-1}$ with the partitions of the subtree rooting in w_1. In case $j > 1$ the subtree of w_1 has changed and we treat w_{m-1} as an additional child of w_0.

$$S_j(T_{w_0}^k) = \begin{cases} S(T_{w_0}^{k-1}) \oplus S_j(T_{w_1}) & \text{for } j = 1, \\ \big(S(T_{w_0}^{k-1}) \oplus S_j(T_{w_1})\big) \oplus S_j(T_{w_{m-1}}) & \text{otherwise.} \end{cases} \tag{4}$$

To compute the set $S(T_{w_0}^k)$ we take the union over all configurations: $S(T_{w_0}^k) = \bigcup_{j=1}^{m-1} S_j(T_{w_0}^k)$. Note that the computations change only for nodes that are part of the cycle. In each configuration we delete only one edge and include a new one. This has no effect on the remaining subtrees of each w_i, i.e. $T_{w_i}^{c_i-1}$, and neither on the computations for those subtrees. For the algorithm we use the bottom-up approach, set $S(T_v^0)$ as above and compute $S(T_v^i)$ for every $1 \le i \le c_v$. If there is a cycle $C(v,w)$ with v as the start node and its i-th child $v_i \in C(v,w)$, we use (4) with $v = w_0$, $v_i = w_1$ and $w = w_{m-1}$ to compute $S(T_v^i)$, using (3) resp. (2) recursively. In the case that either v is not the start node of a cycle or v_i is not part of it, we can compute $S(T_v^i) = S(T_v^{i-1}) \oplus S(T_{v_i})$ as described before.

▶ **Theorem 3.2.** *Given a weighted cactus graph G, a positive integer p and two non-negative integers l and u (with $l \le u$). The p-(l,u)-partition problem can be decided in time $\mathcal{O}(u^2p^2n^2)$ using Algorithm 1.*

Proof sketch. Every set S resp S_j has size $\mathcal{O}(up)$ and therefore every $\oplus$-operation takes $\mathcal{O}(u^2p^2)$ time. For each node v the set $S(T_v)$ (resp. $S_j(T_v)$) contributes to exactly one $\oplus$-operation. Since the number j is in $\mathcal{O}(n)$, we have $\mathcal{O}(n^2)$ $\oplus$-operations and hence an overall runtime of $\mathcal{O}(u^2p^2n^2)$. ◀

Algorithm 1: Algorithm for deciding the p-(l,u)-partition problem for a cactus graph.

```
forall v ∈ V bottom up do
    S(T_v^0) = {(w(v), 1)}
    for 1 ≤ i ≤ c_v do
        if there is a cycle C(v,w) with v_i ∈ C(v,w) then
            m = length(C(v,w))
            for 1 ≤ j ≤ m − 1 do
                Compute S_j(T_v^i) using (4) (with v = w_0, v_1 = w_1, w = w_{m−1}) and
                 recursively (2) and (3)
            S(T_v^i) = ⋃_{j=1}^{m−1} S_j(T_v^i)
        else
            S(T_v^i) = S(T_v^{i−1}) ⊕ S(T_{v_i})
if (x,k) ∈ S(T_r) such that l ≤ x ≤ u and k = p then return yes
```

▶ Remark. If we store for each tuple in the partition sets how it was computed – meaning if we did or did not merge and therefore did or did not delete an edge – we can use backtracking to compute the actual p-(l,u)-partition of the graph (assuming there is one).

Polynomial-time Algorithm

Analogously to Ito et al. we can use intervals to reduce the size of computed sets $S(T_v)$. The idea is the following: Instead of storing partitions as tuples (x,k), we store them as $([a,b],k)$. Each interval $[a,b]$ is a maximal d-consecutive subset of weights, where $d = u - l$ is the difference between the upper and lower weight bound. If we adjust the $\oplus$-operation and Alg. 1 to compute interval sets $I(T_v^i)$ (resp. $I_j(T_v^i)$) instead of $S(T_v^i)$ (resp. $S_j(T_v^i)$), we obtain an algorithm with the same number of $\oplus$-operations as before. Since the size of a set $I(T_v^i)$ is $\mathcal{O}(p^2)$ (instead of $\mathcal{O}(up)$), the overall runtime for solving the p-(l,u)-partition problem for cactus graphs is reduced to $\mathcal{O}(p^4n^2)$. Ito et al.'s work is missing a convincing polynomial-time method for the computation of a partition. See the full version of our paper for details about the time reduction and a method for the computation which has polynomial time and space and is applicable for both trees and cactus graphs [1].

Minimum and Maximum Partition Problem

We showed that for a given number p we can partition a cactus graph into p clusters such that the weight of each cluster lies between the lower bound l and upper bound u. Now we consider the problem of finding a (l,u)-partition with the minimal resp. maximal number of clusters. Ito et al. showed that the minimal resp. maximal number of clusters can be found in $\mathcal{O}(n^5)$ for trees. Similar reasoning and our computation method leads us to the following result for cactus graphs.

▶ **Theorem 3.3.** *Given a weighted cactus graph G and l and u as above. The minimum and maximum (l, u)-partition problem can be solved in time $\mathcal{O}(n^6)$.*

References

1 Maike Buchin and Leonie Selbach. A polynomial-time partitioning algorithm for weighted cactus graphs, 2020. arXiv:2001.00204.

2 Roberto Cordone and Francesco Maffioli. On the complexity of graph tree partition problems. *Discrete Applied Mathematics*, 134:51–65, 01 2004. doi:10.1016/S0166-218X(03)00340-8.

3 M.E. Dyer and A.M. Frieze. On the complexity of partitioning graphs into connected subgraphs. *Discrete Applied Mathematics*, 10(2):139 – 153, 1985. doi:https://doi.org/10.1016/0166-218X(85)90008-3.

4 Takehiro Ito, Takao Nishizeki, Michael Schröder, Takeaki Uno, and Xiao Zhou. Partitioning a weighted tree into subtrees with weights in a given range. *Algorithmica*, 62(3-4):823–841, April 2012. doi:10.1007/s00453-010-9485-y.

5 Takehiro Ito, Xiao Zhou, and Takao Nishizeki. Partitioning a graph of bounded tree-width to connected subgraphs of almost uniform size. *Journal of Discrete Algorithms*, 4(1):142 – 154, 2006. doi:10.1016/j.jda.2005.01.005.

6 Caterina De Simone, Mario Lucertini, Stefano Pallottino, and Bruno Simeone. Fair dissections of spiders, worms, and caterpillars. *Networks*, 20(3):323–344, 1990. doi:10.1002/net.3230200305.

The topological correctness of the PL-approximation of isomanifolds

Jean-Daniel Boissonnat[1] and Mathijs Wintraecken[2]

1 **Université Côte d'Azur, INRIA**
Sophia-Antipolis, France
jean-daniel.boissonnat@inria.fr

2 **IST Austria**
Klosterneuburg, Austria
m.h.m.j.wintraecken@gmail.com

Abstract

Isomanifolds are the generalization of isosurfaces to arbitrary dimension and codimension, i.e. manifolds defined as the zero set of some multivariate multivalued function $f : \mathbb{R}^d \to \mathbb{R}^{d-n}$. A natural (and efficient) way to approximate an isomanifold is to consider its Piecewise-Linear (PL) approximation based on a triangulation $\mathcal{T}$ of the ambient space $\mathbb{R}^d$. In this paper, we give conditions under which the PL-approximation of an isomanifold is topologically equivalent to the isomanifold. The conditions can always be met by taking a sufficiently fine triangulation $\mathcal{T}$.

1 Introduction

Isosurfacing especially in low dimensions is used in many fields of application, such as CT scans in medicine, biochemistry, biomedicine, deformable modeling, digital sculpting, environmental science, and mechanics and dynamics, see [29] and the references mentioned there. The marching cube approach [26] being the most popular approach taken. However the result of the marching cube algorithm is not necessarily topologically correct.

Some results on provable correctness were achieved within the computational geometry community [7, 30] in three dimensions. In case the isosurfacing is based on simplices instead of cubes, such as in the marching tetrahedra algorithm [19], some bounds can be given [1, 2], on for example the one-sided Hausdorff distance. In general homeomorphism proofs in higher dimensions rely on some perturbation scheme to prove that a triangulation is correct [34, 5, 8, 9, 12]. This is a major difference with one and two dimensional surfaces where no such requirements exist [21], [31, Section 10.2]. In this paper we shall see that no perturbations are necessary for isomanifolds as well.

The techniques used here are also different from many of the standard tools. Manifold triangulation/reconstruction algorithms use often Delaunay triangulations [32, 18, 14] and use the closed ball property [23], see for example [3, 13]. Others use Whitney's lemma [10] or are based on collapses [4]. While the current paper mainly relies on the non-smooth implicit function theorem [16] with some Morse theory.

We also emphasize that because we do not use a perturbation scheme, we cannot give lower bounds on the quality of the linear pieces in the Piecewise-Linear (PL) approximation. This is a clear difference with previous methods [34, 12, 11, 9] whose output is a thick triangulation. Although thickness is an appealing property, it complicates the analysis further and requires perturbation schemes that work fine in theory, but the constants are miserable and the methods do not work in practice in high dimension.

A full version of this paper is available at https://hal.inria.fr/hal-02386193.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

2 Isomanifolds without boundary

Let $f : \mathbb{R}^d \to \mathbb{R}^{d-n}$ be a smooth function and suppose that 0 is a regular value of f, meaning that at every point x such that $f(x) = 0$, the Jacobian of f is non-degenerate. Then the zero set of f is a manifold as a direct consequence of the implicit function theorem, see for example [20, Section 3.5]. We further assume that $f^{-1}(0)$ is compact. As in [1] we consider a triangulation $\mathcal{T}$ of $\mathbb{R}^d$. The function f_{PL} is a linear interpolation of the values of f at the vertices if restricted to a single simplex $\sigma \in \mathcal{T}$. *For any function $g : \mathbb{R}^d \to \mathbb{R}^{d-n}$ we write g^i for the components of g.*

We prove that under certain conditions there is an isotopy from the zero set of f to the zero set of f_{PL}. The proof will be using the Piecewise-Linear (PL) map $F_{PL}(x, \tau) = (1-\tau)f(x)+\tau f_{PL}(x)$, which interpolates between f and f_{PL} and is based on the generalized implicit function theorem. The isotopy is in fact stronger than just the existence of a homeomorphism from the zero set of f to that of f_{PL}.

Our result in particular implies that the zero set f_{PL} is a manifold. So this significantly improves on the result of Allgower and Georg [2, Theorem 15.4.1]. The conditions are also weaker, because we do not require that the zero set avoids simplices that have dimension less than the codimension, see [2, Definition 12.2.2] and the text above [2, Theorem 15.4.1]. The idea to avoid these low dimensional simplices originates with Whitney [34], with whom Allgower and George [2, 1] were apparently unfamiliar. Very heavy perturbation schemes for the vertices of the ambient triangulation $\mathcal{T}$ are required for the manifold to be sufficiently far from simplices in the ambient triangulations that have dimension less than the codimension of the manifold [34, 12]. Various techniques have been developed to compute such perturbations with guarantees. They typically consist in perturbing the position of the sample points or in assigning weights to the points. Complexity bounds are then obtained using volume arguments. See, for example [13, 11, 8, 6]. However, these techniques suffer from several drawbacks.

We are, by definition, only interested in $f^{-1}(0)$ so we can ignore points that are sufficiently far from this zero set. So,

▶ Remark. Write Σ_0 for the set of all $\sigma \in \mathcal{T}$, such that $(f^i)^{-1}(0) \cap \sigma \neq \emptyset$ for all i. Then for all τ, $\{x \mid F_{PL}(x, \tau) = 0\} \subset \Sigma_0$.

Now we define a few constants, depending only on f and the ambient triangulation $\mathcal{T}$, which will be useful in the statements of the main results.

▶ **Definition 2.1.** We define:

$$\gamma_0 = \inf_{x \in \Sigma_0} |\det(\operatorname{grad}(f^i) \cdot \operatorname{grad}(f^j))_{i,j}|,$$
$$\gamma_1 = \sup_{x \in \Sigma_0} \max_i |\operatorname{grad}(f^i)|,$$

where $|\cdot|$ denotes the Euclidean norm,

$$\alpha = \sup_{x \in \Sigma_0} \max_i \|\operatorname{Hes}(f^i)\|_2 = \sup_x \max_i \|(\partial_k \partial_l f^i)_{k,l}\|_2,$$

D is the longest edge length of a simplex in Σ_0, T is the smallest thickness of a simplex in Σ_0. Here $\operatorname{grad}(f^i) = (\partial_j f_i)_j$ denotes the gradient of component f^i, $\det(\operatorname{grad}(f^i) \cdot \operatorname{grad}(f^j))_{i,j}$ denotes the determinant of the matrix with entries $\operatorname{grad}(f^i) \cdot \operatorname{grad}(f^j)$, $\|\cdot\|_2$ the operator 2-norm, and $(\partial_k \partial_l f_i)_{k,l}$ the matrix of second order derivatives, that is the Hessian (Hes). The thickness is the ratio of the height over the longest edge length. We recall the definition

of the operator norm: $\|A\|_p = \max_{x\in\mathbb{R}^n} \frac{|Ax|_p}{|x|_p}$, with $|\cdot|_p$ the p-norm on $\mathbb{R}^n$. We will assume that $\gamma_0, \gamma_1, \alpha, D, T \in (0, \infty)$.

A good choice for $\mathcal{T}$ is the Coxeter triangulation of type A_d, see [17, 15], or the related Freudenthal triangulations, see [24, 25, 22, 33], which can be defined for different values of D while keeping T constant. In this paper, we will thus think of all the above quantities as well as d and n as constants except D and our results will hold for D small enough.

The result

We are going to construct an ambient isotopy. The zero set of $F_{PL}(x, 0)$ (or $f(x)$) gives the smooth isosurface, while the zero set of $F_{PL}(x, 1)$ (or $f_{PL}(x)$) gives the PL approximation, that is the triangulation of the isosurface after possible barycentric subdivision. The map $\tau \mapsto \{x \mid F_{PL}(x, \tau) = 0\}$ in fact gives an isotopy.

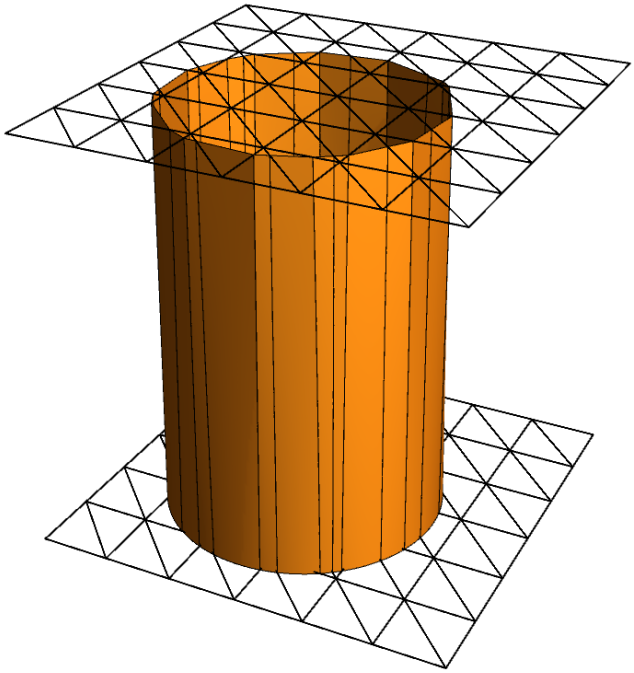


Figure 1 Similarly to Morse theory we find that $f_{PL}^{-1}(0)$ (top) and $f^{-1}(0)$ (bottom) are isotopic if the function τ restricted to $F_{PL}^{-1}(0)$ does not encounter a Morse critical point.

Proving isotopy consists of two technical steps, as well as the use of a standard observation from Morse theory/gradient flow in the third step. The technical steps are 1) Let $\sigma \in \mathcal{T}$. In Corollary 2.6 we show that $\{(x, \tau) \mid F_{PL}(x, \tau) = 0\} \cap (\sigma \times [0, 1])$ is a smooth manifold. 2) In Corollary 2.15 we prove that $F_{PL}^{-1}(0)$ is a manifold using nonsmooth analysis.
In Proposition 2.7 we also see that $F_{PL}^{-1}(0)$ is never tangent to the $\tau = c$ planes, where c is a constant. So we find that the gradient field of τ restricted to $F_{PL}^{-1}(0)$, is piecewise smooth and never vanishes.

Now we arrive at the third step, where we use gradient flows to find an isotopy. This is similar to a standard observation in Morse theory [27, 28], with the exception that we now consider piecewise-smooth instead of smooth vector fields. We refer to Milnor [27] for an excellent introduction, see Lemma 2.4 and Theorem 3.1 in particular.

▶ **Lemma 2.2** (Gradient flow induced isotopies)**.** *The flow of a non-vanishing piecewise-smooth gradient vector field of a function τ on a compact manifold generates a isotopy from $\tau = c_1$ to $\tau = c_2$, where c_1 and c_2 are constants.*

2.1 Estimates for a single simplex

We now first concentrate on a single simplex σ and write f_L for the linear function whose values on the vertices of σ coincide with f.

2.1.1 Preliminaries and variations of know results

We need simple estimates similar to Propositions 2.1 and 2.2 of Allgower and Georg [1].

▶ **Lemma 2.3.** *Let $\sigma \subset \Sigma_0$ and let f_L be as described above. Then $|f_L^i(x) - f^i(x)| \leq 2D^2\alpha$ for all $x \in \sigma$.*

▶ **Proposition 2.4.** *Let $\sigma \subset \Sigma_0$ and let f_L be as described above. Then, for all $x \in \sigma = \{v_k\}$,*

$$|\mathrm{grad} f_L^i - \mathrm{grad} f^i| = \sqrt{\sum_j (\partial_j f_L^i(x) - \partial_j f^i(x))^2} \leq \frac{4dD\alpha}{T}.$$

2.1.2 Estimates on the gradient inside a single simplex

We write $F_L(x,\tau) = (1-\tau)f(x) + \tau f_L(x)$. We note that F_L extends smoothly outside σ. Here and throughout we restrict ourselves to the setting where $\tau \in [0,1]$. The function F_L has $\mathbb{R}^{d-n}$ as image.

We can now state the following

▶ **Lemma 2.5.** *If we write $\mathrm{grad}_{(x,\tau)}$ for the gradient that includes the τ component, we have*

$$|\det(\mathrm{grad}_{(x,\tau)}(F_L^i) \cdot \mathrm{grad}_{(x,\tau)}(F_L^j))_{i,j}| > \gamma_0 - g_1(D), \tag{1}$$

with $g_1(D) = \mathcal{O}(D)$. See (2) in Appendix 3 for the exact expression of g_1.

From the previous statement we immediately have that

▶ **Corollary 2.6** ($F_L^{-1}(0)$ is a manifold in a neighbourhood of $\sigma \times [0,1]$)**.** *If $\gamma_0 > g_1(D)$ the implicit function theorem applies to $F_L(x,\tau)$ inside $\sigma \times [0,1]$. (In fact it applies to an open neighbourhood of this set). In particular, we have proven the first of our two technical steps, $\{(x,\tau) \mid F_{PL}(x,\tau) = 0\} \cap (\sigma \times [0,1])$ is a smooth manifold.*

2.1.3 Transversality with regard to the τ-direction

We will also prove the main result which we need for the third step, that is the gradient of τ restricted to $F_{PL}(x,\tau) = 0$, is piecewise smooth and never vanishes. We now prove inside each $\sigma \times [0,1]$ the gradient of τ on $F_L = 0$ is smooth and does not vanish.

▶ **Proposition 2.7.** *Let $v^1, \ldots, v^{d-n} \in \mathbb{R}^d$, $|v^i| \leq \gamma_1$, for all i, and assume that $\det(v^i \cdot v^j)_{i,j} > \gamma_0$. If $\gamma_0 > g_1(D)$, and $\sqrt{\gamma_0}/\gamma_1^{d-n-1} > \frac{4dD\alpha}{T}$, then inside each $\sigma \times [0,1]$ the gradient of τ on $F_L^{-1}(0)$ is smooth and does not vanish.*

2.2 Global result

We are now going to prove the global result. For this, we need to recall some definitions and results from non-smooth analysis. We refer to [16] for an extensive introduction.

▶ **Definition 2.8** (Generalized Jacobian, Definition 2.6.1 of [16])**.** Let $F : \mathbb{R}^{d+1} \to \mathbb{R}^{d-n}$, where F is assumed to be just Lipschitz. The generalized Jacobian of F at x_0 denoted by $J_F(x_0)$, is the convex hull of all $(d-n) \times (d+1)$-matrices B obtained as the limit of a sequence of the form $J_F(x_i)$, where $x_i \to x_0$ and F is differentiable at x_i.

▶ **Definition 2.9** ([16, page 253])**.** The generalized Jacobian $J_F(x_0)$ is said to be of maximal rank provided every matrix in $J_F(x_0)$ is of maximal rank.

Write $\mathbb{R}^{d+1} = \mathbb{R}^{n+1} \times \mathbb{R}^{d-n}$ and denote the coordinates of $\mathbb{R}^{d+1}$ by (x, y) accordingly. Fix a point (a, b), with $F(a, b) = 0 \in \mathbb{R}^{d-n}$. We now write:

▶ **Notation 2.10** ([16, page 256])**.** *$J_F(x_0, y_0)|_y$ is the set of all $(n+1) \times (n+1)$-matrices M such that, for some $(n+1) \times (d-n)$-matrix N, the $(n+1) \times (d+1)$-matrix $[N, M]$ belongs to $J_F(x_0, y_0)$.*

▶ **Theorem 2.11** (The generalized implicit function theorem [16, page 256])**.** *Suppose that $J_F(a, b)|_y$ is of maximal rank. Then there exists an open set $U \subset \mathbb{R}^{n+1}$ containing a such that there exists a Lipschitz function $g : U \to \mathbb{R}^{d-n}$, such that $g(a) = b$ and $F(x, g(x)) = 0$ for all $x \in U$.*

Because of the definition of α, see Definition 2.1, and Proposition 2.4, we have that $\operatorname{grad}_{(x,\tau)} F_{PL}(x, \tau)$ and $\operatorname{grad}_{(x,\tau)} F_{PL}(\tilde{x}, \tau)$ are close if x and $\tilde{x}$ are. In particular,

▶ **Lemma 2.12.** *Let v be a vertex in $\mathcal{T}$, $x_1, x_2 \in \mathit{star}(v)$, and $\tau_1, \tau_2 \in [0, 1]$, such that $\operatorname{grad}_{(x,\tau)} F^i_{PL}(x_1, \tau_1)$ and $\operatorname{grad}_{(x,\tau)} F^i_{PL}(x_2, \tau_2)$ are well defined, then*

$$|\operatorname{grad}_{(x,\tau)} F^i_{PL}(x_1, \tau_1) - \operatorname{grad}_{(x,\tau)} F^i_{PL}(x_2, \tau_2)| \leq \frac{10d^2 D\alpha}{T} + 4\gamma_1 D + 4D^2\alpha.$$

We now immediately have the same bound on points in the convex hull of a number of such vectors:

▶ **Corollary 2.13.** *Suppose we are in the setting of Lemma 2.12 and $x_0, x_1, \ldots, x_m \in \mathit{star}(v)$, $\tau_0, \ldots, \tau_m \in [0, 1]$, and suppose that $\mu_1, \ldots, \mu_m$ are positive weights such that $\mu_1 + \cdots + \mu_m = 1$ then, $\left|\operatorname{grad}_{(x,\tau)} F^i_{PL}(x_0, \tau_0) - \sum_{k=1}^m \mu_k \operatorname{grad}_{(x,\tau)} F^i_{PL}(x_k, \tau_k)\right| \leq \frac{10d^2 D\alpha}{T} + 4\gamma_1 D + 4D^2\alpha$.*

Using Lemma 2.5 we see

▶ **Lemma 2.14.** *Let v be a vertex in $\mathcal{T}$, $x_1, \ldots, x_m \in \mathit{star}(v)$, and $\tau_1, \ldots, \tau_m \in [0, 1]$, such that $\operatorname{grad}_{(x,\tau)} F^i_{PL}(x_k, \tau_k)$, $k = 0, \ldots, m$ are well defined. If we moreover assume $D \leq 1$, and $\frac{6dD\alpha}{T} \leq \gamma_1$ we have that*

$$\left|\det\left(\left(\sum_{k=1}^m \mu_k \operatorname{grad}_{(x,\tau)} F^i_{PL}(x_k, \tau_k)\right) \cdot \left(\sum_{k=1}^m \mu_k \operatorname{grad}_{(x,\tau)} F_{PL}(x_k, \tau_k)\right)\right)_{i,j}\right| \geq \gamma_0 - g_2(D),$$

with $g_2(D) = \mathcal{O}(D)$. See (3) in Appendix 3 for the exact expression of g_2.

▶ **Corollary 2.15** ($\{x \mid F_{PL}(x, \tau) = 0\}$ is a manifold)**.** *If $D \leq 1$, $\frac{6dD\alpha}{T} \leq \gamma_1$, and $\gamma_0 > g_2(D)$ the generalized implicit function theorem, Theorem 2.11, applies to $F_{PL}(x, \tau) = 0$. In particular, $\{x \mid F_{PL}(x, \tau) = 0\}$ is a manifold.*

This bound is stronger than the one in Corollary 2.6. So, $\{x \mid F_{PL}(x,\tau) = 0\}$ is a Piecewise-Smooth manifold if the conditions of Corollary 2.15 hold. The fact that $F_L(x,\tau) = 0$ is a Piecewise-Smooth manifold and Proposition 2.7 give that the gradient of τ is a Piecewise-Smooth vector field whose flow we can integrate to give an isotopy from the zero set of f to that of f_{PL}. Thus,

▶ **Theorem 2.16.** *If, $D \leq 1$, $\frac{6dD\alpha}{T} \leq \gamma_1$, $\sqrt{\gamma_0}/\gamma_1^{d-n-1} > \frac{4dD\alpha}{T}$, and $\gamma_0 > g_2(D)$ then the zero set of f is isotopic to the zero set of f_{PL}.*

We stress that one can satisfy all conditions by choosing D sufficiently small.

3 Overview of constants

We give an overview. We write Σ_0 for the set of all $\sigma \in \mathcal{T}$, such that $(f^i)^{-1}(0) \cap \sigma \neq$ for all i. We write

$$\begin{aligned}
\gamma_0 &= \inf_{x\in\Sigma_0} |\det(\operatorname{grad}(f^i)\cdot\operatorname{grad}(f^j))_{i,j}| \\
\gamma_1 &= \sup_{x\in\Sigma_0} \max_i |\operatorname{grad}(f^i)| \\
\alpha &= \sup_{x\in\Sigma_0} \max_i \|\operatorname{Hes}(f^i)\|_2 = \sup_x \max_i \|(\partial_k\partial_l f^i)_{k,l}\|_2 \\
D &: \text{the longest edge length of a simplex in } \Sigma_0 \\
T &: \text{the smallest thickness of a simplex in } \Sigma_0.
\end{aligned}$$

$\Xi = \mathbb{R}^d \subset \mathbb{R}^{d+1}$ is the space spanned by the d basis vectors corresponding to the x-directions. The precise expressions for the $g_i(D)$ are:

$$g_1(D) = n^{n+1}\left(\gamma_1 + \frac{6dD\alpha}{T}\right)^{2n-1}\left(2\gamma_1\frac{4dD\alpha}{T} + \left(\frac{6dD\alpha}{T}\right)^2\right) \tag{2}$$

$$g_2(D) = n^{n+1}\left(2^{2n-1}\gamma_1^{2n}\left(\frac{14dD\alpha}{T}\right) + 5^{n-1}\gamma_1^{2n-1}(2d+5)\left(\frac{24d^2D\alpha}{T} + 9\gamma_1 D\right)\right) \tag{3}$$

If $\frac{4dD\alpha}{T} \leq \gamma_1$, $g_1(D)$ can be replaced by the simpler $34\cdot\left(\frac{5}{2}\right)^{2n-1} n^{n+1}\gamma_1^{2n}\frac{dD\alpha}{T}$.

References

1 Eugene L. Allgower and Kurt Georg. Estimates for piecewise linear approximations of implicitly defined manifolds. *Applied Mathematics Letters*, 2(2):111–115, 1989.

2 Eugene L. Allgower and Kurt Georg. *Numerical continuation methods: an introduction*, volume 13. Springer Science & Business Media, 1990.

3 N. Amenta and M. Bern. Surface reconstruction by Voronoi filtering. *Discrete and Computational Geometry*, 22(4):481–504, 1999.

4 Dominique Attali and André Lieutier. Geometry-driven collapses for converting a Čech complex into a triangulation of a nicely triangulable shape. *Discrete Comput. Geom.*, 54(4):798–825, December 2015. URL: `http://dx.doi.org/10.1007/s00454-015-9733-7`, `doi:10.1007/s00454-015-9733-7`.

5 J.-D. Boissonnat, R. Dyer, and A. Ghosh. The Stability of Delaunay Triangulations. *International Journal of Computional Geometry & Applications*, 23(4-5):303–334, 2013. URL: `http://dx.doi.org/10.1142/S0218195913600078`, `doi:10.1142/S0218195913600078`.

6 Jean-Daniel Boissonnat, Frédéric Chazal, and Mariette Yvinec. *Geometric and Topological Inference*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2018. doi:10.1017/9781108297806.

7 Jean-Daniel Boissonnat, David Cohen-Steiner, and Gert Vegter. Isotopic implicit surface meshing. *Discrete & Computational Geometry*, 39(1):138–157, Mar 2008. URL: https://doi.org/10.1007/s00454-007-9011-4, doi:10.1007/s00454-007-9011-4.

8 Jean-Daniel Boissonnat, Ramsay Dyer, and Arijit Ghosh. Delaunay stability via perturbations. *International Journal of Computational Geometry & Applications*, 24(02):125–152, 2014.

9 Jean-Daniel Boissonnat, Ramsay Dyer, and Arijit Ghosh. Delaunay Triangulation of Manifolds. *Foundations of Computational Mathematics*, 45:38, 2017. URL: https://hal.inria.fr/hal-01509888, doi:10.1007/s10208-017-9344-1.

10 Jean-Daniel Boissonnat, Ramsay Dyer, Arijit Ghosh, André Lieutier, and Mathijs Wintraecken. Local conditions for triangulating submanifolds of Euclidean space. Accepted for Discrete and Computational Geometry with minor revision, July 2019. URL: https://hal.inria.fr/hal-02267620.

11 Jean-Daniel Boissonnat and Arijit Ghosh. Manifold reconstruction using tangential Delaunay complexes. *Discrete & Computational Geometry*, 51(1):221–267, 2014.

12 Jean-Daniel Boissonnat, Siargey Kachanovich, and Mathijs Wintraecken. Triangulating submanifolds: An elementary and quantified version of Whitney's method. Preprint, December 2018. URL: https://hal.inria.fr/hal-01950149.

13 S-W. Cheng, T. K. Dey, and E. A. Ramos. Manifold reconstruction from point samples. In *Proc. 16th ACM-SIAM Symp. Discrete Algorithms*, pages 1018–1027, 2005.

14 S.-W. Cheng, T. K. Dey, and J. R. Shewchuk. *Delaunay Mesh Generation*. Computer and information science series. CRC Press, 2013.

15 Aruni Choudhary, Siargey Kachanovich, and Mathijs Wintraecken. Coxeter triangulations have good quality. Preprint, December 2017. URL: https://hal.inria.fr/hal-01667404.

16 Frank H. Clarke. *Optimization and Nonsmooth Analysis*, volume 5 of *Classics in applied mathematics*. SIAM, 1990.

17 Harold SM Coxeter. Discrete groups generated by reflections. *Annals of Mathematics*, pages 588–621, 1934.

18 T. K. Dey. *Curve and Surface Reconstruction; Algorithms with Mathematical Analysis*. Cambridge University Press, 2007.

19 Akio Doi and Akio Koide. An efficient method of triangulating equi-valued surfaces by using tetrahedral cells. *IEICE TRANSACTIONS on Information and Systems*, E74-D, 1991.

20 J. J. Duistermaat and J. A. C. Kolk. *Multidimensional Real Analysis I: Differentiation*. Number 86 in Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2004.

21 R. Dyer, H. Zhang, and T. Möller. Surface sampling and the intrinsic Voronoi diagram. *Computer Graphics Forum (Special Issue of Symp. Geometry Processing)*, 27(5):1393–1402, 2008.

22 B Curtis Eaves. *A course in triangulations for solving equations with deformations*, volume 234. Lecture Notes in Economics and Mathematical Systems, 1984.

23 Herbert Edelsbrunner and Nimish R. Shah. Triangulating topological spaces. *International Journal of Computational Geometry & Applications*, 7(04):365–378, 1997.

24 Hans Freudenthal. Simplizialzerlegungen von beschrankter flachheit. *Annals of Mathematics*, pages 580–582, 1942.

25 Harold W Kuhn. Some combinatorial lemmas in topology. *IBM Journal of research and development*, 4(5):518–524, 1960.

26 William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM siggraph computer graphics*, volume 21, pages 163–169. ACM, 1987.

27 J. Milnor. *Morse Theory*. Princeton University Press, 1969.

28 John Milnor. *Lectures on the H-Cobordism Theorem*. Princeton University Press, 1965. URL: http://www.jstor.org/stable/j.ctt183psc9.

29 Timothy S. Newman and Hong Yi. A survey of the marching cubes algorithm. *Computers & Graphics*, 30(5):854 – 879, 2006. URL: http://www.sciencedirect.com/science/article/pii/S0097849306001336, doi:https://doi.org/10.1016/j.cag.2006.07.021.

30 Simon Plantinga and Gert Vegter. Isotopic meshing of implicit surfaces. *The Visual Computer*, 23(1):45–58, 2007.

31 Mael Rouxel-Labbé, Mathijs Wintraecken, and Jean-Daniel Boissonnat. Discretized Riemannian Delaunay Triangulations. Research Report RR-9103, INRIA Sophia Antipolis - Méditerranée, October 2017. URL: https://hal.inria.fr/hal-01612924.

32 Jonathan Richard Shewchuk. Lecture notes on Delaunay mesh generation, 1999.

33 Michael J Todd. *The computation of fixed points and applications*, volume 124. Lecture Notes inEconomics and Mathematical Systems, 1976.

34 H. Whitney. *Geometric Integration Theory*. Princeton University Press, 1957.

Holes and islands in random point sets*

Martin Balko[1], Manfred Scheucher[2], and Pavel Valtr[1]

1 **Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic,**
{balko}@kam.mff.cuni.cz

2 **Institut für Mathematik, Technische Universität Berlin, Germany,**
{scheucher}@math.tu-berlin.de

Abstract

For $d \in \mathbb{N}$, let S be a finite set of points in $\mathbb{R}^d$ in general position. A set H of k points from S is a *k-hole* in S if all points from H lie on the boundary of the convex hull $\mathrm{conv}(H)$ of H and the interior of $\mathrm{conv}(H)$ does not contain any point from S. A set I of k points from S is a *k-island* in S if $\mathrm{conv}(I) \cap S = I$. Note that each k-hole in S is a k-island in S.

For fixed positive integers d, k and a convex body K in $\mathbb{R}^d$ with d-dimensional Lebesgue measure 1, let S be a set of n points chosen uniformly and independently at random from K. We show that the expected number of k-islands in S is in $O(n^d)$. In the case $k = d+1$, we prove that the expected number of empty simplices (that is, $(d+1)$-holes) in S is at most $2^{d-1} \cdot d! \cdot \binom{n}{d}$. Our results improve and generalize previous bounds by Bárány and Füredi (1987), Valtr (1995), Fabila-Monroy and Huemer (2012), and Fabila-Monroy, Huemer, and Mitsche (2015).

1 Introduction

For $d \in \mathbb{N}$, let S be a finite set of points in $\mathbb{R}^d$. The set S is in *general position* if, for every $k = 1, \ldots, d-1$, no $k+2$ points of S lie in an affine k-dimensional subspace. A set H of k points from S is a *k-hole* in S if H is in convex position and the interior of the convex hull $\mathrm{conv}(H)$ of H does not contain any point from S; see Figure 1 for an illustration in the plane. We say that a subset of S is a *hole* in S if it is a k-hole in S for some integer k.

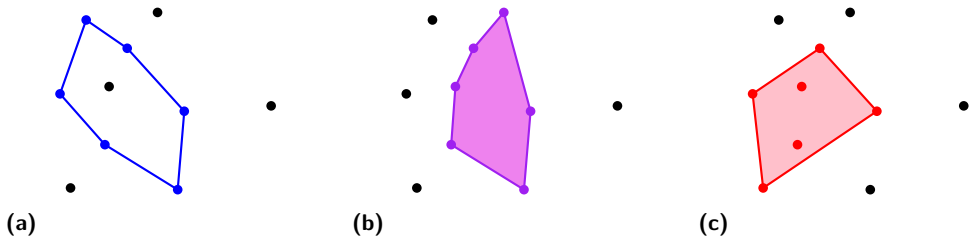


Figure 1 (a) A 6-tuple of points in convex position in a planar set S of 10 points. (b) A 6-hole in S. (c) A 6-island in S whose points are not in convex position.

Let $h(k)$ be the smallest positive integer N such that every set of N points in general position in the plane contains a k-hole. In the 1970s, Erdős [7] asked whether the number $h(k)$

* M. Balko was supported by the grant no. 18-19158S of the Czech Science Foundation (GAČR), by the Center for Foundations of Modern Computer Science (Charles University project UNCE/SCI/004), and by the PRIMUS/17/SCI/3 project of Charles University. M. Scheucher was supported by DFG Grant FE 340/12-1. P. Valtr was supported by the grant no. 18-19158S of the Czech Science Foundation (GAČR) and by the PRIMUS/17/SCI/3 project of Charles University. This article is part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 810115).

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

exists for every $k \in \mathbb{N}$. It was shown in the 1970s and 1980s that $h(4) = 5$, $h(5) = 10$ [12], and that $h(k)$ does not exist for every $k \geq 7$ [13]. That is, while every sufficiently large set contains a 4-hole and a 5-hole, Horton constructed arbitrarily large sets with no 7-holes. His construction was generalized to so-called *Horton sets* by Valtr [18]. The existence of 6-holes in every sufficiently large point set remained open until 2007, when Gerken [11] and Nicolas [16] independently showed that $h(6)$ exists; see also [20].

These problems were also considered in higher dimensions. For $d \geq 2$, let $h_d(k)$ be the smallest positive integer N such that every set of N points in general position in $\mathbb{R}^d$ contains a k-hole. In particular, $h_2(k) = h(k)$ for every k. Valtr [18] showed that $h_d(k)$ exists for $k \leq 2d + 1$ but it does not exist for $k > 2^{d-1}(P(d-1)+1)$, where $P(d-1)$ denotes the product of the first $d-1$ prime numbers. The latter result was obtained by constructing multidimensional analogues of the Horton sets.

After the existence of k-holes was settled, counting the minimum number $H_k(n)$ of k-holes in any set of n points in the plane in general position attracted a lot of attention. It is known, and not difficult to show, that $H_3(n)$ and $H_4(n)$ are in $\Omega(n^2)$. The currently best known lower bounds on $H_3(n)$ and $H_4(n)$ were proved in [1]. The best known upper bounds are due to Bárány and Valtr [6]. Altogether, these estimates are

$$n^2 + \Omega(n \log^{2/3} n) \leq H_3(n) \leq 1.6196n^2 + o(n^2)$$

and

$$\tfrac{n^2}{2} + \Omega(n \log^{3/4} n) \leq H_4(n) \leq 1.9397n^2 + o(n^2).$$

For $H_5(n)$ and $H_6(n)$, the best quadratic upper bounds can be found in [6]. The best lower bounds, however, are only $H_5(n) \geq \Omega(n \log^{4/5} n)$ [1] and $H_6(n) \geq \Omega(n)$ [21]. For more details, we also refer to the second author's dissertation [17].

The quadratic upper bound on $H_3(n)$ can be also obtained using random point sets. For $d \in \mathbb{N}$, a *convex body* in $\mathbb{R}^d$ is a compact convex set in $\mathbb{R}^d$ with a nonempty interior. Let k be a positive integer and let $K \subseteq \mathbb{R}^d$ be a convex body with d-dimensional Lebesgue measure $\lambda_d(K) = 1$. We use $EH^K_{d,k}(n)$ to denote the expected number of k-holes in sets of n points chosen independently and uniformly at random from K. The quadratic upper bound on $H_3(n)$ then also follows from the following bound of Bárány and Füredi [5] on the expected number of $(d+1)$-holes:

$$EH^K_{d,d+1}(n) \leq (2d)^{2d^2} \cdot \binom{n}{d} \tag{1}$$

for any d and K. In the plane, Bárány and Füredi [5] proved $EH^K_{2,3}(n) \leq 2n^2 + O(n \log n)$ for every K. This bound was later slightly improved by Valtr [19], who showed $EH^K_{2,3}(n) \leq 4\binom{n}{2}$ for any K. In the other direction, every set of n points in $\mathbb{R}^d$ in general position contains at least $\binom{n-1}{d}$ $(d+1)$-holes [5, 14].

The expected number $EH^K_{2,4}(n)$ of 4-holes in random sets of n points in the plane was considered by Fabila-Monroy, Huemer, and Mitsche [10], who showed

$$EH^K_{2,4}(n) \leq 18\pi D^2 n^2 + o(n^2) \tag{2}$$

for any K, where $D = D(K)$ is the diameter of K. Since we have $D \geq 2/\sqrt{\pi}$, by the Isodiametric inequality [8], the leading constant in (2) is at least 72 for any K.

In this paper, we study the number of k-holes in random point sets in $\mathbb{R}^d$. In particular, we obtain results that imply quadratic upper bounds on $H_k(n)$ for any fixed k and that both strengthen and generalize the bounds by Bárány and Füredi [5], Valtr [19], and Fabila-Monroy, Huemer, and Mitsche [10].

2 Our results

Throughout the whole paper we only consider point sets in $\mathbb{R}^d$ that are finite and in general position.

2.1 Islands and holes in random point sets

First, we prove a result that gives the estimate $O(n^d)$ on the minimum number of k-holes in a set of n points in $\mathbb{R}^d$ for any fixed d and k. In fact, we prove the upper bound $O(n^d)$ even for so-called k-islands, which are also frequently studied in discrete geometry. A set I of k points from a point set $S \subseteq \mathbb{R}^d$ is a *k-island* in S if $\mathrm{conv}(I) \cap S = I$; see part (c) of Figure 1. Note that k-holes in S are exactly those k-islands in S that are in convex position. A subset of S is an *island* in S if it is a k-island in S for some integer k.

▶ **Theorem 2.1.** *Let $d \geq 2$ and $k \geq d+1$ be integers and let K be a convex body in $\mathbb{R}^d$ with $\lambda_d(K) = 1$. If S is a set of $n \geq k$ points chosen uniformly and independently at random from K, then the expected number of k-islands in S is at most*

$$2^{d-1} \cdot \left(2d^{2d-1}\binom{k}{\lfloor d/2 \rfloor}\right)^{k-d-1} \cdot (k-d) \cdot \frac{n(n-1)\cdots(n-k+2)}{(n-k+1)^{k-d-1}},$$

which is in $O(n^d)$ for any fixed d and k.

The bound in Theorem 2.1 is tight up to a constant multiplicative factor that depends on d and k, as, for any fixed $k \geq d$, every set S of n points in $\mathbb{R}^d$ in general position contains at least $\Omega(n^d)$ k-islands. To see this, observe that any d-tuple T of points from S determines a k-island with $k-d$ closest points to the hyperplane spanned by T (ties can be broken by, for example, taking points with lexicographically smallest coordinates), as S is in general position and thus T is a d-hole in S. Any such k-tuple of points from S contains $\binom{k}{d}$ d-tuples of points from S and thus we have at least $\binom{n}{d}/\binom{k}{d} \in \Omega(n^d)$ k-islands in S.

Thus, by Theorem 2.1, random point sets in $\mathbb{R}^d$ asymptotically achieve the minimum number of k-islands. This is in contrast with the fact that, unlike Horton sets, they contain arbitrarily large holes. Quite recently, Balogh, González-Aguilar, and Salazar [3] showed that the expected number of vertices of the largest hole in a set of n random points chosen independently and uniformly over a convex body in the plane is in $\Theta(\log n/(\log\log n))$.

For k-holes, we modify the proof of Theorem 2.1 to obtain a slightly better estimate.

▶ **Theorem 2.2.** *Let $d \geq 2$ and $k \geq d+1$ be integers and let K be a convex body in $\mathbb{R}^d$ with $\lambda_d(K) = 1$. If S is a set of $n \geq k$ points chosen uniformly and independently at random from K, then the expected number $EH^K_{d,k}(n)$ of k-holes in S is in $O(n^d)$ for any fixed d and k. More precisely,*

$$EH^K_{d,k}(n) \leq 2^{d-1} \cdot \left(2d^{2d-1}\binom{k}{\lfloor d/2 \rfloor}\right)^{k-d-1} \cdot \frac{n(n-1)\cdots(n-k+2)}{(k-d-1)! \cdot (n-k+1)^{k-d-1}}.$$

For $d = 2$ and $k = 4$, Theorem 2.2 implies $EH^K_{2,4}(n) \leq 128 \cdot n^2 + o(n^2)$ for any K, which is a worse estimate than (2) if the diameter of K is at most $8/(3\sqrt{\pi}) \simeq 1.5$. However, the proof of Theorem 2.2 can be modified to give $EH^K_{2,4}(n) \leq 12n^2 + o(n^2)$ for any K, which is always better than (2). We believe that the leading constant in $EH^K_{2,4}(n)$ can be estimated even more precisely and we hope to discuss this direction in future work.

In the case $k = d+1$, the bound in Theorem 2.2 simplifies to the following estimate on the expected number of $(d+1)$-holes (also called *empty simplices*) in random sets of n points in $\mathbb{R}^d$.

▶ **Corollary 2.3.** *Let $d \geq 2$ be an integer and let K be a convex body in $\mathbb{R}^d$ with $\lambda_d(K) = 1$. If S is a set of n points chosen uniformly and independently at random from K, then the expected number of $(d+1)$-holes in S satisfies*

$$EH^K_{d,d+1}(n) \leq 2^{d-1} \cdot d! \cdot \binom{n}{d}.$$

Corollary 2.3 is stronger than the bound (1) by Bárány and Füredi [5] and, in the planar case, coincides with the bound $EH^K_{2,3}(n) \leq 4\binom{n}{2}$ by Valtr [19]. In fact, the bound in the plane seems to be tight up to a smaller order term. Again, we hope to discuss this direction in future work.

We also consider islands of all possible sizes and show that their expected number is in $2^{\Theta\left(n^{(d-1)/(d+1)}\right)}$.

▶ **Theorem 2.4.** *Let $d \geq 2$ be an integer and let K be a convex body in $\mathbb{R}^d$ with $\lambda_d(K) = 1$. Then there are constants $C_1 = C_1(d)$, $C_2 = C_2(d)$, and $n_0 = n_0(d)$ such that for every set S of $n \geq n_0$ points chosen uniformly and independently at random from K the expected number $\mathbb{E}[X]$ of islands in S satisfies*

$$2^{C_1 \cdot n^{(d-1)/(d+1)}} \leq \mathbb{E}[X] \leq 2^{C_2 \cdot n^{(d-1)/(d+1)}}.$$

Since each island in S has at most n points, there is a $k \in \{1, \ldots, n\}$ such that the expected number of k-islands in S is at least $(1/n)$-fraction of the expected number of all islands, which is still in $2^{\Omega(n^{(d-1)/(d+1)})}$. This shows that the expected number of k-islands can become asymptotically much larger than $O(n^d)$ if k is not fixed.

2.2 Islands and holes in d-Horton sets

To our knowledge, Theorem 2.1 is the first nontrivial upper bound on the minimum number of k-islands a point set in $\mathbb{R}^d$ with $d > 2$ can have. For $d = 2$, Fabila-Monroy and Huemer [9] showed that, for every fixed $k \in \mathbb{N}$, the Horton sets with n points contain only $O(n^2)$ k-islands. For $d > 2$, Valtr [18] introduced a d-dimensional analogue of Horton sets. Perhaps surprisingly, these sets contain asymptotically more than $O(n^d)$ k-islands for $k \geq d+1$. For each k with $d+1 \leq k \leq 3 \cdot 2^{d-1}$, they even contain asymptotically more than $O(n^d)$ k-holes.

▶ **Theorem 2.5.** *Let $d \geq 2$ and k be fixed positive integers. Then every d-dimensional Horton set H with n points contains at least $\Omega(n^{\min\{2^{d-1},k\}})$ k-islands in H. If $k \leq 3 \cdot 2^{d-1}$, then H even contains at least $\Omega(n^{\min\{2^{d-1},k\}})$ k-holes in H.*

3 Idea of the proof of Theorem 2.1

Let d and k be fixed integers with $k > d \geq 2$. To show that the number of k-islands in a set S of n points chosen uniformly and independently at random from the convex body $K \subset \mathbb{R}^d$ is of order $O(n^d)$, we prove an $O(1/n^{k-d})$ bound on the probability that an ordered k-tuple $I = (p_1, \ldots, p_k)$ of points from S determines a k-island in S with the following two additional properties:

(P1) The points $p_1, \ldots, p_{d+1}$ determine the largest volume simplex $\triangle$ with vertices in I.
(P2) For some $a \in \{0, \ldots, k-d-1\}$, the points $p_{d+2}, \ldots, p_{d+1+a}$ lie inside $\triangle$ and the points $p_{d+2+a}, \ldots, p_k$ lie outside $\triangle$. Moreover, roughly speaking, the points $p_{d+2+a}, \ldots, p_k$ have increasing distance to $\triangle$ as their index increases.

First, we prove an $O(1/n^{a+1})$ bound on the probability that $\triangle$ contains precisely the points $p_{d+2}, \dots, p_{d+1+a}$ from S, which means that the points $p_1, \dots, p_{d+1+a}$ determine an island in S.

Next, for $i = d+2+a, \dots, k$, we show that, conditioned on the fact that the $(i-1)$-tuple $(p_1, \dots, p_{i-1})$ determines an island in S satisfying (P1) and (P2), the i-tuple $(p_1, \dots, p_i)$ determines an island in S satisfying (P1) and (P2) with probability $O(1/n)$.

Then it immediately follows that the probability that I determines a k-island in S with the desired properties is at most

$$O\left(1/n^{a+1} \cdot (1/n)^{k-(d+1+a)}\right) = O(1/n^{k-d}).$$

Since there are $n \cdot (n-1) \cdots (n-k+1) = O(n^k)$ possibilities to select such an ordered subset I and each k-island in S is counted at most $k!$ times, we obtain the desired bound

$$O\left(n^k \cdot n^{d-k} \cdot k!\right) = O(n^d)$$

on the expected number of k-islands in S.

To be more precise: to get rid of technical difficulties and also to obtain better multiplicative constants, we consider a so-called *canonical labeling* of the points $p_1, \dots, p_k$ which requires more conditions on I than properties (P1) and (P2). This labeling is unique and therefore we avoid the above mentioned overcounting and get rid of the factor $k!$.

References

1. O. Aichholzer, M. Balko, T. Hackl, J. Kynčl, I. Parada, M. Scheucher, P. Valtr, and B. Vogtenhuber. A Superlinear Lower Bound on the Number of 5-Holes. In *33rd International Symposium on Computational Geometry (SoCG 2017)*, volume 77 of *Leibniz International Proceedings in Informatics*, pages 8:1–8:16, 2017. Full version: `http://arXiv.org/abs/1703.05253`.
2. G. E. Andrews, R. Askey, and R. Roy. *Special functions*, volume 71 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 1999.
3. J. Balogh, H. González-Aguilar, and G. Salazar. Large convex holes in random point sets. *Computational Geometry*, 46(6):725–733, 2013.
4. I. Bárány. A note on Sylvester's four-point problem. *Studia Scientiarum Mathematicarum Hungarica. A Quarterly of the Hungarian Academy of Sciences*, 38:73–77, 2001.
5. I. Bárány and Z. Füredi. Empty simplices in Euclidean space. *Canadian Mathematical Bulletin*, 30(4):436–445, 1987.
6. I. Bárány and P. Valtr. Planar point sets with a small number of empty convex polygons. *Studia Scientiarum Mathematicarum Hungarica*, 41(2):243–266, 2004.
7. P. Erdős. Some more problems on elementary geometry. *Australian Mathematical Society Gazette*, 5:52–54, 1978.
8. L. C. Evans and R. F. Gariepy. *Measure theory and fine properties of functions*. Textbooks in Mathematics. CRC Press, Boca Raton, FL, revised edition, 2015.
9. R. Fabila-Monroy and C. Huemer. Covering Islands in Plane Point Sets. In *Computational Geometry: XIV Spanish Meeting on Computational Geometry, EGC 2011*, volume 7579 of *Lecture Notes in Computer Science*, pages 220–225. Springer, 2012.
10. R. Fabila-Monroy, C. Huemer, and D. Mitsche. Empty non-convex and convex four-gons in random point sets. *Studia Scientiarum Mathematicarum Hungarica. A Quarterly of the Hungarian Academy of Sciences*, 52(1):52–64, 2015.
11. T. Gerken. Empty Convex Hexagons in Planar Point Sets. *Discrete & Computational Geometry*, 39(1):239–272, 2008.

12 H. Harborth. Konvexe Fünfecke in ebenen Punktmengen. *Elemente der Mathematik*, 33:116–118, 1978. In German.

13 J. D. Horton. Sets with no empty convex 7-gons. *Canadian Mathematical Bulletin*, 26:482–484, 1983.

14 M. Katchalski and A. Meir. On empty triangles determined by points in the plane. *Acta Mathematica Hungarica*, 51(3-4):323–328, 1988.

15 J. Matoušek. *Lectures on discrete geometry*, volume 212 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 2002.

16 M. C. Nicolas. The Empty Hexagon Theorem. *Discrete & Computational Geometry*, 38(2):389–397, 2007.

17 M. Scheucher. *Points, Lines, and Circles: Some Contributions to Combinatorial Geometry*. PhD thesis, Technische Universität Berlin, Institut für Mathematik, 2019.

18 P. Valtr. Sets in $\mathbb{R}^d$ with no large empty convex subsets. *Discrete Mathematics*, 108(1):115–124, 1992.

19 P. Valtr. On the minimum number of empty polygons in planar point sets. *Studia Scientiarum Mathematicarum Hungarica*, pages 155–163, 1995.

20 P. Valtr. On empty hexagons. In *Surveys on Discrete and Computational Geometry: Twenty Years Later*, volume 453 of *Contemporary Mathematics*, pages 433–441. American Mathematical Society, 2008.

21 P. Valtr. On empty pentagons and hexagons in planar point sets. In *Proceedings of Computing: The Eighteenth Australasian Theory Symposium (CATS 2012)*, pages 47–48, Melbourne, Australia, 2012.

Computing Area-Optimal Simple Polygonalizations

Sándor P. Fekete[1], Andreas Haas[1], Phillip Keldenich[1], Michael Perk[1], and Arne Schmidt[1]

1 **Department of Computer Science, TU Braunschweig, Germany**
{s.fekete, a.haas, p.keldenich, m.perk, arne.schmidt}@tu-bs.de

Abstract

We consider methods for finding a simple polygon of minimum (Min-Area) or maximum (Max-Area) possible area for a given set of points in the plane. Both problems are known to be NP-hard; at the center of the recent CG Challenge, practical methods have received considerable attention. However, previous methods focused on heuristic methods, with no proof of optimality. We develop exact methods, based on a combination of geometry and integer programming. As a result, we are able to solve instances of up to $n = 25$ points to provable optimality. While this extends the range of solvable instances by a considerable amount, it also illustrates the practical difficulty of both problem variants.

1 Introduction

While the classic geometric Traveling Salesman Problem (TSP) is to find a (simple) polygon with a given set of vertices that has shortest perimeter, it is natural to look for a simple polygon with a given set of vertices that minimizes another basic geometric measure: the enclosed area. The problem Min-Area asks for a simple polygon with minimum enclosed area, while Max-Area demands one of maximum area; see Figure 1 for an illustration.

Both problem variants were shown to be NP-complete by Fekete [2,3,6], who also showed that no polynomial-time approximation scheme (PTAS) exists for Min-Area problem and gave a $\frac{1}{2}$-approximation algorithm for Max-Area.

1.1 Related Work

Most previous practical work has focused on finding heuristics for both problems. Taranilla et al. [11] proposed three different heuristics. Peethambaran [9,10] later proposed randomized and greedy algorithms on solving Min-Area as well as the d-dimensional variant of both Min-Area and Max-Area. Considerable recent attention and progress was triggered by the 2019 CG Challenge, which asked contestants to find good solutions for a wide spectrum of benchmark instances with up to 1,000,000 points; details will be described in a forthcoming special issue of the *Journal of Experimental Algorithms* [1].

Despite this focus, there has only been a limited amount of work on computing provably optimal solutions for instances of interesting size. The only notable exception is by Fekete et al. [4], who were able to solve all instances of Min-Area with up to $n = 14$ and some with up to $n = 16$ points, as well as all instances of Max-Area with up to $n = 17$ and some with up to $n = 19$ points. This illustrates the inherent practical difficulty, which differs considerably from the closely related TSP, for which even straightforward IP-based approaches can yield provably optimal solutions for instances with hundreds of points, and sophisticated methods can solve instances with tens of thousands of points.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

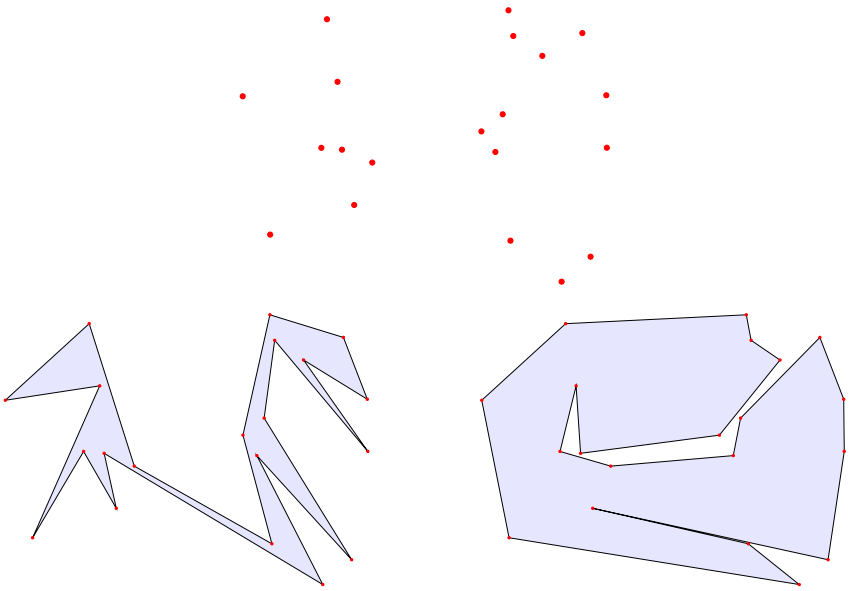


Figure 1 (Top) A set of 20 points. (Bottom Left) Min-Area solution. (Bottom Right) Max-Area solution.

1.2 Our Results

We present a systematic study of exact methods for Min-Area and Max-Area polygonizations. We show that a number of careful enhancements can help to extend the range of instances that can be solved to provable optimality, with different approaches working better for the two problem variants. On the other hand, our work shows that the problems appear to be practically harder than other geometric optimization problems such as the Euclidean TSP.

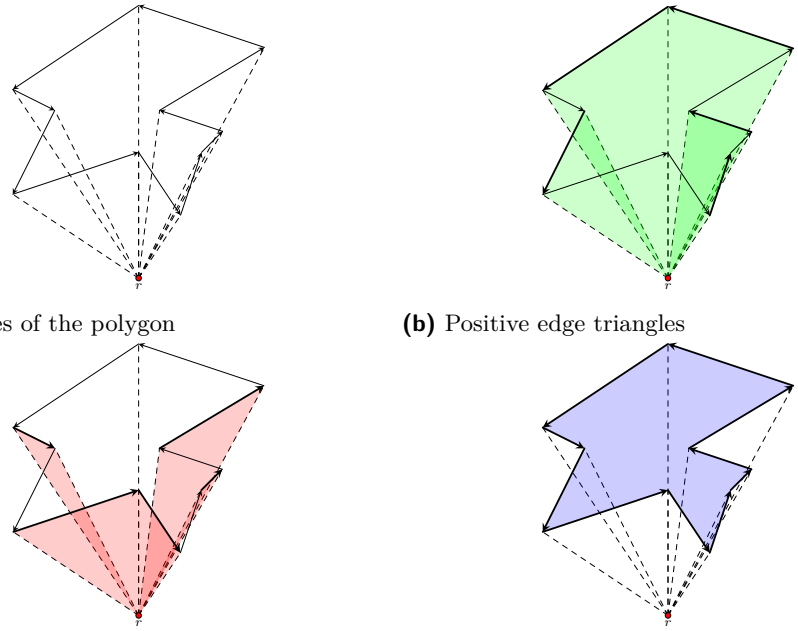
2 Tools

We considered two models based on integer programming: an *edge-based formulation* (described in Section 2.1) and a *triangle-based formulation* (described in Section 2.2). In addition, we developed a number of further refinements and improvements (described in Section 2.3).

2.1 Edge-Based Formulation

The first formulation is based on considering *directed* edges of the polygon boundary. As shown in Figure 2, the area $A_{\mathcal{P}}$ of a polygon $\mathcal{P}$ can be computed by adding the (signed) triangle areas f_e that are formed by edges e and an arbitrary, fixed reference point r.

This gives rise to an integer program in which the choice of half-edges $e = (i, j)$ is modeled by 0-1 variables $z_e = z_{ij}$. In contrast to Euclidean TSP, intersections between edges must be prevented with *intersection constraints* (5). The *slab inequalities* (6) ensure that the polygon is oriented in a counterclockwise manner and thus the area calculation yields the correct result. A *slab* D is a vertical strip bounded by the x-coordinates of two consecutive points in the order of x-coordinates of points. The edges of slab D get ordered by the y-coordinate at the intersection with the (centered) halving line between the points. Now the bottommost chosen edge has to be oriented from left to right and the topmost one from right to left,



(a) Triangles of the polygon

(b) Positive edge triangles

(c) Negative edge triangles

(d) Calculated difference between (b) and (c)

Figure 2 Area computation of a polygon using a reference point r

while chosen edges inbetween have to alternate in their direction. Furthermore, we introduce *subtour constraints* (7) that enforce a polygonization that visits all points in S.

$$\{\min, \max\} \sum_{e^+ \in E^r} z_{e^+} \cdot f_e - \sum_{e^- \in E^r} z_{e^-} \cdot f_e \tag{1}$$

$$\forall s_i \in S: \quad \sum_{(j,i) \in \delta^+(s_i)} z_{ji} = 1 \tag{2}$$

$$\forall s_i \in S: \quad \sum_{(i,j) \in \delta^-(s_i)} z_{ij} = 1 \tag{3}$$

$$\forall e = \{i,j\} \in E: \quad z_{ij} + z_{ji} \leq 1 \tag{4}$$

$$\forall \text{ intersecting } \{i,j\},\{k,l\} \in E: \quad z_{ij} + z_{ji} + z_{kl} + z_{lk} \leq 1 \tag{5}$$

$$(\forall \text{ slabs } D)(\forall m = 1, \ldots, |D|): \quad \sum_{i=1}^{m} z_{e^{lr}_{i_D}} - z_{e^{rl}_{i_D}} \tag{6}$$

$$\forall D \subsetneq S, D \neq \emptyset: \quad \begin{matrix} \sum_{(k,l) \in \delta^-(D)} z_{kl} \geq 1 \\ \sum_{(k,l) \in \delta^+(D)} z_{kl} \geq 1 \end{matrix} \tag{7}$$

$$\forall e = \{i,j\} \in E: \quad z_{ij}, z_{ji} \in \{0,1\} \tag{8}$$

As there are $\Theta(n^2)$ possible edges, the number of intersection constraints may be as big as $\Theta(n^4)$. Moreover, the number of subtour constraints (7) may be exponential, so they are only added when necessary in an incremental fashion.

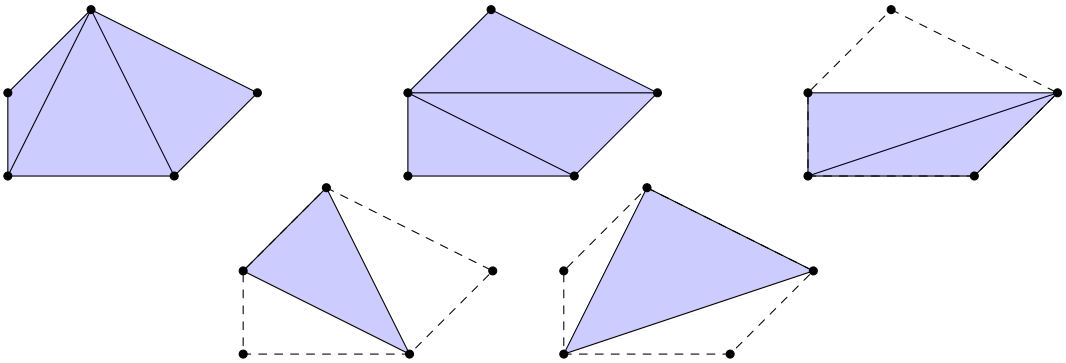


Figure 3 A set of five points and its ten empty triangles.

2.2 Triangle-Based Formulation

An alternative is the triangle-based formulation, which considers the set $T(P)$ of possibly $\binom{n}{3}$ many empty triangles of a point set P; see Figure 3 for an illustration. Making use of the fact that a simple polygon with n vertices consists of $(n-2)$ empty triangles with non-intersection interiors, we get the following IP formulation, in which the presence of an empty triangle $\triangle$ is described by a 0-1 variable $x_\triangle$.

The objective function (9) is the sum over the chosen triangles areas. *Triangle constraint* (10) ensures that we choose exactly $n-2$ triangles, which is the number of triangles in a triangulation of a simple polygon. Furthermore, *point constraints* (11) guarantee that a solution has at least one adjacent triangle at each point $s_i \in S$. Moreover, *intersection constraints* (12) ensure that we only select triangles with disjoint interiors. Finally, the *subtour constraints* (13) ensure that the set of selected triangles forms a simple polygon.

$$\{\min, \max\} \sum_{\triangle \in T} f_\triangle \cdot x_\triangle \tag{9}$$

$$\sum_{\triangle \in T} x_\triangle = n - 2 \tag{10}$$

$$\forall s_i \in S: \quad \sum_{\triangle \in \delta(s_i)} x_\triangle \geq 1 \tag{11}$$

$$\forall \text{intersecting} \quad \triangle_i, \triangle_j \in T: \quad x_{\triangle_i} + x_{\triangle_j} \leq 1 \tag{12}$$

$$\forall D \subsetneq T, D \neq \emptyset, |D| \leq n-3: \quad \sum_{\triangle \in D} x_\triangle \leq \sum_{\triangle \in \delta(D)} x_\triangle + |D| - 1 \tag{13}$$

$$\forall \triangle \in T: \quad x_\triangle \in \{0, 1\} \tag{14}$$

As there are $\Theta(n^3)$ possible empty triangles, the number of intersection constraints may be as big as $\Theta(n^6)$. Again, the number of subtour constraints (13) may be exponential, so they are only added when necessary in an incremental fashion.

2.3 Enhancing the Integer Programs

Given the considerable size of the described IP formulations, we developed a number of enhancements to improve efficiency. For points on the **convex hull**, only a reduced number of neighbors need to be considered. Employing good **initial solutions** improves the performance in branch-and-bound searching; we used a number of greedy heuristics, as well as the $\frac{1}{2}$-approximation of Fekete. The large number of corresponding inequalities made it particularly important to deal with **intersections** in an efficient manner: we condensed the constraints for cliques of intersecting objects into single inequalities, and introduced special *halfspace inequalities* for the triangle-based approach. Further increases in efficiency were obtained by careful choices of how to **branch on variables** and careful maintenance of **subtour constraints**.

3 Experiments

Based on the described approaches, we ran experiments on some machines with some specifications and parameters. We used CPLEX on an Intel(R) Core(TM) i7-6700K CPU 4.00GHz with four cores and 8 threads utilizing an L3 Cache with 8MB. The solver was able to use a maximum amount of 64GB RAM.

3.1 Edge-Based Solvers

EdgeV1 is a basic integer program of the edge-based approach. It adds all intersection constraints before starting the solving process and adds subtour constraints in every integer solution. This integer program is an improvement to the edge-based MinArea integer program presented by Papenberg et al. [4,8]. In the former approach cycle based subtour constraints were added after an optimal solution has been found. This resulted in poor computing times even for small point sets. EdgeV2 extends the previous version by adding intersection constraints at interim solutions. Moreover, this version includes a branching extension where branching on a variable z_e results in intersecting edges getting branched to zero. We also utilize properties of the convex hull to exclude certain variables, i.e., edges that connect two non-adjacent points on the convex hull, from the computation. EdgeV2 makes use of this concept by setting these variables to zero. Fekete et al. [4] introduced the concept of a boundary index. Their results indicate small improvements in computation time when adding the constraints. EdgeBIV2 extends the previous version by adding boundary index constraints. The upcoming sections will show that the boundary index constraints will increase the computation time of our integer program. Because of this, we removed the concept in favor of version three. In EdgeV3 we additionally search for subtours in fractional interim solutions and add slab constraints during the solving process. Furthermore, we pass a start solution to the solver which was generated by an abstraction of the Greedy Min-Area heuristic of Taranilla et al. [11].

3.2 Triangle-Based Solvers

TriangulationV1 is the first version of the triangle-based approach. Compared to the basic triangulation approach of Papenberg [8], we have fewer variables and different subtour constraints (13). Similar to the edge-based approaches, we pass a start solution obtained from Greedy Min-Area as an input to the solver. We added further *halfspace inequalities* as well as equalities for edges which connect non-adjacent vertices of the convex hull.

In TriangulationV1 we add subtour constraints and intersection constraints in every integer solution. TriangulationV2 extends the first version with so-called *subtour angle constraints*. These are added at every integer solution. We are able to reuse the connected components we need to compute along the way. This allows us to add constraints (13) without much additional computation time. TriangulationV3 makes use of additional results on ineffective subtour constraints. In addition to the constraints of TriangulationV2, we add point-based subtour constraints to every intermediate integer solution.

3.3 Results for Minimization

As Figure 4a shows, our various enhancements result in a considerable reduction of the computation times, compared to the approach by Papenberg et al. [4, 8]. Furthermore, it turned out that the triangle-based approach was able to compute optimal solutions for larger instances, as shown in Figure 5.

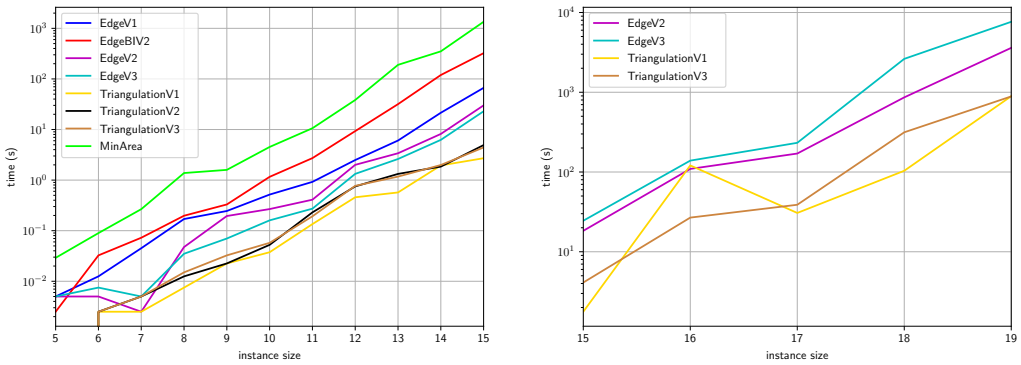


Figure 4 Computation times of Min-Area of the implemented solver versions. The computing time values are the average over 5 instances for each size. (a) Comparison with the MinArea version of Papenberg [8] for random instances of size $5 - 15$. MinArea operated on different instances than the rest. (b) Comparison of the best solver versions of both approaches for random instances of size $16 - 19$.

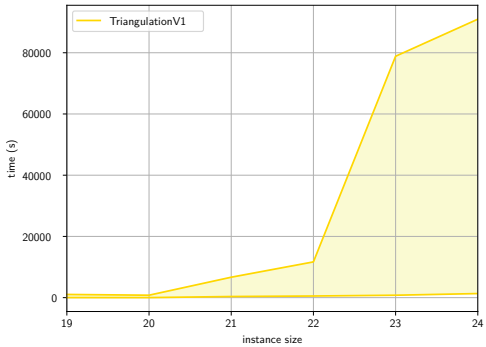


Figure 5 Computation time of TriangulationV1 of both approaches for random instances of size $19 - 24$. Shown are the minimum and maximum computation time needed to optimality.

3.4 Results for Maximization

For Max-Area, the edge-based approach turned out to be more useful: As Figure 6a shows, the runtime for the triangle-based solvers grew significantly faster. This seems to be mostly due to the fact that for the maximization version, intersections of "fat" intermediate subpolygons occur more frequently than for the "skinny" ones in the minimization version. Furthermore, we were able to expand the size of solvable instances in reasonable time to 23, as shown in Figure 7b.

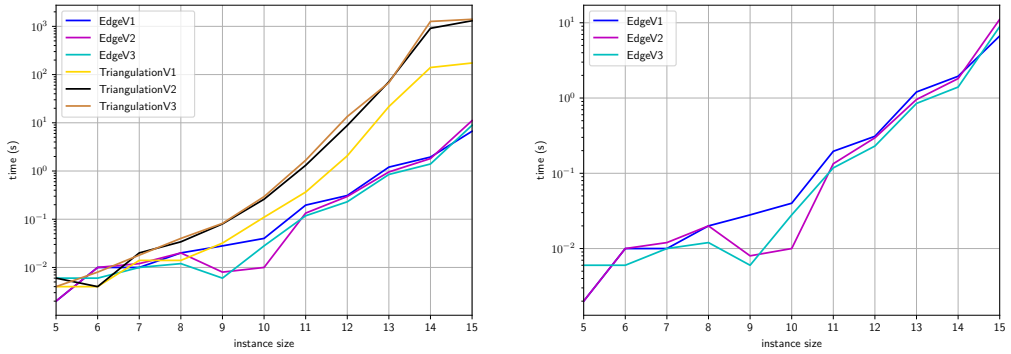


Figure 6 Computation times of all solver versions of Max-Area using both approaches for random instances of size $5 - 15$. The computing time values are the average over 10 instances for each size. (a) Comparison of solver version from both approaches. (b) Comparison of all edge-based solver versions

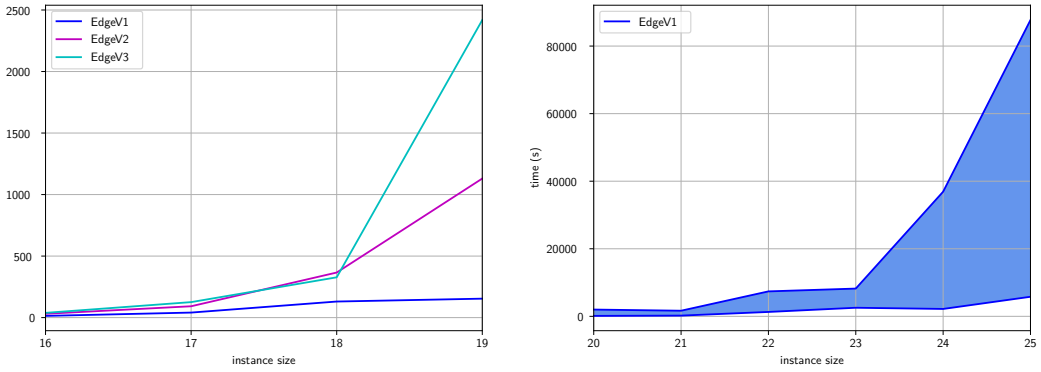


Figure 7 Computation time of Max-Area using different edge-based solver versions. (a) The computing time of all edge-based solver versions on random instances of size $16 - 19$. The values are the average over 5 instances for each size. (b) Range of computation time for EdgeV1 for random instances of size $20 - 25$.

4 Conclusions

While our work shows that with some amount of algorithm engineering, it is possible to extend the range of instances that can be solved to provable optimality, it also illustrates the practical difficulty of the problem. This reflects the limitations of such IP-based methods:

The edge-based approach makes use of an asymmetric variant of the TSP, which is known to be harder than the symmetric TSP, while the triangle-based approach suffers from its inherently large number of variables and constraints. Furthermore, the non-local nature of Min-Area and Max-Area polygons (which may contain edges that connect far-away points) makes it difficult to reduce the set of candidate edges.

As a result, Min-Area and Max-Area turn out to be prototypes of geometric optimization problems that are difficult both in theory and practice. This differs fundamentally from a problem such as Minimum Weight Triangulation, for which provably optimal solutions to huge point sets can be found [7], and practically difficult instances seem elusive [5].

References

1 Erik D. Demaine, Sándor P. Fekete, and Joseph S.B. Mitchell. The 2019 CG Challenge: Area-optimal polygonalizations. *Manuscript*, 2020.

2 Sándor P. Fekete. *Geometry and the Travelling Salesman Problem.* Ph.D. thesis, Department of Combinatorics and Optimization, University of Waterloo, Waterloo, ON, 1992.

3 Sándor P. Fekete. On simple polygonizations with optimal area. *Discrete & Computational Geometry*, 23(1):73–110, 2000.

4 Sándor P. Fekete, Stephan Friedrichs, Michael Hemmer, Melanie Papenberg, Arne Schmidt, and Julian Troegel. Area-and boundary-optimal polygonalization of planar point sets. In *European Workshop on Computational Geometry (EuroCG)*, pages 133–136, 2015.

5 Sándor P. Fekete, Andreas Haas, Dominik Krupke, Yannic Lieder, Eike Niehs, Michael Perk, Victoria Sack, and Christian Scheffer. Hard instances of the minimum-weight triangulation problem. Submitted to *European Workshop on Computational Geometry (EuroCG 2020).*

6 Sándor P. Fekete and William R. Pulleyblank. Area optimization of simple polygons. In *Symposium on Computational Geometry (SoCG)*, pages 173–182, 1993.

7 Andreas Haas. Solving large-scale minimum-weight triangulation instances to provable optimality. In *Symposium on Computational Geometry (SoCG)*, pages 44:1–44:14, 2018.

8 Melanie Papenberg. Exact Methods for area-optimal Polygons. Master's thesis, University of Technology Braunschweig, 2014.

9 Jiju Peethambaran, Amal Dev Parakkat, and Ramanathan Muthuganapathy. A randomized approach to volume constrained polyhedronization problem. *Journal of Computing and Information Science in Engineering*, 15(1):011009, 2015.

10 Jiju Peethambaran, Amal Dev Parakkat, and Ramanathan Muthuganapathy. An empirical study on randomized optimal area polygonization of planar point sets. *Journal of Experimental Algorithmics (JEA)*, 21:1–10, 2016.

11 Maria Teresa Taranilla, Edilma Olinda Gagliardi, and Gregorio Hernández Peñalver. Approaching minimum area polygonization. In *Congreso Argentino de Ciencias de la Computación (CACIC)*, pages 161–170, 2011.

Weighted ε-Nets

Daniel Bertschinger[1] and Patrick Schnider[2]

1 **Department of Computer Science, ETH Zürich, Switzerland**
`daniel.bertschinger@inf.ethz.ch`
2 **Department of Computer Science, ETH Zürich, Switzerland**
`patrick.schnider@inf.ethz.ch`

Abstract

Motivated by recent work of Bukh and Nivasch [4] on one-sided ε-approximants, we introduce the notion of *weighted ε-nets*. It is a geometric notion of approximation for point sets in $\mathbb{R}^d$ similar to ε-nets and ε-approximations, where it is stronger than the former and weaker than the latter. The main idea is that small sets can contain many points, whereas large sets must contain many points of the weighted ε-net.

In this paper, we analyze weak weighted ε-nets with respect to convex sets and axis-parallel boxes and give upper and lower bounds on ε for weighted ε-nets of size two and three. Some of these bounds apply to classical ε-nets as well.

1 Introduction

Representing large, complicated objects by smaller, simpler ones is a common theme in mathematics. For one-dimensional data sets this is realized by the notions of medians, means and quantiles. One fundamental difference between medians and quantiles on the one side and the mean on the other side is the robustness of the former against outliners of the data.

Centerpoint. Medians and quantiles are one-dimensional concepts, whereas modern data sets are often multidimensional. Hence, many generalizations of medians and quantiles to higher dimensions have been introduced and studied. One example is the notion of a centerpoint, that is, a point c such that for every closed halfspace h containing c we know that h contains at least a $\frac{1}{d+1}$-fraction of the whole data, where d denotes the dimension. The Centerpoint Theorem ensures that for any point set in $\mathbb{R}^d$ there always exists such a centerpoint [11].

Instead of representing a data set by a single point, one could take a different point set as a representative. This is exactly the idea of an *ε-net*.

▶ **Definition 1.1.** Given any range space $(X, \mathcal{R})$, an *ε-net* on a point set $P \subseteq X$ is a subset $N \subseteq P$ such that every $R \in \mathcal{R}$ with $|R \cap P| \geq \varepsilon |P|$ has nonempty intersection with N. If the condition that an ε-net needs to be a subset of P is dropped, then N is called a *weak ε-net*.

In this language, a centerpoint is a weak $\frac{d}{d+1}$-net for the range space of halfspaces. The concept of ε-nets has been studied in a huge variety; first, there are statements on the existence and the size of ε-nets, if ε is given beforehand. On the other hand, one can fix the size of the ε-net a priori and try to bound the range of ε in which there always exists an ε-net. For the former, it is known that every range space of VC-dimension δ has an ε-net of size at most $\mathcal{O}(\frac{\delta}{\varepsilon} \log \frac{1}{\varepsilon})$ [7].

ε-Approximations. For some applications though, ε-nets may not retain enough information. For every range we only know that it has a nonempty intersection with the net; however, we do not know anything about the size of this intersection. Hence, the following definition of ε-approximations comes naturally.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

▶ **Definition 1.2.** Given any range space $(X, \mathcal{R})$ and any parameter $0 \leq \varepsilon \leq 1$, an *$\varepsilon$-approximation* on a point set $P \subset X$ is a subset $A \subset P$ such that for every $R \in \mathcal{R}$ we have $\left|\frac{|R \cap P|}{|P|} - \frac{|R \cap A|}{|A|}\right| \leq \varepsilon$.

Initiated by the work of Vapnik and Chervonenkis [13], one general idea is to construct ε-approximations by uniformly sampling a random subset $A \subseteq X$ of large enough size. This results in statements about the existence of ε-approximations depending on the VC-dimension of the range space. In particular every range space of VC-dimension δ allows an ε-approximation of size $\mathcal{O}(\frac{\delta}{\varepsilon^2} \log \frac{1}{\varepsilon})$ [5, 6, 8].

Convex Sets. It is well-known that the range space of convex sets has unbounded VC-dimension; therefore, none of the results mentioned above can be applied. While constant size weak ε-nets still exist for the range space of convex sets [1, 12], the same cannot be said for weak ε-approximations (Proposition 1 in [4]). Motivated by this, Bukh and Nivasch [4] introduced the notion of *one-sided weak ε-approximants*. The main idea is that small sets can contain many points, whereas large sets must contain many points of the approximation. Bukh and Nivasch show that constant size one-sided weak ε-approximants exist for the range space of convex sets. In this work, we define a similar concept, called *weighted ε-nets*. In contrast to one-sided weak ε-approximants, our focus is to understand what bounds can be achieved for a fixed small value of k, which is given a priori. In this sense our approach is similar to the one taken by Aronov et al. [2] (for standard ε-nets).

▶ **Definition 1.3.** Given any point set $P \subset \mathbb{R}^d$ of size n, a *weighted ε-net of size k* (with respect to some range space) is defined as a set of points $p_1, \ldots, p_k$ and some values $\varepsilon = (\varepsilon_1, \ldots, \varepsilon_k)$ such that every set in the range space containing more than $\varepsilon_i n$ points of P contains at least i of the points $p_1, \ldots, p_k$.

Following historic conventions, we denote a weighted ε-net as *strong* if $p_1, \ldots, p_k \in P$ and as *weak* otherwise. In this work, we focus on weak weighted ε-nets of small size for the range space of convex sets and axis-parallel boxes.

2 Weighted ε-nets for the range space of convex sets

Weighted ε-nets for the range space of halfspaces were already studied by Pilz and Schnider [10]. In this section we generalize one of their results to the range space of convex sets.

▶ **Theorem 2.1.** *Let P be a set of n points in general position in $\mathbb{R}^d$. Let $0 < \varepsilon_1 \leq \varepsilon_2 < 1$ be arbitrary constants with (i) $d\varepsilon_1 + \varepsilon_2 \geq d$ and (ii) $\varepsilon_1 \geq \frac{2d-1}{2d+1}$. Then there are two points p_1 and p_2 in $\mathbb{R}^d$ such that*

1. *every convex set containing more than $\varepsilon_1 n$ points of P contains at least one of the points p_1 or p_2, and*
2. *every convex set containing more than $\varepsilon_2 n$ points of P contains both p_1 and p_2.*

In the following we briefly sketch the proof. For a full proof, we refer the interested reader to the full version of this paper [3].

Sketch of Proof. The main idea is to create two classes $\mathcal{A}$ and $\mathcal{B}$, containing convex subsets of $\mathbb{R}^d$. We put every convex subset of $\mathbb{R}^d$ containing more than $\varepsilon_2 n$ points of P (denoted as *big sets*) into both classes $\mathcal{A}$ and $\mathcal{B}$. Further, we put every convex subset of $\mathbb{R}^d$ containing more than $\varepsilon_1 n$ points of P (called the *small sets*) into one of the classes $\mathcal{A}$ or $\mathcal{B}$. To this end we halve P with a $(d-1)$-dimensional hyperplane H. Every small set containing more

points of P below H than above H is put into $\mathcal{B}$. Every small set which is not in $\mathcal{B}$, is put into $\mathcal{A}$. It can now be shown that $\mathcal{A}$ as well as $\mathcal{B}$ satisfies the Helly property. We then define p_1 and p_2 as the two Helly points. ◀

3 Lower Bounds on ε

Having seen an existential result for weighted ε-nets with respect to convex sets, we are interested in the best possible value for ε. In this chapter we present some lower bounds on ε. First, an example given in [10] can be adapted to show that inequality (i) of Theorem 2.1 is needed in the following sense: In the plane we cannot simultaneously have $\varepsilon_1 > \frac{3}{5}$ and $\varepsilon_2 > \frac{4}{5}$. To see this, consider the point set in Figure 1. Note that one of the two points needs to lie in $l^+_{a,d} \cap l^-_{a,d}$. The same is true for all intersections depicted in the right part of Figure 1. However, these five intersections cannot be stabbed using only two points.

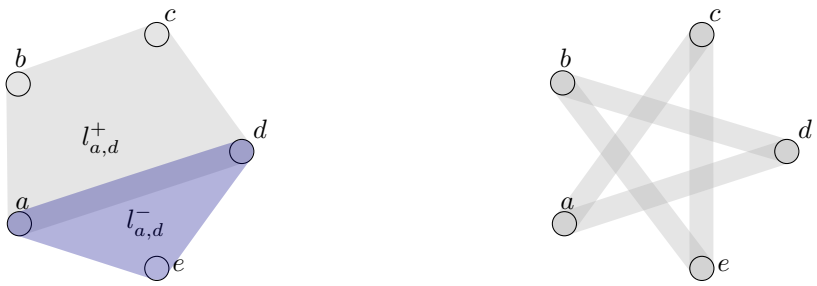


Figure 1 A point set of five regions in convex position, each containing exactly k points. Two particular regions containing four (three, respectively) of the regions. The intersections of interest are drawn on the right side.

3.1 Lower bounds on ε_1

On the other hand, one can give lower bounds on ε_1, independently of the value of ε_2. This setting is exactly the same as giving lower bounds on ε for any ε-net. Hence, any bound given in this chapter is also a lower bound on ε for ε-nets as well. Mustafa and Ray [9] have studied this in dimension 2, showing that there exist point sets P in $\mathbb{R}^2$ such that for every two points p_1 and p_2, not necessarily in P, we can find a convex set containing at least $\frac{4n}{7}$ points of P but neither p_1 nor p_2.

For higher dimension, to our knowledge the bounds given here are among the first and currently the best lower bounds for the range space of convex sets.

▶ **Lemma 3.1.** *There are point sets $P \subset \mathbb{R}^3$, such that for any two points p_1 and p_2 in $\mathbb{R}^3$ we can always find a compact convex set containing at least $\frac{5n}{8}$ points of P, but neither p_1 nor p_2.*

Sketch of Proof. Consider a point set in three dimensions consisting of eight points. There is a hexagon in the xy-plane, one point above the hexagon (denoted as u_1), and one point below the hexagon (denoted as u_2), see Figure 2.

It can be observed that every set of four points of the hexagon and every set of three points together with the center of the hexagon (indicated by the cross) should contain one of p_1 and p_2 for the Lemma to be wrong. However, it is not possible to place p_1 and p_2 accordingly. For more detail we again refer to [3]. ◀

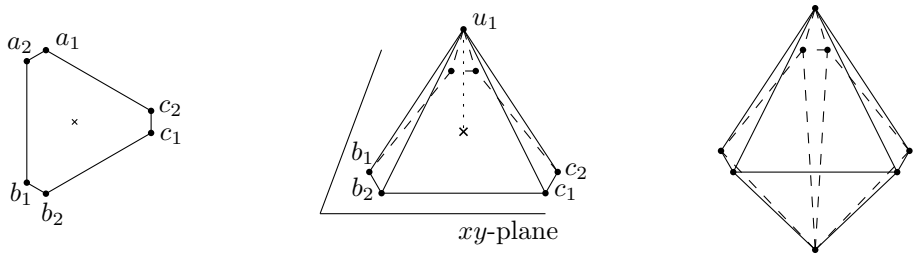


Figure 2 A point set in three dimensions, with six points in the xy-plane arranged in a hexagon, one point above the xy-plane and one point below the xy-plane.

For general dimensions a lower bound on ε_1 is given in the following Lemma. The corresponding examples for the proof consist of a $(d-1)$-dimensional simplex $\mathcal{S}$ in $\mathbb{R}^d$ with exactly one point in every 0-face of the simplex. There are two additional points, one *above* and one *below* the simplex, where above and below refer to the dimension not used for $\mathcal{S}$. A detailed discussion of the arguments can be found in [3].

▶ **Lemma 3.2.** *There are point sets P in $\mathbb{R}^d$ such that for any two points $p_1, p_2 \in \mathbb{R}^d$ there is a compact convex set containing $\frac{d}{d+2}$ of the points of P, but neither p_1 nor p_2.*

4 The range space of axis-parallel boxes

In this section, we study weighted ε-nets of size 2 and 3 for the range space of axis-parallel boxes. Axis-parallel boxes have the property that they allow a much stronger Helly-type result.

▶ **Observation 4.1.** *Let $\mathcal{F}$ be a family of compact, axis-parallel boxes in $\mathbb{R}^d$ such that any two of them have a common intersection. Then the whole collection has a nonempty intersection.*

As a direct consequence of this observation we note that for any point set P in $\mathbb{R}^d$, there always exists a (weighted) $\frac{1}{2}$-net of size 1 for the range space of axis-parallel boxes. For weighted ε-nets of larger size we find the following.

▶ **Theorem 4.2.** *Let P be a set of n points in general position in $\mathbb{R}^d$. Let $0 < \varepsilon_1 \leq \varepsilon_2 < 1$ be arbitrary constants with (i) $\varepsilon_1 \geq \frac{3^{d-1}}{2 \cdot 3^{d-1}+1}$ and (ii) $\varepsilon_1 + \varepsilon_2 \geq 1$. Then there exist two points p_1 and p_2 such that*

1. *every axis-parallel box containing more than $\varepsilon_1 n$ points of P contains at least one of the points p_1 and p_2, and*
2. *every axis-parallel box containing more than $\varepsilon_2 n$ points of P contains both, p_1 and p_2.*

Sketch of Proof. For the sake of simplicity, we only present a proof in $\mathbb{R}^2$ with fixed values $\varepsilon_1 = \frac{3}{7}$ and $\varepsilon_2 = \frac{4}{7}$. For other values the proof works analogously. First, divide the point set with a horizontal line l_1, such that there are $\frac{3n}{7}$ points below l_1 and $\frac{4n}{7}$ points above l_1. Then add two lines l', l'' perpendicular to l_1 splitting the point set below l_1 into three parts containing the same number of points, see Figure 3 (left).

Now one of the two outside areas above l_1, without loss of generality B_1, contains at most $\frac{2n}{7}$ points of P. We then move l' slightly towards l'', until we have the same number of points in B_1' as in A'. We now define $p_1 := l_1 \cap l_2$.

As the area left of l_2 and the area below l_1 contain $\frac{3n}{7}$ of the points of P every big box contains p_1 for sure. On the other hand every small box not containing p_1 lies completely

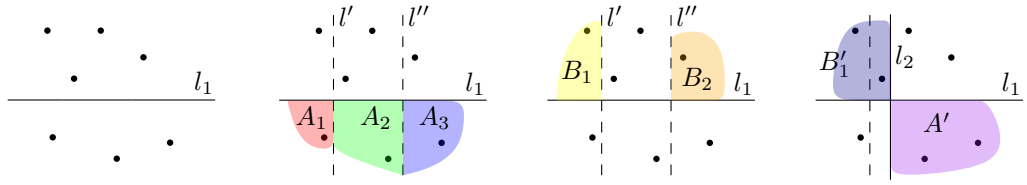


Figure 3 An example of the construction of p_1. First the point set P is split by a line l_1. Then the lines l' and l'' split the point set below l_1 into three disjoint parts containing the same number of points, namely A_1, A_2 and A_3. One of B_1 and B_2 has to contain "few" points of P, without loss of generality B_1, and by slightly changing l' we can ensure that B_1' and A' contain the same number of points of P. The resulting lines define $p_1 := l_1 \cap l_2$.

above l_1 or completely right of l_2. By a simple counting argument, any two small boxes not containing p_1 intersect. Any small box intersects any big box as a consequence of inequality (i); hence, applying Observation 4.1 we find p_2 satisfying the conditions of the Theorem.

For higher dimensions, we use hyperplanes instead of lines and we repeat the second step $d-1$ times (once in every direction except the first). ◀

A similar spitting idea works for weighted ε-nets of size 3: Let l_1 be a horizontal halving line and let l_2 be a vertical halving line. Let A and B be the areas above and below l_1 and let L and R be the areas left and right of l_2. The lines define four quadrants, where two opposite ones, say $L \cap A$ and $R \cap B$, both contain at least $\frac{n}{4}$ points of P. Define $p_1 := l_1 \cap l_2$. For every relevant box $\square$, assign $\square$ to the area $X \in \{A, B, L, R\}$ for which $|\square \cap X|$ is maximized. Put every box assigned to A and L into $\mathcal{A}$ and every box assigned to B and U into $\mathcal{B}$. Choosing the right values for $\varepsilon_1, \varepsilon_2$ and ε_3, we can apply Observation 4.1 to $\mathcal{A}$ and $\mathcal{B}$ to get the following:

▶ **Theorem 4.3.** *Let P be a set of n points in the plane. Let $0 < \varepsilon_1 \leq \varepsilon_2 \leq \varepsilon_3 < 1$ be arbitrary constants with (i) $\varepsilon_1 \geq \frac{3}{8}$, (ii) $\varepsilon_2 \geq \frac{1}{2}$, and (iii) $\varepsilon_1 + \varepsilon_3 \geq 1$. Then there exist three points p_1, p_2 and p_3 in $\mathbb{R}^2$ such that every axis-parallel box containing more than $\varepsilon_i n$ points of P contains at least i of the points p_1, p_2 and p_3.*

5 Conclusion

We have given bounds for weak weighted ε-nets of size 2 for convex sets and axis-parallel boxes. It remains an interesting question to find bounds for larger sizes. For axis-parallel boxes, we gave a construction for weighted ε-nets of size 3 in the plane. Unfortunately our construction does not generalize to higher dimensions. It is a natural question whether a similar statement in higher dimensions can be shown using a different construction.

References

1. N. Alon, I. Bárány, Z. Füredi, and D. J. Kleitman. Point Selections and Weak ε-Nets for Convex Hulls. *Combinatorics, Probability and Computing*, 1(3):189–200, 1992. URL: https://doi.org/10.1017/S0963548300000225.
2. B. Aronov, F. Aurenhammer, F. Hurtado, S. Langerman, D. Rappaport, C. Seara, and S. Smorodinsky. Small weak epsilon-nets. *Computational Geometry*, 42(5):455 – 462, 2009. URL: https://doi.org/10.1016/j.comgeo.2008.02.005.
3. Daniel Bertschinger and Patrick Schnider. Weighted epsilon-nets, 2020. arXiv:2002.08693.
4. B. Bukh and G. Nivasch. One-Sided Epsilon-Approximants. In *A Journey Through Discrete Mathematics: A Tribute to Jiří Matoušek*, pages 343–356. Springer, 2017. URL: https://doi.org/10.1007/978-3-319-44479-6_12.
5. S. Har-Peled. *Geometric Approximation Algorithms*. American Mathematical Society, 2011. URL: https://dlnext.acm.org/doi/10.5555/2031416.
6. S. Har-Peled and M. Sharir. Relative (p,ε)-Approximations in Geometry. *Discrete & Computational Geometry*, 45(3):462–496, 2011. URL: https://doi.org/10.1007/s00454-010-9248-1.
7. D. Haussler and E. Welzl. ε-nets and simplex range queries. *Discrete & Computational Geometry*, 2(2):127–151, 1987. URL: https://doi.org/10.1007/BF02187876.
8. J. Matoušek, E. Welzl, and L. Wernisch. Discrepancy and approximations for bounded VC-dimension. *Combinatorica*, 13(4):455–466, 1993. URL: https://doi.org/10.1007/BF01303517.
9. N. H. Mustafa and S. Ray. An optimal extension of the centerpoint theorem. *Computational Geometry*, 42(6):505 – 510, 2009. URL: https://doi.org/10.1016/j.comgeo.2007.10.004.
10. A. Pilz and P. Schnider. Extending the Centerpoint Theorem to Multiple Points. In *29th International Symposium on Algorithms and Computation (ISAAC 2018)*, volume 123, pages 53:1–53:13, 2018. URL: http://drops.dagstuhl.de/opus/volltexte/2018/10001.
11. R. Rado. A Theorem on General Measure. *Journal of the London Mathematical Society*, s1-21(4):291–300, 10 1946. URL: https://dx.doi.org/10.1112/jlms/s1-21.4.291.
12. N. Rubin. An Improved Bound for Weak Epsilon-Nets in the Plane. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, 2018. URL: https://doi.org/10.1109/FOCS.2018.00030.
13. V. Vapnik and A. Chervonenkis. On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities. *Theory of Probability & Its Applications*, 16(2):264–280, 1971. URL: https://doi.org/10.1137/1116025.

Homotopic Curve Shortening and the Affine Curve-Shortening Flow*

Sergey Avvakumov[1] and Gabriel Nivasch[2]

1 **Institute of Science and Technology Austria (IST Austria), Am Campus 1, 3400 Klosterneuburg, Austria**
sergey.avvakumov@ist.ac.at

2 **Ariel University, Ariel, Israel**
gabrieln@ariel.ac.il

Abstract

We define and study a discrete process that generalizes the convex-layer decomposition of a planar point set. Our process, which we call *homotopic curve shortening* (HCS), starts with a closed curve (which might self-intersect) in the presence of a set $P \subset \mathbb{R}^2$ of point obstacles, and evolves in discrete steps, where each step consists of (1) taking shortcuts around the obstacles, and (2) reducing the curve to its shortest homotopic equivalent.

We find experimentally that, if the initial curve is held fixed and P is chosen to be either a very fine regular grid or a uniformly random point set, then HCS behaves at the limit like the affine curve-shortening flow (ACSF). This connection between HCS and ACSF generalizes the link between "grid peeling" and the ACSF observed by Eppstein et al. (2017), which applied only to convex curves, and which was studied only for regular grids.

We prove that HCS satisfies some properties analogous to those of ACSF: HCS is invariant under affine transformations, preserves convexity, and does not increase the total absolute curvature. Furthermore, the number of self-intersections of a curve, or intersections between two curves (appropriately defined), does not increase. Finally, if the initial curve is simple, then the number of inflection points (appropriately defined) does not increase.

1 Introduction

Let $\mathbb{S}^1$ be the unit circle. In this paper we call a piecewise-smooth function $\gamma : [0, 1] \to \mathbb{R}^2$ a *path*, and a piecewise-smooth function $\gamma : \mathbb{S}^1 \to \mathbb{R}^2$ a *closed curve*, or simply a *curve*. If γ is injective then the curve or path is said to be *simple*. We say that two paths or curves γ, δ are ε*-close* to each other if their Fréchet distance is at most ε, i.e. if they can be re-parametrized such that for every t, the Euclidean distance between the points $\gamma(t)$, $\delta(t)$ is at most ε.

1.1 Shortest Homotopic Curves

Let P be a finite set of points in the plane, which we regard as obstacles. Two curves γ, δ that avoid P are said to be *homotopic* if there exists a way to continuously transform γ into δ while avoiding P at all times. And two paths γ, δ that avoid P (except possibly at the endpoints) and satisfy $\gamma(0) = \delta(0)$, $\gamma(1) = \delta(1)$ are said to be *homotopic* if there exists a way to continuously transform γ into δ, without moving their endpoints, while avoiding P at all times (except possibly at the endpoints). We extend these definitions to the case where γ avoids obstacles but δ does not, by requiring the continuous transformation of γ into δ to avoid obstacles at all times except possibly at the last moment.

* Sergey Avvakumov was supported by the Austrian Science Fund (FWF), Project P31312-N35.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

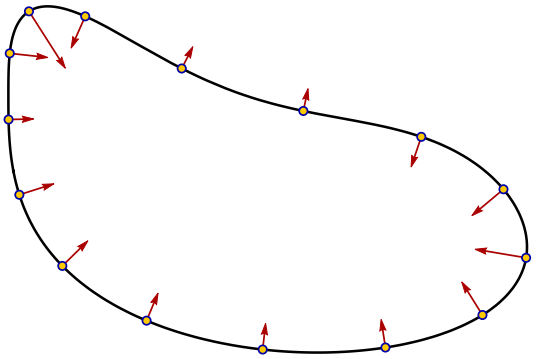


Figure 1 ACSF. Arrows indicate instantaneous velocity of points at the shown moment.

Then, for every curve (resp. path) γ in the presence of obstacles there exists a unique shortest curve (resp. path) δ that is homotopic to γ. The problem of computing the shortest path or curve homotopic to a given piecewise-linear path or curve, under the presence of polygonal or point obstacles, has been studied extensively [6, 8, 11, 17, 25, 26].

1.2 The Affine Curve-Shortening Flow

In the *affine curve-shortening flow*, a smooth curve $\gamma \subset \mathbb{R}^2$ varies with time in the following way. At each moment in time, each point of γ moves perpendicularly to the curve, towards its local center of curvature, with instantaneous velocity $r^{-1/3}$, where r is that point's radius of curvature at that time. See Figure 1.

The ACSF was first studied by Alvarez et al. [2] and Sapiro and Tannenbaum [27]. It differs from the more usual *curve-shortening flow* (CSF) [9, 13], in which each point is given instantaneous velocity r^{-1}. Unlike the CSF, the ACSF is invariant under affine transformations. It has applications in computer vision [9].

Under either the CSF or the ACSF, a simple curve remains simple, and its length decreases strictly with time ([13], [27], resp.). Furthermore, a pair of disjoint curves, run simultaneously, remain disjoint at all times ([28], [4], resp.). More generally, the number of intersections between two curves never increases ([3], [4], resp.). The total absolute curvature of a curve decreases strictly with time and tends to 2π ([20, 21], [4], resp.). The number of inflection points of a simple curve does not increase with time ([3], [4], resp.). Under the CSF, a simple curve eventually becomes convex and then converges to a circle as it collapses to a point [20, 21]. Correspondingly, under the ACSF, a simple curve becomes convex and then converges to an ellipse as it collapses to a point [4].

When the initial curve is not simple, a self-intersection might collapse and form a singularity that lasts for an instant. Unfortunately, unlike for the case of the CSF, for the ACSF no rigorous results have been obtained for self-intersecting curves [4]. Still, ACSF computer simulations can be run on curves that have self-intersections or singularities with little difficulty.

1.3 Relation to Grid Peeling

Let P be a finite set of points in the plane. The *convex-layer decomposition* (also called the *onion decomposition*) of P is the partition of P into sets $P_1, P_2, P_3, \ldots$ obtained as follows: Let $Q_0 = P$. Then, for each $i \geq 1$ for which $Q_{i-1} \neq \emptyset$, let P_i be the set of vertices of the

convex hull of Q_{i-1}, and let $Q_i = Q_{i-1} \setminus P_i$. In other words, we repeatedly remove from P the set of vertices of its convex hull. See [5, 12, 15, 16].

Eppstein et al. [18], following Har-Peled and Lidický [23], studied *grid peeling*, which is the convex-layer decomposition of subsets of the integer grid $\mathbb{Z}^2$. Eppstein et al. found an experimental connection between ACSF for convex curves and grid peeling. Specifically, let γ be a fixed convex curve. Let n be large, let $(\mathbb{Z}/n)^2$ be the uniform grid with spacing $1/n$, and let $P_n(\gamma)$ be the set of points of $(\mathbb{Z}/n)^2$ that are contained in the region bounded by γ. Then, as $n \to \infty$, the convex-layer decomposition of $P_n(\gamma)$ seems experimentally to converge to the ACSF evolution of γ, after the time scale is adjusted appropriately. They raised the question whether there is a way to generalize the grid peeling process so as to approximate ACSF for non-convex curves as well.

1.4 Our Contribution

In this paper we describe a generalization of the convex-layer decomposition to non-convex, and even non-simple, curves. We call our process *homotopic curve shortening*, or HCS. Under HCS, an initial curve evolves in discrete steps in the presence of point obstacles. We find that, if the obstacles form a uniform grid, then HCS shares the same experimental connection to ACSF that grid peeling does. Hence, HCS is the desired generalization sought by Eppstein et al. [18]. We also find that the same experimental connection between ACSF and HCS (and in particular, between ACSF and the convex-layer decomposition) holds when the obstacles are distributed uniformly at random, with the sole difference being in the constant of proportionality.

Although the experimental connection between HCS and ACSF seems hard to prove, we do prove that HCS satisfies some simple properties analogous to those of ACSF: HCS is invariant under affine transformations, preserves convexity, and does not increase the total absolute curvature. Furthermore, the number of self-intersections of a curve, or intersections between two curves (appropriately defined), does not increase. Finally, if the initial curve is simple, then the number of inflection points (appropriately defined) does not increase.

2 Homotopic Curve Shortening

Let P be a finite set of obstacle points. A *P-curve* (resp. *P-path*) is a curve (resp. path) that is composed of straight-line segments, where each segment starts and ends at obstacle points.

Homotopic curve shortening (HCS) is a discrete process that starts with an initial P-curve γ_0 (which might self-intersect), and at each step, the current P-curve γ_n is turned into a new P-curve $\gamma_{n+1} = \mathrm{HCS}_P(\gamma_n)$.

The definition of $\gamma' = \mathrm{HCS}_P(\gamma)$ for a given P-curve γ is as follows. Let $(p_0, \ldots, p_{m-1})$ be the circular list of obstacle points visited by γ. Call p_i *nailed* if γ goes straight through p_i, i.e. if $\angle p_{i-1} p_i p_{i+1} = \pi$.[1] Let $(q_0, \ldots, q_{k-1})$ be the circular list of nailed vertices of γ. Suppose first that $k \geq 1$. Then γ' is obtained through the following three substeps:

1. *Splitting.* We split γ into k P-paths $\delta_0, \ldots, \delta_{k-1}$ at the nailed vertices, where each δ_i goes from q_i to q_{i+1}.
2. *Shortcutting.* For each non-endpoint vertex p_i of each δ_i, we make the curve avoid p_i by taking a small shortcut. Specifically, let $\varepsilon > 0$ be sufficiently small, and let C_{p_i} be a

[1] All indices in circular sequences are modulo the length of the sequence.

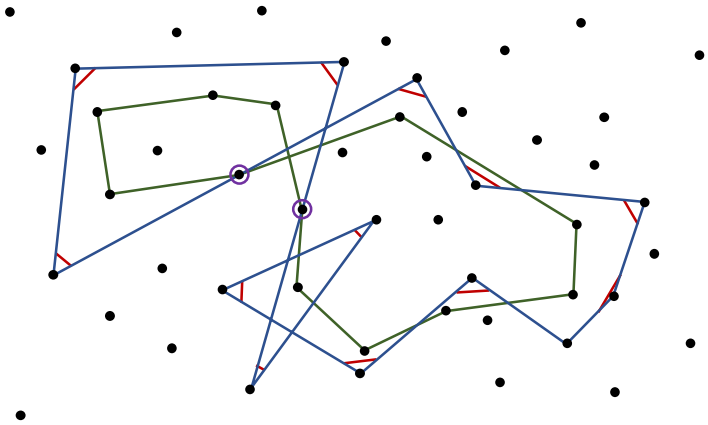


Figure 2 Computation of a single step of HCS: Given a P-curve γ (blue), we first identify its nailed vertices (purple). In this case, the two nailed vertices split γ into two paths δ_0, δ_1. In each δ_i we take a small shortcut around each intermediate vertex (red). Then we replace each δ_i by the shortest path homotopic to it, obtaining the new P-curve $\gamma' = \mathrm{HCS}_P(\gamma)$ (green).

circle of radius ε centered at p_i. Let e_i be the segment $p_{i-1}p_i$ of δ_i. Let $x_i = e_i \cap C_{p_i}$ and $y_i = e_{i+1} \cap C_{p_i}$. Then we make the path go straight from x_i to y_i instead of through p_i. Call the resulting path ρ_i, and let ρ be the curve obtained by concatenating all the paths ρ_i.

3. *Shortening.* Each ρ_i in ρ is replaced by the shortest P-path homotopic to it. The resulting curve is γ'.

If γ has no nailed vertices ($k = 0$) then γ' is obtained by performing the shortcutting and shortening steps on the single closed curve γ. Figure 2 illustrates one HCS step on a sample curve.

The process terminates when the curve collapses to a point. If the initial curve γ_0 is the boundary of the convex hull of P, then the HCS evolution of γ_0 is equivalent to the convex-layer decomposition of P.

3 Experimental Connection Between ACSF and HCS

Our experiments on a variety of curves show that HCS, using $P = (\mathbb{Z}/n)^2$ as the obstacle set, approximates ACSF at the limit as $n \to \infty$, just as grid peeling approximates ACSF for convex curves. Furthermore, we find that the connection between ACSF and HCS also holds if the uniform grid $(\mathbb{Z}/n)^2$ is replaced by a random point set, though with a different constant of time proportionality. See Figure 3 for an example.

4 Properties of Homotopic Curve Shortening

We prove that HCS satisfies some properties analogous to those of ACSF.

▶ **Theorem 4.1.** *HCS is invariant under affine transformations.*

▶ **Theorem 4.2.** *Under HCS, once a curve becomes the boundary of a convex polygon, it stays that way.*

The *total absolute curvature* of a piecewise-linear curve γ with vertices $(p_0, \ldots, p_{m-1})$ is the sum of the exterior angles $\sum_{i=0}^{m-1}(\pi - |\angle p_{i-1}p_ip_{i+1}|)$.

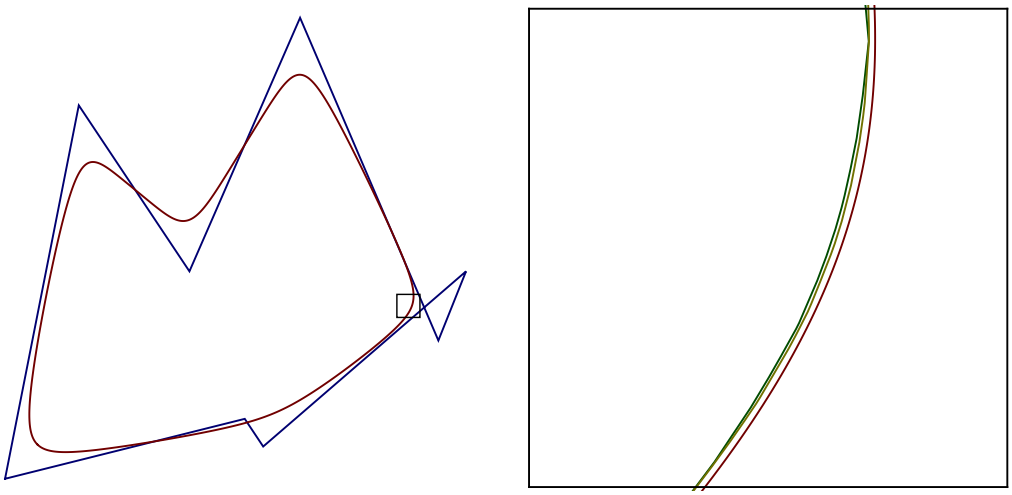


Figure 3 Left: Initial curve (blue) and simulated ACSF result after the curve's length reduced to 70% of its original length (red). Right: Comparison between ACSF approximation (red), HCS with $n = 10^7$ uniform-grid obstacles (green), and HCS with $n = 10^7$ random obstacles (yellow) on a small portion of the curve.

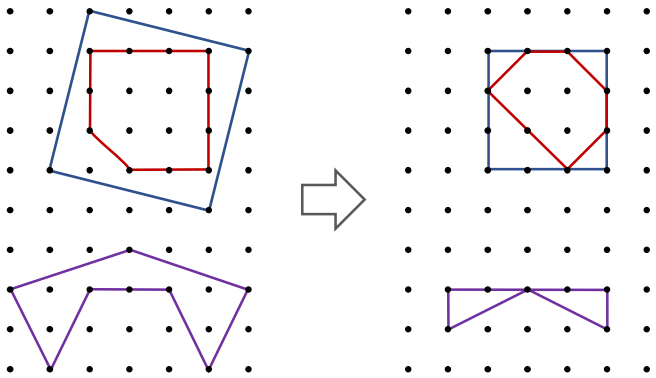


Figure 4 HCS might cause disjoint curves to intersect, or a simple curve to self-intersect.

▶ **Theorem 4.3.** *Under HCS, the total absolute curvature of a curve never increases.*

If γ, δ are disjoint P-curves, then $\mathrm{HCS}_P(\gamma), \mathrm{HCS}_P(\delta)$ are not necessarily disjoint. Similarly, if γ is a simple P-curve, then $\mathrm{HCS}_P(\gamma)$ is not necessarily simple. See Figure 4.

Curves γ, δ are called *disjoinable* if they can be made into disjoint curves by peforming on them an arbitrarily small perturbation. Similarly, a curve γ is called *self-disjoinable* if it can be turned into a simple curve by an arbitrarily small perturbation. (Akitaya et al. [1] recently found an $O(n \log n)$-time algorithm that can decide, in particular, whether a given curve is self-disjoinable.)

An intersection between two curves, or between two portions of one curve, is called *transversal*, if at the point of intersection both curves are differentiable and their normal vectors are not parallel at that point. If all intersections between curves γ_1 and γ_2 are transversal, then we say that γ_1, γ_2 are themselves *transversal*. Similarly, if all self-intersections of γ are transversal, then we say that γ is *self-transversal*. (Transversal and self-transversal curves are sometimes called *generic*, see e.g. [10].)

If γ is self-transversal, we denote by $\chi(\gamma)$ the number of self-intersections of γ. If γ is not self-transversal, then we define $\chi(\gamma)$ as the minimum of $\chi(\widehat{\gamma})$ among all self-transversal curves $\widehat{\gamma}$ that are ε-close to γ, for all small enough $\varepsilon > 0$. Hence, $\chi(\gamma) = 0$ if and only if γ is self-disjoinable. We define similarly the number of intersections $\chi(\gamma_1, \gamma_2)$ between two curves. Then, γ_1 and γ_2 are disjoinable if and only if $\chi(\gamma_1, \gamma_2) = 0$. (Fulek and Tóth recently proved that the problem of computing $\chi(\gamma)$ is NP-hard [19].)

▶ **Theorem 4.4.** *Under HCS, the intersection and self-intersection numbers never increase.*

We say that an obstacle set P is in *general position* if no three points of P lie on a line. Note that if P is in general position then there are no nailed vertices in HCS.

▶ **Theorem 4.5.** *Under HCS with obstacles in general position, a simple curve stays simple, and a pair of disjoint curves stay disjoint.*

Let γ be a simple piecewise-linear curve with vertices $(v_0, \ldots, v_{n-1})$. An *inflection edge* of γ is an edge v_iv_{i+1} such that the previous and next vertices v_{i-1}, v_{i+2} lie on opposite sides of the line through v_i, v_{i+1}. Let $\varphi(\gamma)$ be the number of inflection edges of γ. Note that $\varphi(\gamma)$ is always even, since every inflection edge lies either after a sequence of clockwise vertices and before a sequence of counterclockwise vertices, or vice versa.

If γ is not simple but self-disjoinable, then we define $\varphi(\gamma)$ as the minimum of $\varphi(\gamma')$ over all simple piecewise-linear curves γ' that are ε-close to γ, for all sufficiently small $\varepsilon > 0$.

▶ **Theorem 4.6.** *Under HCS on a self-disjoinable curve, the curve's number of inflection edges never increases.*

5 Discussion

One of the reasons continuous curve-shortening flows were introduced and studied was to overcome the shortcomings of the *Birkhoff curve-shortening process* ([7], see also e.g. [14]), specifically the fact that it might cause the number of curve intersections to increase [22, 24]. As we have shown, HCS is a discrete process that overcomes this flaw without introducing analytical difficulties, at least in the plane. It would be interesting to check whether HCS can be applied on more general surfaces.

For more details see our full paper at [arXiv:1909.00263].

Acknowledgements. Thanks to Arseniy Akopyan, Imre Bárány, Jeff Erickson, Radoslav Fulek, Jeremy Schiff, Arkadiy Skopenkov, and Peter Synak for useful discussions. Thanks also to the referees for their useful comments.

References

1 Hugo A. Akitaya, Radoslav Fulek, and Csaba D. Tóth. Recognizing weak embeddings of graphs. In *Proc. 29th Symp. on Discrete Algorithms*, pages 274–292, 2018. doi:10.1137/1.9781611975031.20.

2 Luis Alvarez, Frédéric Guichard, Pierre-Luis Lions, and Jean-Michel Morel. Axioms and fundamental equations of image processing. *Arch. Rational Mech. Anal.*, 123(3):199–257, 1993. doi:10.1007/BF00375127.

3 Sigurd Angenent. Parabolic equations for curves on surfaces: Part II. Intersections, blow-up and generalized solutions. *Annals of Mathematics*, 133(1):171–215, 1991. doi:10.2307/2944327.

4 Sigurd Angenent, Guillermo Sapiro, and Allen Tannenbaum. On the affine heat equation for non-convex curves. *J. Amer. Math. Soc.*, 11(3):601–634, 1998. doi:10.1090/S0894-0347-98-00262-8.

5 Vic Barnett. The ordering of multivariate data. *J. Roy. Statist. Soc. Ser. A*, 139(3):318–355, 1976. doi:10.2307/2344839.

6 Sergei Bespamyatnikh. Computing homotopic shortest paths in the plane. *Journal of Algorithms*, 49(2):284–303, 2003. doi:https://doi.org/10.1016/S0196-6774(03)00090-7.

7 George D. Birkhoff. Dynamical systems with two degrees of freedom. *Trans. Amer. Math. Soc.*, 18:199–300, 1917.

8 Sergio Cabello, Yuanxin Liu, Andrea Mantler, and Jack Snoeyink. Testing homotopy for paths in the plane. *Discrete & Computational Geometry*, 31(1):61–81, 2004. doi:10.1007/s00454-003-2949-y.

9 Frédéric Cao. *Geometric Curve Evolution and Image Processing*, volume 1805 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 2003. doi:10.1007/b10404.

10 Hsien-Chih Chang and Jeff Erickson. Untangling planar curves. *Discrete & Computational Geometry*, 58:889–920, 2017. doi:10.1007/s00454-017-9907-6.

11 Bernard Chazelle. A theorem on polygon cutting with applications. In *Proc. 23rd Annual Symposium on Foundations of Computer Science (FOCS 1982)*, pages 339–349, 1982. doi:10.1109/SFCS.1982.58.

12 Bernard Chazelle. On the convex layers of a planar set. *IEEE Trans. Inform. Theory*, 31(4):509–517, 1985. doi:10.1109/TIT.1985.1057060.

13 Kai-Seng Chou and Xi-Ping Zhu. *The Curve Shortening Problem*. Chapman & Hall/CRC, Boca Raton, FL, 2001. doi:10.1201/9781420035704.

14 Cristopher B. Croke. Area and the length of the shortest closed geodesic. *J. Differential Geometry*, 27:1–21, 1988.

15 Ketan Dalal. Counting the onion. *Random Struct. Algor.*, 24(2):155–165, 2004. doi:10.1002/rsa.10114.

16 William F. Eddy. Convex Hull Peeling. In *COMPSTAT 1982 5th Symposium held at Toulouse 1982*, pages 42–47. Physica-Verlag, 1982. doi:10.1007/978-3-642-51461-6_4.

17 Alon Efrat, Stephen G. Kobourov, and Anna Lubiw. Computing homotopic shortest paths efficiently. *Computational Geometry*, 35(3):162–172, 2006. doi:https://doi.org/10.1016/j.comgeo.2006.03.003.

18 David Eppstein, Sariel Har-Peled, and Gabriel Nivasch. Grid peeling and the affine curve-shortening flow. *Experimental Mathematics*, page to appear, 2018. https://doi.org/10.1080/10586458.2018.1466379. doi:10.1080/10586458.2018.1466379.

19 Radoslav Fulek and Csaba D. Tóth. Crossing minimization in perturbed drawings. In T. Biedl and A. Kerren, editors, *Proc. 26th Symp. Graph Drawing and Network Visualization*, pages 229–241. Springer, 2018. doi:10.1007/978-3-030-04414-5_16.

20 Michael Gage and Richard S. Hamilton. The heat equation shrinking convex plane curves. *J. Differential Geom.*, 23(1):69–96, 1986. doi:10.4310/jdg/1214439902.

21 Matthew A. Grayson. The heat equation shrinks embedded plane curves to round points. *J. Differential Geom.*, 26(2):285–314, 1987. doi:10.4310/jdg/1214441371.

22 Matthew A. Grayson. Shortening embedded curves. *Annals of Mathematics*, 129(1):79–111, 1989.

23 Sariel Har-Peled and Bernard Lidický. Peeling the grid. *SIAM J. Discrete Math.*, 27(2):650–655, 2013. doi:10.1137/120892660.

24 Joel Hass and Peter Scott. Shortening curves on surfaces. *Topology*, 33:25–43, 1994. doi:10.1016/0040-9383(94)90033-7.

25 John Hershberger and Jack Snoeyink. Computing minimum length paths of a given homotopy class. *Computational Geometry*, 4(2):63–97, 1994. doi:https://doi.org/10.1016/0925-7721(94)90010-8.

26 Der-Tsai Lee and Franco P. Preparata. Euclidean shortest paths in the presence of rectilinear barriers. *Networks*, 14(3):393–410, 1984. doi:10.1002/net.3230140304.

27 Guillermo Sapiro and Allen Tannenbaum. Affine invariant scale-space. *Int. J. Comput. Vision*, 11(1):25–44, 1993. doi:10.1007/bf01420591.

28 Brian White. Evolution of curves and surfaces by mean curvature. In *Proceedings of the International Congress of Mathematicians, Vol. I (Beijing, 2002)*, pages 525–538, 2002. URL: https://www.mathunion.org/fileadmin/ICM/Proceedings/ICM2002.1/ICM2002.1.ocr.pdf.

Applications of Concatenation Arguments to Polyominoes and Polycubes*

Gill Barequet[1], Gil Ben-Shachar[1], and Martha Carolina Osegueda[2]

1 **Dept. of Computer Science**
The Technion—Israel Inst. of Technology
Haifa 3200003, Israel
{barequet,gilbe}@cs.technion.ac.il

2 **Dept. of Computer Science**
Univ. of California, Irvine, CA 92717.
mosegued@uci.edu

Abstract

We present several concatenation arguments for polyominoes and polycubes, and show their applications to setting lower and upper bounds on the growth constants of some of their families, whose enumerating sequences are pseudo sub- or super-multiplicative. *Inter alia*, we provide bounds on the growth constants of general and tree polyominoes, and general polycubes.

1 Introduction

A *polycube* of size n is a connected set of n cells on $\mathbb{Z}^d$, where connectivity is through $(d-1)$-dimensional facets. Two-dimensional polycubes are also called *polyominoes*. Two *fixed* polycubes are *equivalent* if one can be translated into the other. We consider only fixed polycubes, hence we simply call them "polycubes." The study of polycubes began in statistical physics [4, 14], where they are called *lattice animals*. Counting polyominoes and polycubes is a long-standing problem. Let $A_d(n)$ denote the number of d-dimensional polycubes of size (area) n. Values of $A_2(n)$ are currently known up to $n = 56$ [8]. The growth constant of polyominoes also attracted much attention. Klarner [9] showed that $\lambda_d := \lim_{n\to\infty} \sqrt[n]{A_d(n)}$ exists. The convergence of $\frac{A_d(n+1)}{A_d(n)}$ to λ_d $(n\to\infty)$ was proven much later [11]. The best known lower [1] and upper [10] bounds on λ_2 are 4.0025 and 4.6496, respectively.

In this paper, we develop methods for deriving bounds on the growth constants of families of polyominoes and polycubes, for which the enumerating sequences are pseudo sub- or super-multiplicative. Such a property can be derived from a generalized polyomino-concatenation argument, as we show below. We demonstrate various applications of this method to general polyominoes and polycubes, as well as to specific families, such as tree polycubes.

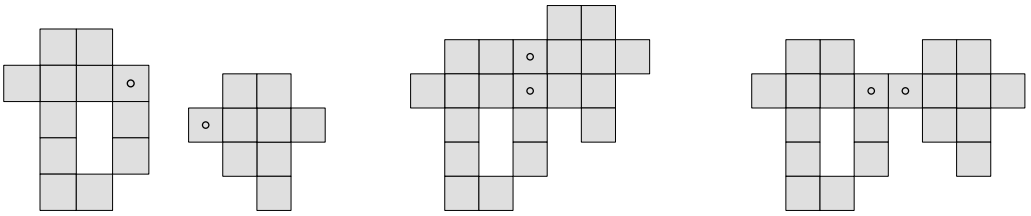
2 Preliminaries

2.1 Concatenation and Sub-/Super-multiplicative Sequences

A sequence $(Z(n))$ is *super-multiplicative* (resp., *sub-multiplicative*) if $Z(m)Z(n) \le Z(m+n)$ (resp., $Z(m)Z(n) \ge Z(m+n)$) $\forall m, n \in \mathbb{N}$. It is known [13, p. 171] that a super-multiplicative (resp., sub-multiplicative) sequence $Z(n)$, with the property that $Z'(n) = \sqrt[n]{Z(n)}$ is bounded from above (resp., below), has a *growth constant*. That is, the quantity $\lim_{n\to\infty} Z'(n)$ exists.

* Work on this paper by the first and second authors has been supported in part by ISF Grant 575/15. Work on this paper by the first author has also been supported in part by BSF Grant 2017684. Work on this paper by the third author has been supported in part by NSF Grant 1815073.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.



(a) Two polyominoes (b) Vertical concatenation (c) Horizontal concatenation

Figure 1 Concatenations of two polyominoes.

Let us define a total order on cells of the cubical lattice: First by x_1 (x in two dimensions), then by x_2 (y in two dimensions), and so on. Thus, in two dimensions, the *smallest* (resp., *largest*) square of a polyomino P is the lowest (resp., highest) cell in the leftmost (resp., rightmost) column of P. The vertical (resp., horizontal) *concatenation* of two polyominoes P_1, P_2 is the positioning of P_2 such that its smallest cell lies immediately *above* (resp., *to the right of*) the largest cell of P_1 (see Figure 1).

Similarly, two d-dimensional polycubes can be concatenated in d ways. Concatenating two polycubes always yields a valid polycube (connected and with no overlapping cells), and two different pairs of polycubes of sizes m, n always yield by concatenation two different polycubes of size $m+n$. Many polycubes, however, can be represented as the concatenations of several pairs of polycubes, whereas others cannot be represented at all as concatenations of smaller polycubes.

The following is a folklore concatenation argument for polyominoes, setting a rather weak lower bound on their growth constant. A direct consequence of the discussion above is that $A_2^2(n) < A_2(2n)$. That is, $\sqrt[n]{A_2(n)} < \sqrt[2n]{A_2(2n)}$. Hence, a sequence of the form $A^* = \left(\sqrt[n_0 2^i]{A_2(n_0 2^i)}\right)_{i=0}^{\infty}$ is monotone increasing for any natural number n_0. Since the entire sequence $A = (A_2(n))$ is super-multiplicative, and the sequence $A' = \left(\sqrt[n]{A_2(n)}\right)$ is bounded from above [6], the sequence A has a growth constant λ_2. Obviously, every subsequence of A' also converges to λ_2. In addition, since any such subsequence A^* is monotone increasing, any element of it, $\sqrt[n_0]{A_2(n_0)}$, is a lower bound on λ_2. Empirically, the best (largest) lower bound is obtained this way by setting $n_0 = 56$ (the largest value of n for which $A_2(n)$ is known), yielding the bound $\lambda_2 > 3.7031$.

Remarks. 1. Although A^* is a subsequence of A', the upward monotonicity of the former does *not* imply the monotonicity of the latter. Nevertheless, the known values of $A_d(n)$ (in all dimensions) *suggest* that A' be also monotone increasing. We later refer to this as "the unproven monotonicity of the root sequence" ("$\sqrt{}$UM," in short).
2. A stronger observed phenomenon is the upward monotonicity of the *ratio* sequence, *i.e.*, $A_d(n)/A_d(n-1) < A_d(n+1)/A_d(n)$ (for $n \geq 2$). Equivalently, $A_d^2(n) < A_d(n+1)A_d(n-1)$, and in a more general form, $A_d^2(n) < A_d(n+k)A_d(n-k)$ (for $n > k \geq 1$).
3. Yet another observation is that $\sqrt[n]{A_d(n)} < \frac{A_d(n)}{A_d(n-1)}$ for all $n > 1$, and in a more general form (using the convention $A(0) = 1$), $\sqrt[n]{A_d(k)A_d(n-k)} < \frac{A_d(n-k)}{A_d(n-1-k)}$ (for $n > k \geq 0$).
4. Property (2) implies Properties (1) and (3). If all are true, then the ratio sequence $(\sqrt[n]{A_d(n)})$ converges to λ_d faster than the root sequence $(\frac{A_d(n+1)}{A_d(n)})$, as is widely believed to be case.

Similarly, for any sub-multiplicative sequence $(B(n))$, for which $B'(n) = \sqrt[n]{B(n)}$ is bounded from below, and for which we can show that $B'(n) \geq B'(2n)$ for any $n \in \mathbb{N}$, any known value $B(n_0)$ sets the upper bound $B'(n_0)$ on the growth constant of $(B(n))$.

2.2 Pseudo Super- and Sub-Multiplicativity

A sequence $(Z(n))$ is *pseudo super-multiplicative* (resp., *pseudo sub-multiplicative*) if $P(m+n)Z(m)Z(n) \le Z(m+n)$ (resp., $Z(m)Z(n) \ge P(m+n)Z(m+n)$), for all $m, n \in \mathbb{N}$ and for a positive subexponential function $P(\cdot)$. (Hereafter, we will consider cases in which this function is polynomial.) In such cases, we use the fact that $\lim_{n\to\infty} \sqrt[n]{P(n)} = 1$ and obtain bounds on the growth constant of $Z(n)$ from known values of $Z(n)$.

▶ **Theorem 2.1.** *Assume that for a sequence $(Z(n))$, the limit $\mu := \lim_{n\to\infty} \sqrt[n]{Z(n)}$ exists. Let c_i $(c_1 \neq 0)$ be some constants, and $\diamond \in \{\text{`}\le\text{'}, \text{`}\ge\text{'}\}$. Then:*

(a) (multiplicative polynomial) If $c_1 n^{c_2} Z^2(n) \diamond Z(2n)$ $\forall n \in \mathbb{N}$, then $\sqrt[n]{c_1 (2n)^{c_2} Z(n)} \diamond \mu$ $\forall n \in \mathbb{N}$.

(b) (index shift) If $c_1 Z^2(n+c_3) \diamond Z(2n)$ $\forall n \in \mathbb{N}$, then $\sqrt[n]{c_1 Z(n+2c_3)} \diamond \mu$ $\forall n \in \mathbb{N}$. Equivalently, if $c_1 Z^2(n) \diamond Z(2n+c_3)$ $\forall n \in \mathbb{N}$, then $\sqrt[n]{c_1 Z(n-c_3)} \diamond \mu$ $\forall n > c_3$.

Proof. In both cases, we manipulate the given relation and reach a relation of the form $\zeta(n) \diamond \zeta(2n)$. Then, we follow closely the logic of the basic argument given in the introduction.

(a) Simple manipulations of the given relation show that $\sqrt[n]{c_1 (2n)^{c_2} Z(n)} \diamond \sqrt[2n]{c_1 (4n)^{c_2} Z(2n)}$. Then, by setting $\zeta(n) = \sqrt[n]{c_1 (2n)^{c_2} Z(n)}$, we see that $\zeta(n) \diamond \zeta(2n)$. It follows that the subsequence $(\zeta(2^{i+1} n_0))_{i=0}^{\infty}$ is monotone increasing (if $\diamond =$ '$\le$') or monotone decreasing (if $\diamond =$ '$\ge$'), and converging to μ, for any natural number n_0. The claim follows.

(b) In this case, we substitute $n := m + c_3$ in the given relation and manipulate as above, obtaining that $c_1^2 Z^2(m+2c_3) \diamond c_1 Z(2m+2c_3)$. Hence, $\sqrt[m]{c_1 Z(m+2c_3)} \diamond \sqrt[2m]{c_1 Z(2m+2c_3)}$. Elementary calculus shows that the limits of the sequences $\left(\sqrt[m]{c_1 Z(m)}\right)$ and $\left(\sqrt[m]{c_1 Z(m+2c_3)}\right)$, as $m \to \infty$, are equal. Finally, we fix $\zeta(m) = \sqrt[m]{c_1 Z(m+2c_3)}$ and continue as above. The equivalent case, where the shift is in the right side of the relation, is treated similarly. ◀

3 Methods of Concatenation

We now list a few concatenation methods. For ease of exposition, we use relations that yield lower bounds on the growth constants. The sequence and its growth constant are denoted by $Z(n)$ and λ_Z, respectively. Polycubes P_1, P_2 are to be concatenated, and the largest (resp., smallest) cell of P_1 (P_2) is a_1 (a_2). Consistently with $\sqrt{}$UM, we observed that the best (largest) lower bounds on λ_d were obtained by using the largest known $A_d(n)$.

[E] The most elementary method of concatenation attaches cell a_1 to cell a_2 in a *single* way. This leads to the relation $Z^2(n) \le Z(2n)$, which implies that $\lambda_Z \ge \sqrt[n]{Z(n)}$ for all $n \in \mathbb{N}$.

[C] A simple improvement on Method [E] is achieved by considering all possible (lattice dependent) c ways of attaching a_1 to a_2, s.t. a_1 is smaller than a_2. This leads to the relation $cZ^2(n) \le Z(2n)$, which, by Theorem 2.1(a), implies that $\lambda_Z \ge \sqrt[n]{cZ(n)}$ $\forall n \in \mathbb{N}$.

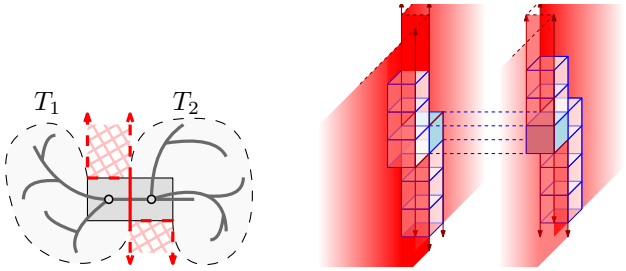
[M] A possible improvement on Method [C] can be obtained by considering all possible polycubes of size k concatenated in between P_1 and P_2. As in Method [C], there are c ways of attachments. This leads to the relation $c^2 Z(k) Z^2(n) \le Z(2n+k)$, which, by Theorem 2.1(b), implies that $\lambda_Z \ge \sqrt[n]{c^2 Z(k) Z(n-k)}$ for all $k, n \in \mathbb{N}$, such that $n > k$.

[O1] One can also *overlap* cells a_1 and a_2. This always yields a valid polycube, and different pairs of polycubes generate by this method different polycubes. This leads to $Z^2(n) \le Z(2n-1)$, which, by Theorem 2.1(b), implies that $\lambda_Z \ge \sqrt[n]{Z(n+1)}$ for all $n \in \mathbb{N}$.

[MO] One can also concatenate P_1 and P_2 through all possible polycubes of size k, using 1-cell overlaps in the middle concatenations. This leads to $Z(k) Z^2(n) \le Z(2n+k-2)$, which, by Thm. 2.1(b), implies that $\lambda_Z \ge \sqrt[n]{Z(k) Z(n-k+2)}$ for all $k, n \in \mathbb{N}$, s.t. $n > k-2$.

Dimensions	Known Values	OEIS Sequence	Concatenation Methods [E]	[C]	[O1]	Other Methods
2	56	A001168	3.703120	3.749241	3.792324	**4.00253** [1]
3	19	A001931	**6.021134**	6.379548	6.652636	—
4	16	A151830	**8.462728**	9.228670	9.757631	—
5	15	A151831	**10.909365**	12.144998	12.939813	—
6	15	A151832	**13.523756**	15.239618	16.288833	—
7	14	A151833	**15.598535**	17.924538	19.269014	—
8	12	A151834	**16.647767**	19.797643	21.497519	—
9	12	A151835	**18.841772**	22.627780	24.606050	—

Table 1 Lower bounds on the growth constants of polycubes, obtained by different methods. (Best previously-published lower bounds appear in boldface.)



(a) Two dimensions (b) Three dimensions

Figure 2 Concatenating trees.

4 Simple Applications

4.1 General

We applied methods [E], [C], and [O1] to polyominoes and polycubes, and found lower bounds on λ_d (for $2 \le d \le 9$). Table 1 summarizes our findings. (In two dimensions, the bounds are inferior to the bound 4.00253 obtained by the much stronger *twisted cylinders* method [1], which, unfortunately, cannot be generalized efficiently to higher dimensions because it becomes computationally intractable.) We show in Section 5 how to improve all known bounds in $d \ge 3$ dimensions.

4.2 Trees

A polycube is a *tree* if its cell-adjacency graph is acyclic. Tree polycubes and their respective growth constants also attracted interest in the literature (see, *e.g.*, [5]). In order to preserve the tree property, special restrictions must be enforced while concatenating them. Figures 2(a,b) show two tree polycubes T_1, T_2, having cells c_1, c_2 as their largest and smallest cells, resp., concatenated by Method [E]. To remain a tree, the only valid concatenation is the one in which c_1 and c_2 are aligned with the most dominant axis of the lexicographic order. This leaves a $(d-1)$-dimensional buffer that prevents cycles in the concatenation of T_1 and T_2.

Let $A_{d;T}(n)$ and $\lambda_{d;T}$ denote the number of n-cell tree polycubes in d dimensions, and their growth constant, respectively. As in similar examples, we obtain the relation $A^2_{d;T}(n) \le A_{d;T}(2n)$, which, by Theorem 2.1(b), implies that $\lambda_{d;T} \ge \sqrt[n]{A_{d;T}(n)}$ for all $n \in \mathbb{N}$. Table 2 shows the best lower bounds obtained this way in dimensions 2–8, in all cases using the largest known values of the respective sequences.

Dimensions	Known Values	OEIS Sequence	Method [E]
2	44	A066158	3.4045
3	17	A118356	5.5592
4	10	A191094	6.7698
5	10	A191095	8.8035
6	8	A191096	9.4576
7	7	A191097	10.0909
8	7	A191098	11.4891

Table 2 Lower bounds on the growth constants of tree polycubes of various dimensions.

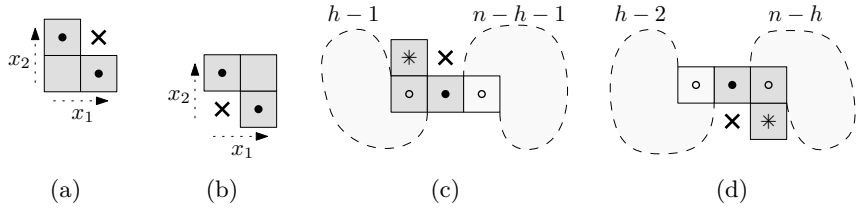


Figure 3 Constructions for the proof of Theorem 5.1.

5 Recursive Bounding

We now present a recursive scheme for improving bounds obtained by all methods described above. Let us demonstrate the scheme by a concrete example of setting lower bounds on the growth constants of polycubes. As observed earlier, the sequence enumerating d-dimensional polycubes is super-multiplicative and it has a growth constant, hence, by concatenation Method [O1], any term of the form $\sqrt[n-1]{A_d(n)}$ is a lower bound on λ_d. In practice, we can prove relations which are tighter than the super-multiplicativity condition, for example:

▶ **Theorem 5.1.** *Let $h = \lfloor (n+1)/2 \rfloor$. Then, for every $n \geq 4$, we have that*

$$A_d(n) \geq A_d(h)A_d(n-h+1) + \frac{d(d-1)^2}{2}(A_d(h-1)A_d(n-h-1) + A_d(h-2)A_d(n-h)). \quad (1)$$

Proof. The term on the left side of Relation 1 is the number of all d-dimensional polycubes of size n, whereas the terms on the right count a subset of these polycubes.

We distinguish between three types of polycubes by the connectedness of their cells.

The first term, $A_d(h)A_d(n-h+1)$, counts d-dimensional polycubes obtained by concatenating two polycubes of sizes h and $n-h+1$ with a 1-cell overlap. In these combinations, the lower h cells, as well the upper $n-h+1$ cells, form valid polycubes which share the hth cell.

The second factor of the second term, $A_d(h-1)A_d(n-h-1) + A_d(h-2)A_d(n-h)$, counts two types of constructions. In both types, we place 3-cell L-shapes (Figures 3(a,b)) in the middle in order to force the hth cell to be disconnected from either the upper $n-h$ cells or the lower $h-1$ cells (Figures 3(c,d)). The largest (smallest) cell of the lower (upper) polycube is marked by an empty circle, and the hth cell of the resulting polycube is marked by an asterisk. The trick is to mix between no-overlap and 1-overlap on the two sides of the L. To this aim, the L-shape in Figure 3(a) is overlapped with the lower polycube of size $h-1$, and concatenated to the upper polycube of size $n-h-1$ (Figure 3(c)). Similarly, the L-shape in Figure 3(b) is concatenated to the lower polycube of size $h-2$ and overlapped with the upper polycube of size $n-h$ (Figure 3(d)).

d	Bound
2	3.7944
3	**6.6621**
4	**9.7714**
5	**12.9569**
6	**16.3087**
7	**19.2927**
8	**21.5298**
9	**24.6416**

Table 3 Improved lower bounds on λ_d for $3 \leq d \leq 9$.

Let us finally explain the first factor of the second term. First, there are $\binom{d}{2}$ options for choosing the orientation of L. (Directions are denoted by x_1 and x_2, where x_1 has precedence over x_2 in the lexicographic order.) Second, the no-overlap concatenation (on one of the sides) can be done in $d-1$ ways: All directions are valid *except* direction x_2, the direction which would cover the forbidden cell (marked with an "$\times$" in Figures 3(a,b)). This constraint avoids multiple counting; otherwise, we would have in the middle a 2×2 square which can be created in more than one way. Overall, we have a factor of $\binom{d}{2}(d-1) = d(d-1)^2/2$.

It is easy to see that all resulting polycubes are different by construction. ◀

Unfortunately, we cannot derive from Relation (1) "chains" of lower bounds. However, we can apply a recursive procedure for bounding from below any value of $A_d(n)$. Since we know values of $A_d(n)$ up to some $n = n_0$, we can construct a sequence $B(n)$, such that $B(n) \leq A_d(n)$ for every n: For $1 \leq n \leq n_0$, let $B(n) = A_d(n)$; and for $n > n_0$, set $B(n)$ recursively to the value calculated from the right side of Relation 1.

One can apply this method for large values of n *ad infinitum*, or, more practically, until the available computing resources are exhausted, and choose the best value encountered. We ran this procedure up to $n \approx 12{,}000{,}000$ for $2 \leq d \leq 9$. This improved the lower bounds on λ_d in *all* of $3 \leq d \leq 9$ dimensions. Table 3 summarizes the obtained bounds.

6 Conclusion

We explore concatenation arguments and their applications to setting lower bounds on the growth constants of polycubes and tree polycubes. In the full version of the paper, we also provide a much more complex application of the method to setting an upper bound on the growth constant of *convex* polyominoes.

A possible direction for future work is analyzing the quality of our lower bounds. It was conjectured [2] that λ_d behaves asymptotically like $(2d-3)e + O(1/d)$ (as $d \to \infty$); see the blue line in the graph shown in Figure 4. Bounds obtained by the recursive-bounding method also exhibit a linear dependence on d, surprisingly similar to $3.13d-2.63$ (obtained with Rvalue=0.9998, using Python's linear least-squares regression tool `scipy.stats.linregress`); see the orange line in the same figure. Are the approximate slope π and intercept $-e$ a coincidence, or are they inherently related to the concatenation method?

Acknowledgment

The authors would like to thank Günter Rote and Vuong Bui for many helpful comments on a preliminary draft of this paper.

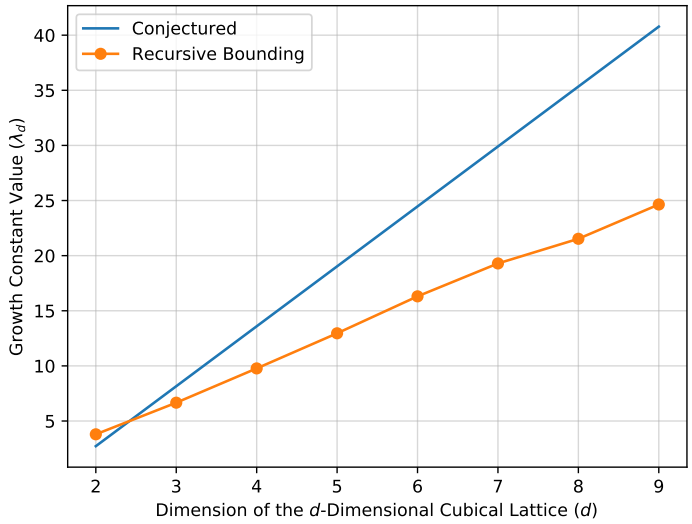


Figure 4 Conjectured growth constants (blue), and lower bounds we obtained (orange).

References

1 G. Barequet, M. Shalah, and G. Rote, $\lambda > 4$: An improved lower bound on the growth constant of polyominoes, *Comm. of the ACM*, 59 (2016), 88–95.

2 R. Barequet, G. Barequet, and G. Rote, Formulae and growth rates of high-dimensional polycubes, *Combinatorica*, 30 (2010), 257–275.

3 G. Barequet, M. Shalah, and Y. Zheng, An improved lower bound on the growth constant of polyiamonds, *J. of Combinatorial Optimization*, 37 (2019), 424–438.

4 S.R. Broadbent and J.M. Hammersley, Percolation processes: I. Crystals and mazes, *Proc. Cambridge Philosophical Society*, 53 (1957), 629–641.

5 J.A.M.S. Duarte and H.J. Ruskin, The branching of real lattice trees as dilute polymers, *J. de Physique*, 42 (1981), 1585–1590.

6 M. Eden, A two-dimensional growth process, *Proc. 4th Berkeley Symp. on Mathematical Statistics and Probability*, IV, Berkeley, CA, 223–239, 1961.

7 I. Jensen, Enumerations of Lattice Animals and Trees, *J. of Statistical Physics*, 102 (2001), 865–881.

8 I. Jensen, Counting polyominoes: A parallel implementation for cluster computing, *Proc. Int. Conf. on Computational Science*, III (Melbourne, Australia and St. Petersburg, Russia, 2003), *Lecture Notes in Computer Science*, 2659, Springer, 203–212.

9 D.A. Klarner, Cell growth problems, *Canadian J. of Mathematics*, 19 (1967), 851–863.

10 D.A. Klarner and R.L. Rivest, A procedure for improving the upper bound for the number of n-ominoes, *Canadian J. of Mathematics*, 25 (1973), 585–602.

11 N. Madras, A pattern theorem for lattice clusters, *Annals of Combinatorics*, 3 (1999), 357–384.

12 The on-line encyclopedia of integer sequences (OEIS), available at `http://www.oeis.org` .

13 G. Pólya and G. Szego, *Aufgaben und Lehrsätze aus der Analysis*, vol. 1, Julius Springer, Berlin, 1925.

14 H.N.V. Temperley, Combinatorial problems suggested by the statistical mechanics of domains and of rubber-like molecules, *Physical Review 2*, 103 (1956), 1–16.

Scheduling drones to cover outdoor events

O. Aichholzer[1], L. E. Caraballo[2], J.M. Díaz-Báñez[2], R. Fabila-Monroy[3], I. Parada[1], I. Ventura[2], and B. Vogtenhuber[1]

1 **Graz University of Technology**
oaich@ist.tugraz.at,iparada@ist.tugraz.at,bvogt@ist.tugraz.at
2 **University of Seville**
lcaraballo@us.es,dbanez@us.es,iventura@us.es
3 **Cinvestav**
ruyfabila@math.cinvestav.edu.mx

Abstract

Task allocation is an important aspect of many multi-robot systems. In this paper, we consider a new task allocation problem that appears in multi-robot aerial cinematography. The main goal is to distribute a set of tasks (shooting actions) among the team members optimizing a certain objective function. The tasks are given as sequences of waypoints with associated time intervals (scenes). We prove that the task allocation problem maximizing the total filmed time by k aerial robots (drones) can be solved in polynomial time when the drones do not require battery recharge. We also consider the problem in which the drones have a limited battery endurance and must periodically go to a static base station. For this version, we show how to solve the problem in polynomial time when only one drone is available.

1 Introduction

In the last twenty years, the use of aerial robots and aerial multi-robot systems in monitoring, surveillance, delivery of goods, network coverage, etc. [8], has brought several challenges that have attracted the attention of both, robotics and mathematics research communities. For example, the Vehicle Routing Problem (VRP) and Traveling Salesman Problem (TSP) are classical problems in the area of operations research that have got renewed attention with these applications; see [1] for a survey on VRP instances with applications to multi-objective unmanned aerial vehicle operations and [2, 4, 5] for studies in commercial scenarios that consider a combination of trucks and drones to perform the so-called last-mile delivery. Also, task allocation problems have been widely addressed in the interplay of aerial robotics and operations research. See [7] for a review on multi-robot task allocation.

The problems studied in this paper are inspired by the use of unmanned aerial vehicles (drones, also called UAVs) for autonomous cinematography planning, aimed at filming outdoor events (e.g., sport events such as cycling); see [9]. Drones are ideal to cover outdoor events in large spaces, as they do not require the existence of previous infrastructure and they can operate in places of difficult access. The input is provided by the media production director, who specifies multiple shots that should be executed to film different scenes of the event. These are basically time intervals associated with waypoints on the terrain. Each shot may require one or multiple cameras, but in the end, all information can be translated into a set

This research has been supported by the projects GALGO (Spanish Ministry of Economy and Competitiveness and MTM2016-76272-R AEI/FEDER,UE, MULTIDRONE, Grant Agreement number: 731667-H2020-ICT-2016-2017), by the Austrian Science Fund within the collaborative DACH project *Arrangements and Drawings* as FWF project I 3340-N35, and by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 734922.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

of filming tasks with temporal and spatial constraints to be accomplished by the team. The goal is to schedule the drones optimally in order to cover the filming tasks.

Problem statement

Consider a scenario in which an outdoor event is filmed by a set of drones. In the model we assume that all drones fly at unit speed and start at a point p^* in the plane called the *base*. The production director specifies certain locations and time intervals at which the filming of certain scenes is desired. We represent the input as a set F of n tuples (p_i, I_i) with $i \in \{1, \ldots, n\}$, where p_i is a point in space and I_i is a time interval. We call F the *film plan* and each (p_i, I_i) a *scene*. A scene (p_i, I_i), or part of it, can be filmed by one or multiple drones located at p_i during (part of) the time interval I_i. A *flight plan* P is a sequence of tuples $(q_1, J_1), \ldots, (q_m, J_m)$ such that for every j, q_j is equal to some p_i and $J_j \subseteq I_i$. The flight plan is assumed to be realizable by a drone starting at the base. The goal is to assign flight plans to all drones in order to film as much as possible of the film plan. The *filming time* of a flight plan assignment M is the sum of the lengths of the subintervals of I_i covered by the flight plans. Formally, it is defined as

$$\sum_{i=1}^{n} \left| \bigcup_{\substack{P \in M \\ (p_i, J_j) \in P}} (I_i \cap J_j) \right|.$$

In this paper we study the following algorithmic problems in the above scenario.

▶ **Problem 1.** *Find a flight plan maximizing the filming time for a single drone.*

▶ **Problem 2.** *Find a flight plan assignment maximizing the filming time for $k \geq 2$ drones.*

▶ **Problem 3.** *Find the minimum number of drones needed to capture the complete film plan, plus an according flight plan assignment.*

Typically, drones have limited battery endurance and periodically return to a base station to recharge or change their batteries. Let $L > 0$ be a real number and assume that each drone can fly for time at most L before it must be at the base for a battery change. We call L the *battery life*. In our model, a battery change is assumed to be instantaneous and the drone can resume its flight plan immediately after arriving at the base. The drone can also wait at the base for an arbitrary amount of time without consuming its battery.

Results

We present the following results. In the case of unlimited battery life, Problem 1 can be solved in $O(n^{5/3} + |E|)$ time and $O(|E|)$ space for n time intervals. Here, $|E|$ is the size of a graph that in the worst case is quadratic in n but it is linear in realistic inputs for cycling events [9]. For the general case in which k drones are available, we show how to solve Problem 2 in $O(n^2(\log n + k) + n|E|)$ time, while Problem 3 can be solved in $O(n^{5/3} + \sqrt{n}|E'|)$ time, where $|E'| < |E|$. All according proofs are constructive in the sense that they also provide a flight plan that attains the computed maximum time/minimum number of drones.

The case of limited battery life is more challenging. Note that the complexity of an explicit flight plan not only depends on the number of intervals, but also on the ratio between the total time T of the film plan and the battery life L. We show that Problem 1 can still be solved in polynomial time by augmenting the drone's model so that repeated identical instructions can be formulated in a compact way. We conjecture the general case for k drones with limited battery life to be NP-hard.

2 Unlimited battery life

In this section we consider the scenario where the drones do not require to recharge their batteries. We first state a lemma that allows us to discretize the solution space for Problems 1, 2, and 3 in this setting. The intuition behind Lemma 2.1 is the following. In an optimal solution, there is no reason for a drone at a point p_i to leave its current filming position before the corresponding interval I_i ends. Moreover, there is no reason for two drones to be filming at the same location at the same time. The proof of Lemma 2.1 is technical and it is based on applying the following two operations on flight plans.

Let $P := (q_1, J_1), \ldots, (q_m, J_m)$ be a flight plan, t a real number, and $1 \leq i \leq m$. Informally, our first operation yields the flight plan obtained from P by staying additional time t at position q_i and thus arriving at the subsequent intervals at later times. We formalize this definition. For a time interval $[a, b]$ we define

$$\begin{aligned} \text{extend}([a,b],t) &:= [a, b+t] \\ \text{delay}([a,b],t) &:= \begin{cases} [a,b] & \text{if } t \leq 0 \\ [a+t,b] & \text{if } 0 < t < b-a, \\ \emptyset & \text{if } t \geq b-a. \end{cases} \end{aligned}$$

Consider the sequence $P' := (q_1, J_1), \ldots, (q_{i-1}, J_{i-1}), (q_i, \text{extend}(J_i, t)), (q_{i+1}, \text{delay}(J_{i+1}, t)), (q_{i+2}, \text{delay}(J_{i+2}, t - |J_{i+1}|)), (q_{i+3}, \text{delay}(J_{i+3}, t - |J_{i+1}| - |J_{i+2}|)), \ldots, (q_m, \text{delay}(J_m, t - \sum_{j=i+1}^{m-1} |J_j|))$. Let P'' be the flight plan obtained from P' by removing every tuple whose time interval is empty. We call P'' the flight plan obtained from P by *shifting P at interval J_i by time t*; see Figure 1.

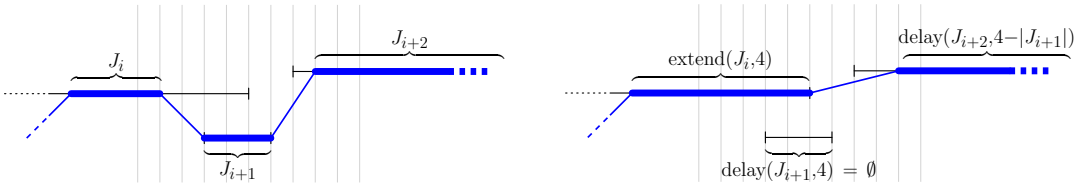


Figure 1 Example of a shifting operation.

Our second operation involves two drones that meet at some point p_i at the same time. Informally, the operation makes the drones swap their flight plans at the point of contact. Let $P_1 := (q_1, J_1), \ldots, (q_m, J_m)$ and $P_2 := (r_1, K_1), \ldots, (r_s, K_s)$ be two flight plans such that for some pair of indices $1 \leq i \leq m$ and $1 \leq j \leq s$ the following holds. The point q_i equals the point r_j and the last point of interval $J_i := [a, b]$ is contained in the interval $K_j := [c, d]$. We call the operation of replacing P_1 and P_2 with the flight plans

$$P_1' := \begin{cases} (q_1, J_1), \ldots, (q_{i-1}, J_{i-1}), (q_i, [a, d]), (r_{j+1}, K_{j+1}), \ldots, (r_s, K_s) & \text{if } a < c, \\ (q_1, J_1), \ldots, (q_{i-1}, J_{i-1}), (q_{i+1}, J_{i+1}), \ldots, (q_m, J_m) & \text{if } a \geq c. \end{cases}$$

$$P_2' := \begin{cases} (r_1, K_1), \ldots, (r_{j-1}, K_{j-1}), (q_{i+1}, J_{i+1}), \ldots, (q_m, J_m) & \text{if } a < c, \\ P_2 & \text{if } a \geq c. \end{cases}$$

swapping P_1 with P_2 at interval J_i; see Figure 2.

Note that any shifting or swapping operation maintains both realizability and total filming time of the involved flight plan(s).

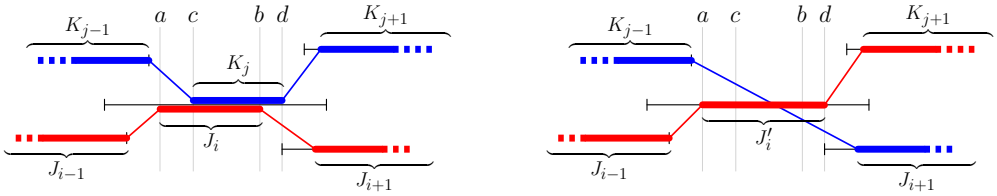


Figure 2 Example of a swapping operation.

▶ **Lemma 2.1.** *For every film plan there exists a flight plan of maximum filming time where*
1) no drone leaves a point p_i before the interval I_i ends; and
2) no two drones are at the same point p_i at the same time.

By Lemma 2.1, we may assume that in an optimal solution to Problems 1, 2, or 3 a drone never leaves a point p_i before the interval I_i ends. We can use this property to translate these problems into problems of covering directed weighted acyclic graphs with directed paths. We construct a directed graph $G = (V, E)$. The vertex set V consists of p^* and the points p_i. A pair (p_i, p_j) is an edge in the edge set E whenever a drone at a point p_i can leave at the end of I_i and arrive at p_j at a time $t \in I_j := [a, b]$. In such a case, (p_i, p_j) is assigned weight $b - t$. Every pair (p^*, p_i) is an edge in E with weight $|I_i|$. Due to geographic and time constrains, not all pairs (p_i, p_j) might be edges of E. We can compute E efficiently, in an output sensitive manner.

▶ **Lemma 2.2.** *The edge set E can be computed in $O(n^{5/3} + |E|)$ time.*

By Lemma 2.1, a flight plan for one drone that maximizes the filming time corresponds to a directed path starting at p^* of maximum weight. Since G is acyclic, such a path can be computed in $O(|E| + n)$ time by first doing a topological sort of G and then finding the desired path via dynamic programming. This solves Problem 1.

▶ **Theorem 2.3.** *Problem 1 with unlimited battery life can be solved in $O(n^{5/3} + |E|)$ time.*

Suppose that $k \geq 2$ drones are available. By Lemma 2.1, a flight plan assignment for these drones that maximizes the filming time corresponds to a set of k internally disjoint paths of G, all starting starting at p^* maximizing the sum of its weights. It is known that the problem of finding k disjoint paths of maximum total weight all starting at a given vertex in a general directed graph is NP-complete [6]. However, since G is acyclic, we can prove the following result.

▶ **Theorem 2.4.** *Problem 2 with unlimited battery life can be solved in $O(n^2(\log n + k) + n|E|)$ time.*

Now, let $G' = (V, E')$ be the following subgraph of G. The set E' is the subset of E of edges (p^*, p_j) and the edges (p_i, p_j) of weight equal to $|I_j|$. This corresponds to the cases in which a drone at p_i leaving when I_i ends can arrive at p_j just when I_j starts. In a similar way to Lemma 2.2, E' can be computed in $O(n^{5/4} + |E'|)$ time. By Lemma 2.1, an optimal solution to Problem 3 corresponds to a set of minimum cardinality of vertex disjoint paths that covers every vertex of G'. Such a set is called a *minimum path cover*. The problem of finding a minimum path cover for general directed graphs is NP-hard. However, if the digraph is acyclic then a minimum path cover can be computed in $O(\sqrt{n}m)$ time [3], where n is the number of vertices and m is the number of edges of the digraph.

▶ **Theorem 2.5.** *Problem 3 with unlimited battery life can be solved in $O(n^{5/3} + \sqrt{n}|E'|)$ time.*

3 Limited battery life for one drone

With bounded battery for each drone, the problem of selecting and scheduling drones optimally to shoot the provided time intervals is a challenging problem. It is not difficult to build an example for two drones in which, for an optimal solution, the leaving time for a drone is not necessarily the end of an interval. We conjecture the problem is NP-hard in general. In this section, we consider the problem for one drone and battery life L. This implies that the drone must return to the base at latest after time L for a battery change. We refer to the subsequence of a flight plan between two consecutive visits to the base as a *round*. Let $T = |\bigcup_{i=1}^{n} I_i|$. Note that the number of rounds of any flight plan of maximum filming time for one drone is at least $\lfloor T/L \rfloor$ (it could be exponential in n). As in the unlimited battery case, we can discretize the solution space.

▶ **Lemma 3.1.** *For every film plan there exists a flight plan of maximum filming time for one drone with limited battery life such that:*

1) *If (p_j, J_i) and (p'_j, J_{i+1}) are consecutive tuples in the flight plan, then the last point of J_i is equal to the last point of I_i.*
2) *If (p^*, J_i) and (p_j, J_{i+1}) are consecutive tuples in the flight plan and $|J_i| > 0$, then the first point of J_i is equal to the first point of I_j*
3) *If (p_j, J_i) and (p^*, J_{i+1}) are consecutive tuples in the flight plan, then the last point of J_i is equal to the last point of I_j or the drone is running out of battery at the last point of J_i.*

Lemma 3.1 implies that the set of theoretically relevant event-times for some optimal solution is discrete rather than continuous. Figure 3 shows an example scenario with an optimal flight plan.

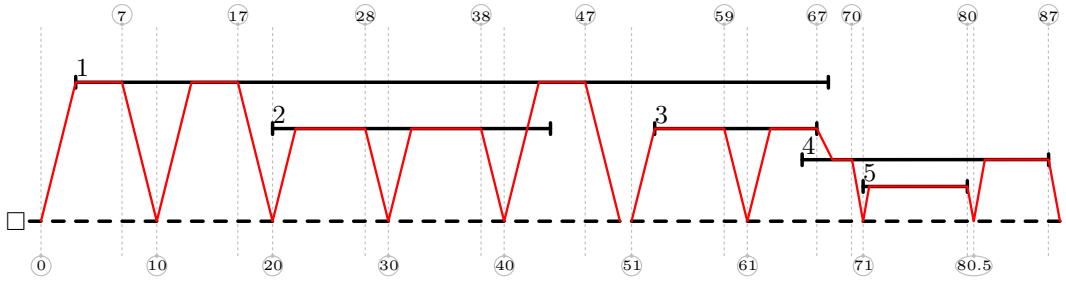


Figure 3 Scenario with 5 scenes represented by solid black lines. The dashed black line represents the base. The red solid stroke represents the movement of the drone following an optimal solution. The battery lasts 10 units of time. Closed in gray circles are the time instants associated to the drone's moves.

We can show that the problem can be solved in polynomial time if we augment the computational model of the drone. More precisely, our drone operates from a set of *driving instructions* of the type "go to the base" and "go to interval I_i". In the augmented model, the drone acts by default according to an internal protocol that allows the drone to perform several moves between two consecutive instructions. In this model, "go to interval I_i" means "repeatedly go to interval I_i, film as long as possible, and return to the base station, until the next instruction comes". Thus, an optimal scheduling can be encoded by a polynomial sequence of driving instructions. A crucial result is the following.

▶ **Lemma 3.2.** *An optimal solution for Problem 1 with limited battery life can be encoded using a linear number of driving instructions.*

The results above allow us to apply dynamic programming to compute an optimal sequence of instructions in polynomial time.

▶ **Theorem 3.3.** *Problem 1 with limited battery life can be solved in polynomial time.*

References

1 M. Adbelhafiz, A. Mostafa, and A. Girard. Vehicle routing problem instances: Application to multi-UAV mission planning. In *AIAA Guidance, Navigation, and Control Conference*, 2010. doi:10.2514/6.2010-8435.

2 N. Agatz, P. Bouman, and M. Schmidt. Optimization approaches for the traveling salesman problem with drone. *Transportation Science*, 52(4):965–981, 2018. doi:10.1287/trsc.2017.0791.

3 F. T. Boesch and J. F. Gimpel. Covering points of a digraph with point-disjoint paths and its application to code optimization. *Journal of the ACM*, 24(2):192–198, 1977. doi:10.1145/322003.322005.

4 John Gunnar Carlsson and Siyuan Song. Coordinated logistics with a truck and a drone. *Management Science*, 64(9):4052–4069, 2018. doi:10.1287/mnsc.2017.2824.

5 C. C. Murray and A. G. Chu. The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research Part C: Emerging Technologies*, 54:86–109, 2015. doi:10.1016/j.trc.2015.03.005.

6 G. Naves, N. Sonnerat, and A. Vetta. Maximum flows on disjoint paths. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques. 13th International Workshop*, pages 326–337, 2010. doi:10.1007/978-3-642-15369-3_25.

7 E. Nunes, M. Manner, H. Mitiche, and M. Gini. A taxonomy for task allocation problems with temporal and ordering constraints. *Robotics and Autonomous Systems*, 90:55–70, 2017. doi:10.1016/j.robot.2016.10.008.

8 H. Shakhatreh, A. H. Sawalmeh, A. Al-Fuqaha, Z. Dou, E. Almaita, I. Khalil, N. S. Othman, A. Khreishah, and M. Guizani. Unmanned aerial vehicles (UAVs): A survey on civil applications and key research challenges. *IEEE Access*, 7:48572–48634, 2019. doi:10.1109/ACCESS.2019.2909530.

9 A. Torres-González, J. Capitán, R. Cunha, A. Ollero, and I. Mademlis. A multidrone approach for autonomous cinematography planning. In *Iberian Robotics Conference*, pages 337–349, 2017.

Edge Guarding Plane Graphs

Paul Jungeblut[1] and Torsten Ueckerdt[2]

1 **Karlsruhe Institute of Technology, Germany**
`paul.jungeblut@kit.edu`
2 **Karlsruhe Institute of Technology, Germany**
`torsten.ueckerdt@kit.edu`

Abstract

Let $G = (V, E)$ be a plane graph. We say that a face f of G is *guarded* by an edge $vw \in E$ if at least one vertex from $\{v, w\}$ is on the boundary of f. For a planar graph class $\mathcal{G}$ we ask for the minimal number of edges needed to guard all faces of any n-vertex graph in $\mathcal{G}$. In this extended abstract we provide new bounds for two planar graph classes, namely the quadrangulations and the stacked triangulations.

1 Introduction

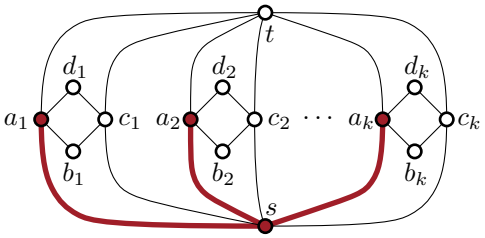
In 1975, Chvátal [4] laid the foundation for the widely studied field of *art gallery problems* by answering how many guards are needed to observe all interior points of any given n-sided polygon P. Here a guard is a point p in P and it can observe any other point q in P, if the line segment pq is fully contained in P. He shows that $\lfloor n/3 \rfloor$ guards are sometimes necessary and always sufficient. Fisk [7] revisited Chvátal's Theorem in 1978 and gave a very short and elegant new proof by introducing diagonals into the polygon P to obtain a triangulated, outerplanar graph. Such graphs are 3-colorable and in each 3-coloring all faces are incident to vertices of all three colors, so the vertices of the smallest color class can be used as guard positions. Bose et al. [3] studied the problem to guard the faces of a plane graph instead of a polygon. A *plane graph* is a graph $G = (V, E)$ with an embedding in $\mathbb{R}^2$ with not necessarily straight edges and no crossings in the interior of any two edges. Here a face f is guarded by a vertex v, if v is on the boundary of f. They show that $\lfloor n/2 \rfloor$ vertices (so called *vertex guards*) are sometimes necessary and always sufficient for n-vertex plane graphs.

We consider a variant of this problem introduced by O'Rourke [9]. He shows that only $\lfloor n/4 \rfloor$ guards are necessary in Chvátal's original setting if each guard is assigned to an edge of the polygon that he can patrol along instead of being fixed to a single point. Considering plane graphs again, an *edge guard* is an edge $vw \in E$ and guards all faces having v and/or w on their boundary. For a given planar graph class $\mathcal{G}$, we ask for the minimal number of edge guards needed to guard all faces of every plane n-vertex graph in $\mathcal{G}$.

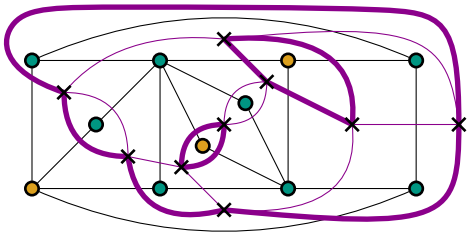
General (not necessarily triangulated) n-vertex plane graphs might need at least $\lfloor n/3 \rfloor$ edge guards, even when requiring 2-connectedness [3]. The best known upper bounds have recently be presented by Biniaz et al. [1] and come in two different fashions: First, any n-vertex plane graph can be guarded by $\lfloor 3n/8 \rfloor$ edge guards found in an iterative process. Second, a coloring approach yields an upper bound of $\lfloor n/3 + \alpha/9 \rfloor$ edge guards where α counts the number of quadrangular faces in G. Looking at n-vertex triangulations, Bose et al. [3] provide a lower bound of $\lfloor (4n-8)/13 \rfloor$ edge guards. A corresponding upper bound of $\lfloor n/3 \rfloor$ edge guards was published earlier in the same year by Everett and Rivera-Campo [6].

This note is based on the master's thesis of the first author [8] and we present our results on quadrangulations and stacked triangulations. For both planar graph classes we give a lower and an upper bound for the number of edge guards. All graphs considered below are assumed to be *plane*, i.e. given with a fixed plane embedding.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.



(a) A quadrangulation with $4k+2$ vertices needing k edge guards (drawn thick and red).



(b) A quadrangulation G (black edges) and its dual G^* (purple edges) with a 2-factor (thick edges). The vertex coloring is a guard coloring.

Figure 1 Lower and upper bound for quadrangulations.

2 Main Results

2.1 Quadrangulations

Quadrangulations are the maximal plane bipartite graphs and every face is bounded by exactly four edges. All coloring approaches developed previously [1, 6] fail on graphs containing quadrangular faces. The previously best known upper bounds are the ones given by Biniaz et al. [1] for general plane graphs, $\lfloor 3n/8 \rfloor$ respectively $\lfloor n/3 + \alpha/9 \rfloor$, where α is the number of quadrilateral faces. For n-vertex quadrangulations we have $\alpha = n - 2$, so $\lfloor n/3 + (n-2)/9 \rfloor = \lfloor (4n-2)/9 \rfloor > \lfloor 3n/8 \rfloor$ for $n \geq 4$. In this section we provide a better upper and a not yet matching lower bound. Closing the gap remains an open problem.

▶ **Theorem 2.1.** *For $k \in \mathbb{N}$ there exists a quadrangulation Q_k with $n = 4k+2$ vertices needing $k = (n-2)/4$ edge guards.*

Proof. Define $Q_k = (V, E)$ with $V := \{s, t\} \cup \bigcup_{i=1}^{k} \{a_i, b_i, c_i, d_i\}$ and $E := \bigcup_{i=1}^{k} \{sa_i,\ sc_i,\ ta_i,\ tc_i,\ a_ib_i,\ a_id_i,\ c_ib_i,\ c_id_i\}$ as the union of k vertex disjoint 4-cycles and two extra vertices connecting them. Figure 1a shows this and a planar embedding. Now for any two distinct $i, j \in \{1, \ldots, k\}$ the two quadrilateral faces (a_i, b_i, c_i, d_i) and (a_j, b_j, c_j, d_j) are only connected via paths through s or t. Therefore, no edge can guard two or more of them and we need at least k edge guards for Q_k. On the other hand it is easy to see that $\{sa_1, \ldots, sa_k\}$ is an edge guard set of size k, so Q_k needs exactly k edge guards. ◀

The following Lemma is from Bose et al. [2] and we cite it using the terminology of Biniaz et al. [1]. A *guard coloring* of a plane graph G is a non-proper 2-coloring of its vertex set, such that each face f of G has at least one boundary vertex of each color and at least one monochromatic edge (i.e. an edge where both endpoints receive the same color). They prove that a guard coloring exists for all graphs without any quadrangular faces.

▶ **Lemma 2.2** ([2, Lemma 3.1])**.** *If there is a guard coloring for an n-vertex plane graph G, then G can be guarded by $\lfloor n/3 \rfloor$ edge guards.*

▶ **Theorem 2.3.** *Every quadrangulation can be guarded by $\lfloor n/3 \rfloor$ edge guards.*

Proof. Let G be a quadrangulation. We show that there is a guard coloring for G, which is sufficient by Lemma 2.2. Consider the dual graph $G^* = (V^*, E^*)$ of G with its inherited plane embedding, so each vertex $f^* \in V^*$ is placed inside the face f of G corresponding to it. Since every face of G is of degree four, its dual graph G^* is 4-regular. Using Petersen's

2-Factor Theorem [10][1] we get that G^* contains a 2-factor H (a spanning 2-regular subgraph). Any vertex of H is of degree 2, so H is a set of vertex-disjoint cycles that can be nested inside each other. Now define a 2-coloring $\text{col} : V \to \{0, 1\}$ for the vertices of G: For each $v \in V$ let c_v be the number of cycles C of H such that v belongs to the region of the embedding surrounded by C. The color of v is determined by the parity of c_v as $\text{col}(v) := c_v \mod 2$.

We claim that this yields a guard coloring of G: Any edge $e = ab \in E$ has a corresponding dual edge e^*. If $e^* \in E(H)$, then e crosses exactly one cycle edge, so $|c_a - c_b| = 1$ and therefore $\text{col}(a) \neq \text{col}(b)$. Otherwise $e \notin E(H)$, so its two endpoints are in the same cycles, thus $\text{col}(a) = \text{col}(b)$ and e is monochromatic. Because H is a 2-factor, each face has exactly two monochromatic edges. ◀

Figure 1b shows an example quadrangulation with a 2-factor in its dual graph. From here it is easy to color the vertices in green and orange to obtain a guard coloring.

In order to bridge the gap between the lower ($\lfloor (n-2)/4 \rfloor$) and the upper bound ($\lfloor n/3 \rfloor$), we also consider the subclass of 2-degenerate quadrangulations in the master's thesis [8, Theorem 5.9]:

▶ **Theorem 2.4.** *Every n-vertex 2-degenerate quadrangulation can be guarded by $\lfloor n/4 \rfloor$ edge guards.*

Note that this bound is best possible, as the quadrangulations constructed in Theorem 2.1 are 2-degenerate.

2.2 Stacked Triangulations

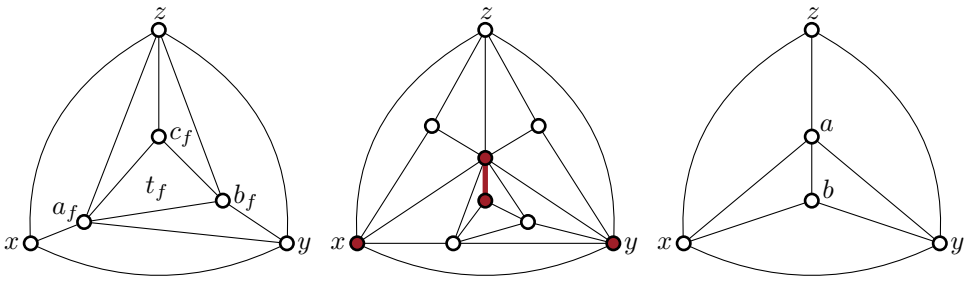
The stacked triangulations (also known as Apollonian networks or planar 3-trees) are a subclass of the triangulations that can recursively be formed by the following rules: (i) A triangle is a stacked triangulation and (ii) if G is a stacked triangulation and f an inner face, then the graph obtained by placing a new vertex into f and connecting it with all three boundary vertices is again a stacked triangulation. We shall prove that the stacked triangulations are a non-trivial subclass of the triangulations that need strictly less than $\lfloor n/3 \rfloor$ edge guards (which is the best known upper bound for general triangulations).

▶ **Theorem 2.5.** *For even $k \in \mathbb{N}$ there is a stacked triangulation G with $n = (7k+4)/2$ vertices needing at least $k = (2n-4)/7$ edge guards.*

Proof. Let S be a stacked triangulation with k faces and therefore $(k+4)/2$ vertices (by Euler's formula). Subdivide each face f of S with three new vertices a_f, b_f, c_f such that the resulting graph is a stacked triangulation and these three vertices form a new triangular face t_f, i.e. f and t_f do not share any boundary vertices. This subdivision is shown in Figure 2a for a single face f. Then G has $n = (k+4)/2 + 3k = (7k+4)/2$ vertices. For any two distinct faces f, g of S the shortest path between any two boundary vertices of the new faces t_f and t_g has length at least 2, so no edge can guard both of them. Therefore G needs at least k edge guards. ◀

▶ **Theorem 2.6.** *Every n-vertex stacked triangulation can be guarded by $\lfloor 2n/7 \rfloor$ edge guards.*

[1] Diestel [5, Corollary 2.1.5] gives a very short and elegant proof of this theorem in his book. He only considers simple graphs there, but all steps in the proof (including the given proof of Hall's Theorem [5, 11, Theorem 2.1.2]) also work for multigraphs like G^* that have at most two edges between any pair of vertices.



(a) Face $f = (x, y, z)$ gets subdivided by three vertices a_f, b_f, c_f forming a new face t_f.

(b) If x, y are guarded, one additional edge suffices to guard all inner faces.

(c) Subdividing with two vertices allows to assume that $x, y \in V(\Gamma)$, where Γ is an edge guard set of minimum cardinality.

Figure 2 Lower and upper bound for stacked triangulations.

A proof of Theorem 2.6 is given in the master's thesis [8, Theorem 4.14] but it is too long for this extended abstract. We restrict ourselves to briefly describing the main idea: We do induction on n, the number of vertices. Given any n-vertex stacked triangulation, we find a triangle $\triangle := \{x, y, z\} \subseteq V(G)$ containing at least $k^- \geq 4$ vertices inside of it but among all possible candidates one where k^- is minimal. Let $V^- \subseteq V$ be the vertices in the interior of $\triangle$. We remove V^- from G, so $\triangle$ becomes a face and we subdivide it with $k^+ < k^-$ new vertices V^+. Call the resulting graph G'. Applying the induction hypothesis on G' provides us with an edge guard set Γ' of size at most $\lfloor 2|V(G')|/7 \rfloor$. We show that Γ' can be augmented to and edge guard set Γ for G with size $|\Gamma| = |\Gamma'| + \ell$, such that $\ell/(k^- - k^+) \leq 2/7$, so that Γ has size at most $\lfloor 2n/7 \rfloor$.

For example consider a stacked triangulation G with a separating triangle $\triangle = \{x, y, z\}$ as shown in Figure 2b with $k^- = 6$ vertices V^- inside (the figure only shows the separating triangle and its interior vertices). Assume for now that $V^+ = \emptyset$, so $\triangle$ is a face in G'. An edge guard set Γ' of G' guards $\triangle$, for example we could have $x \in V(\Gamma')$ and $y, z \notin V(\Gamma')$. But then – after reinserting the vertices of V^- – no single edge can guard all the remaining faces. So in this situation it is impossible to extend Γ' by a single edge to and edge guard set Γ for G. The following lemma tells us how to choose V^+ instead, such that such a situation cannot arise.

▶ **Lemma 2.7.** *Let $\{x, y, z\}$ be a face of a stacked triangulation G. By stacking two new vertices into $\{x, y, z\}$ we can obtain a stacked triangulation H such that for each edge guard set Γ of H there is an edge guard set Γ' with $x, y \in V(\Gamma')$ and $|\Gamma'| \leq |\Gamma|$.*

Proof. Add vertex a with edges xa, ya, za and then vertex b with edges ab, xb, yb to obtain H (see Figure 2c). Now let Γ be any edge guard set for H not yet fulfilling the requirements, so $|\{x, y\} \cap V(\Gamma)| \leq 1$. If $b \in V(\Gamma)$ as part of an edge vb, we can set $\Gamma' := (\Gamma \setminus \{vb\}) \cup \{xy\}$. This is possible, because for any possible neighbor v of b, edge xy guards a superset of the faces that vb guards. If otherwise $b \notin V(\Gamma)$, we assume without loss of generality that $x \in V(\Gamma)$ and $y \notin V(\Gamma)$. Note that $|\{x, y\} \cap V(\Gamma)| \geq 1$, because face $\{x, y, b\}$ must be guarded. Face $\{a, b, y\}$ can then only be guarded by edge va where $v \in \{x, z\}$. Since $N(a) \subseteq N(y)$ we can set $\Gamma' := (\Gamma \setminus \{va\}) \cup \{vy\}$. In both cases $x, y \in V(\Gamma')$ and $|\Gamma'| \leq |\Gamma|$. ◀

Let us go back to the example in Figure 2b: Using Lemma 2.7, we can now remove the six vertices in V^-, add two new ones $V^+ := \{a, b\}$ as in Figure 2c and assume that the

induction hypothesis gives us an edge guard set Γ' with $x, y \in V(\Gamma')$. Then one additional edge is enough to guard the remaining inner faces and $\ell/(|V^-| - |V^+|) = 1/(6-2) \leq 2/7$ as desired. This guard set is shown in Figure 2b in red.

In addition to Lemma 2.7, we prove two more of this kind in the master's thesis [8] and which we list here without a proof. Like the lemma above, they describe how to add new vertices V^+ into a stacked triangulation, such that the resulting graph is still a stacked triangulation and that we can assume certain properties of minimal edge guard sets. Combining them, allows to handle all possible ways how the vertices V^- inside $\triangle$ can be connected.

▶ **Lemma 2.8.** *Let G be a stacked triangulation, v be a vertex of degree 3 and x, y, z its neighbors in G. Then for any edge guard set Γ guarding G we have $|\{v, x, y, z\} \cap V(\Gamma)| \geq 2$.*

▶ **Lemma 2.9.** *Let (x, y, z) be a face of a stacked triangulation G. By stacking three new vertices into (x, y, z) we can obtain a stacked triangulation H such that for each edge guard set Γ of H there is an edge guard set Γ' with $x \in V(\Gamma')$ and an edge $vw \in \Gamma'$ with $v \in \{x, y, z\}$ and w inside (x, y, z). Further $|\Gamma'| \leq |\Gamma|$.*

We conclude this note with the following open problems:

▶ **Open Problems.** How many edge guards are sometimes necessary and always sufficient for quadrangulations, (4-connected) triangulations and general plane graphs?

Acknowledgments

We thank Kolja Knauer for interesting discussions on the topic.

References

1 Ahmad Biniaz, Prosenjit Bose, Aurélien Ooms, and Sander Verdonschot. Improved Bounds for Guarding Plane Graphs with Edges. *Graphs and Combinatorics*, 35(2):437–450, 3 2019. URL: https://doi.org/10.1007/s00373-018-02004-z, doi:10.1007/s00373-018-02004-z.

2 Prosenjit Bose, David Kirkpatrick, and Zaiqing Li. Worst-Case-Optimal Algorithms for Guarding Planar Graphs and Polyhedral Surfaces. *Computational Geometry*, 26(3):209 – 219, 2003. URL: http://www.sciencedirect.com/science/article/pii/S0925772103000270, doi:https://doi.org/10.1016/S0925-7721(03)00027-0.

3 Prosenjit Bose, Thomas Shermer, Godfried Toussaint, and Binhai Zhu. Guarding Polyhedral Terrains. *Computational Geometry*, 7(3):173 – 185, 1997. URL: http://www.sciencedirect.com/science/article/pii/0925772195000348, doi:https://doi.org/10.1016/0925-7721(95)00034-8.

4 Vasek Chvátal. A Combinatorial Theorem in Plane Geometry. *Journal of Combinatorial Theory, Series B*, 18(1):39–41, 1975.

5 Reinhard Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Heidelberg, 5 edition, 8 2016.

6 Hazel Everett and Eduardo Rivera-Campo. Edge Guarding Polyhedral Terrains. *Computational Geometry*, 7(3):201 – 203, 1997. URL: http://www.sciencedirect.com/science/article/pii/0925772195000518, doi:https://doi.org/10.1016/0925-7721(95)00051-8.

7 Steve Fisk. A Short Proof of Chvátal's Watchman Theorem. *Journal of Combinatorial Theory, Series B*, 24(3):374, 1978. URL: http://www.sciencedirect.com/science/article/pii/009589567890059X, doi:https://doi.org/10.1016/0095-8956(78)90059-X.

8 Paul Jungeblut. Edge Guarding Plane Graphs. Master's thesis, Karlsruhe Institute of Technology, 10 2019. URL: https://i11www.iti.kit.edu/extra/publications/j-egpg-19.pdf.

9 Joseph O'Rourke. Galleries Need Fewer Mobile Guards: A Variation on Chvátal's Theorem. *Geometriae Dedicata*, 14(3):273–283, 9 1983. URL: https://doi.org/10.1007/BF00146907, doi:10.1007/BF00146907.

10 Julius Petersen. Die Theorie der regulären graphs. *Acta Mathematica*, 15(1):193–220, 1891.

11 Hall Philip. On Representatives of Subsets. *J. London Math. Soc*, 10(1):26–30, 1935.

Geometric bistellar moves relate triangulations of Euclidean, hyperbolic and spherical manifolds

Tejas Kalelkar[*1] **and Advait Phanse**[†2]

1 **Indian Institute of Science Education and Research Pune**
`tejas@iiserpune.ac.in`
2 **Indian Institute of Science Education and Research Pune**
`advait.phanse@students.iiserpune.ac.in`

Abstract

A geometric triangulation of a Riemannian manifold is a triangulation where the interior of each simplex is totally geodesic. Bistellar moves are local changes to the triangulation which are higher dimensional versions of the flip operation of triangulations in a plane. We show that geometric triangulations of a compact hyperbolic, spherical or Euclidean n-dimensional manifold are connected by geometric bistellar moves (possibly adding or removing vertices), after taking sufficiently many derived subdivisions. For dimension 2 and 3, we show that geometric triangulations of such manifolds are directly related by geometric bistellar moves.

1 Introduction and Notations

If we do not allow adding or removing vertices, it is known that the flip graph of Euclidean triangulations of 2-dimensional polytopes is connected. This forms the basis for the Lawson edge flip algorithm to obtain a Delaunay triangulation. For 5-dimensional polytopes, Santos [7] has given examples of triangulated polytopes with disconnected flip graphs. The problem of connectedness of such a flip graph for 3-dimensional polytopes is still open.

In this article we show that if we allow bistellar moves which add or remove vertices then the flip graph is connected in dimension 3 not just for polytopes but also for geometric triangulations of any compact Euclidean, spherical or hyperbolic manifold. This can be used to show that any quantity calculated from a geometric triangulation which is invariant under geometric Pachner moves is in fact an invariant of the manifold. Furthermore in dimension greater than 3, any two such triangulations are related by bistellar moves after taking suitably many derived subdivisions.

In [4] we give an alternate approach using shellings to relate geometric triangulations via topological bistellar moves (so intermediate triangulations may not be geometric). This gives a tighter bound on the number of such flips required to relate the geometric triangulations, than the one which can be calculated from the algorithm in this article.

A *geometric triangulation* of a Riemannian manifold is a finite triangulation where the interior of each simplex is a totally geodesic disk. Every constant curvature manifold has a geometric triangulation. A *common subcomplex* of simplicial triangulations K_1 and K_2 of M is a simplicial complex structure L on a subspace of M such that $K_1|_{|L|} = K_2|_{|L|} = L$. The main result in this article is the following:

▶ **Theorem 1.1.** *Let K_1 and K_2 be geometric simplicial triangulations of a compact constant curvature manifold M with a (possibly empty) common subcomplex L with $|L| \supset \partial M$. When M is spherical we assume that the diameter of the star of each simplex is less than π. Then*

* Supported by MATRICS grant of SERB, GoI
† Supported by NBHM fellowship, GoI

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

for some $s \in \mathbb{N}$, the s-th derived subdivisions $\beta^s K_1$ and $\beta^s K_2$ are related by geometric bistellar moves which keep $\beta^s L$ fixed.

In dimension 2 and 3, every internal stellar move can be realised by geometric bistellar moves (see for example Lemma 2.11 of [3]), so we get the following immediate corollary:

▶ **Corollary 1.2.** *Let K_1 and K_2 be geometric simplicial triangulations of a closed constant curvature 2 or 3-manifold M. When M is spherical we assume that the diameter of the star of each simplex is less than π. Then K_1 is related to K_2 by geometric bistellar moves.*

An *abstract simplicial complex* consists of a finite set K^0 (the vertices) and a family K of subsets of K^0 (the simplexes) such that if $B \subset A \in K$ then $B \in K$. A *simplicial isomorphism* between simplicial complexes is a bijection between their vertices which induces a bijection between their simplexes. A *realisation* of a simplicial complex K is a subspace $|K|$ of some $\mathbb{R}^N$, where K^0 is represented by a finite subset of $\mathbb{R}^N$ and vertices of each simplex are in general position and represented by the linear simplex which is their convex hull. Every simplicial complex has a realisation in $\mathbb{R}^N$ where N is the size of K_0, by representing K_0 as a basis of $\mathbb{R}^N$. Any two realisations of a simplicial complex are simplicially isomorphic. For A a simplex of K, we denote by ∂A the *boundary complex* of A. When the context is clear, we shall use the same symbol A to denote the simplex and the simplicial complex $A \cup \partial A$. We call K a *simplicial triangulation* of a manifold M if there exists a homeomorphism from a realisation $|K|$ of K to M. The simplexes of this triangulation are the images of simplexes of $|K|$ under this homeomorphism.

▶ **Definition 1.3.** For A and B simplexes of a simplicial complex K, we denote their join $A \star B$ as the simplex $A \cup B$. As the join of totally geodesic disks in a constant curvature manifold gives a totally geodesic disk, operations involving joins are well-defined in the class of geometric triangulations of a constant curvature manifold.

The link of a simplex A in a simplicial complex K is the simplicial complex defined by $lk(A, K) = \{B \in K : A \star B \in K\}$. The (closed) star of A in K is the simplicial complex defined by $st(A, K) = A \star lk(A, K)$.

▶ **Definition 1.4.** Suppose that A is an r-simplex in a simplicial complex K of dimension n then a stellar subdivision on A gives the geometric triangulation $(A, a)K$ by replacing $st(A, K)$ with $a \star \partial A \star lk(A, K)$ for $a \in int(A)$. The inverse of this operation $(A, a)^{-1}K$ is called a stellar weld and they both are together called stellar moves. When $lk(A, K) = \partial B$ for some $(n - r)$-dimensional geometric simplex $B \notin K$, then the bistellar move $\kappa(A, B)$ consists of changing K by replacing $A \star \partial B$ with $\partial A \star B$. It is the composition of a stellar subdivision and a stellar weld, namely $(B, a)^{-1}(A, a)$. Another way of defining this is to take a disk subcomplex D of K which is simplicially isomorphic to a disk D' in $\partial\Delta^{n+1}$ and the flip consists of replacing it with the disk $\partial\Delta^{n+1} \setminus int(D')$.

The derived subdivision βK of K is obtained from K by performing a stellar subdivision on all r-simplexes, and ranging r inductively from n down to 1.

All stellar and bistellar moves we shall consider are geometric in nature. See Fig 1 for a combinatorial bistellar move which is not geometric.

As the supports in $\mathbb{R}^N$ of two triangulations of a manifold may be different so when the manifold is not a polytope we can not take a linear cobordism between them. A subtle point here is that even if we obtain a common geometric refinement of two geometric triangulations, the refinement may not be a simplicial subdivision of the corresponding simplicial complexes. To see a topological subdivision which is not a simplicial subdivision, observe that there

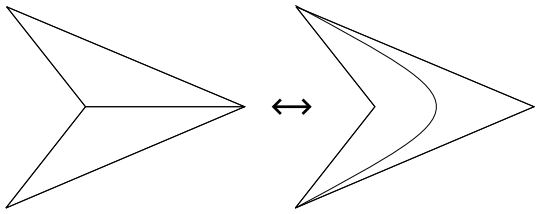


Figure 1 A 2-2 combinatorial bistellar move which is not geometric.

exists a simplicial triangulation K of a 3-simplex Δ which contains in its 1-skeleton a trefoil with just 3 edges [5]. If K were a simplicial subdivision of Δ there would exist a linear embedding of Δ in some $\mathbb{R}^N$ which takes simplexes of K to linear simplexes in $\mathbb{R}^N$. As the stick number of a trefoil is 6, there can exist no such embedding. While there may not exist such a global embedding of a geometric triangulation K as a simplicial complex in $\mathbb{R}^N$ which takes geometric subdivisions to linear (simplicial) subdivisions, for constant curvature manifolds there does exist such a local embedding on stars of simplexes of K. This is the property we exploit in this note.

2 Star-convex flat polyhedra

▶ **Definition 2.1.** We define a flat polyhedron P in $\mathbb{R}^n$ to be the realisation of a simplicial complex in $\mathbb{R}^n$ which is homeomorphic to an n-dimensional closed ball. We call a flat polyhedron P in $\mathbb{R}^n$ strictly star-convex with respect to a point x in its interior if for any $y \in P$, the interior of the segment $[x, y]$ lies in the interior of P.

We call a triangulation K of P regular if there is a function $h : |K| \to \mathbb{R}$ that is linear on each simplex of K and strictly convex across codimension one simplexes of K, i.e., if points x and y are in neighboring top-dimensional simplexes of K then the segment connecting $h(x)$ and $h(y)$ is above the graph of h (except at the end points).

In their proof of the weak Oda conjecture, Morelli [6] and Wlodarczyk [8] proved that any two triangulations of a convex polyhedron are related by a sequence of stellar moves. As interior stellar moves can be given by bistellar moves in dimension 3, Izmestiev and Schlenker [3] have improved on this result to show the following:

▶ **Theorem 2.2** (Lemma 2.11 of [3])**.** *Any two triangulations of a convex polyhedron P in $\mathbb{R}^3$ can be connected by a sequence of geometric bistellar moves, boundary geometric stellar moves and continuous displacements of the interior vertices.*

With their techniques however, even when the two triangulations agree on the boundary, we still need boundary stellar moves to relate them. Our aim in this section is to show that their techniques can be tweaked to give a boundary relative version for triangulations of strictly star-convex flat polyhedra in any dimension. The main theorem of this section is the following:

▶ **Theorem 2.3.** *Let $P \subset \mathbb{R}^n$ be a strictly star-convex flat polyhedron. Let K_1 and K_2 be triangulations of P that agree on the boundary. Then for some $s \in \mathbb{N}$, their s-th derived subdivisions $\beta^s K_1$ and $\beta^s K_2$ are related by geometric bistellar moves.*

We use the following simple observation in the proof:

▶ **Lemma 2.4** (Lemma 4, Ch 1 of [9]). *Let K and L denote two simplicial complexes with $|K| \subset |L|$. Then there exists $r \in \mathbb{N}$ and a subdivision K' of K such that K' is a subcomplex of $\beta^r L$.*

▶ **Lemma 2.5.** *Let K denote a triangulated flat polyhedron. Then for some $s \in \mathbb{N}$, its s-th derived subdivision $\beta^s K$ is regular.*

Proof. Let Δ be an n-simplex with $|\Delta| \supset |K|$. By Lemma 2.4, there exists an $r \in \mathbb{N}$ and subdivision K' of K which is a subcomplex of $\beta^r \Delta$. As Δ is trivially a regular triangulation, so its stellar subdivision $\beta^r \Delta$ is also regular (see for instance [2]). Restricting its regular function to the subcomplex K' we get K' to be regular, as codimension one simplexes of K' are also codimension one simplexes of $\beta^r \Delta$. As $|K| = |K'|$ so applying Lemma 2.4 a second time, we get $s \in \mathbb{N}$ such that $\beta^s K$ is a subdivision of K'. Finally as $\beta^s K$ is the subdivision of a regular subdivision K' of K so by Claim 3 in proof of Theorem 1 of [1], $\beta^s K$ is a regular triangulation. ◀

Proof of 2.3. The techniques in this proof are essentially those of Morelli and Wlodarczyk as detailed in Section 2 of [3].

Choose $a \in \mathbb{R}^{n+1}$ outside K_1 such that the orthogonal projection map $pr : \mathbb{R}^{n+1} \to \mathbb{R}^n$ takes the support of $C(K_1) = a \star K_1 \subset \mathbb{R}^{n+1}$ onto P and takes a to the interior of an n-simplex of K_1. By Lemma 2.5, there exists $s \in \mathbb{N}$ so that $K = \beta^s C(K_1)$ is a regular simplicial cobordism between $\beta^s K_1$ and $\beta^s C(\partial K_1)$. Choose new vertices of the derived subdivision K such that for any simplex $A \in K$ of dimension less than $n+1$, $pr(A)$ is a simplex of the same dimension as A.

Let $h : |K| \to \mathbb{R}$ be a regular function for K. If a simplex σ' has some point above a simplex σ then $\frac{\partial h}{\partial x_{n+1}}$ on σ' is greater than $\frac{\partial h}{\partial x_{n+1}}$ on σ. So inductively removing simplexes in non-increasing order of the vertical derivative of h we ensure that the projection of the upper boundary onto P is always one-to-one. That is, we get a sequence of triangulations $\Sigma_0 = K$, Σ_1, ... , $\Sigma_N = K_1$ such that $\Sigma_{i+1} = \Sigma_i \setminus \sigma_i$ and the orthogonal projection $pr : \partial^+ \Sigma_i \to P$ from the upper boundary of Σ_i onto P is one-to-one for every i. Removing an $n+1$-simplex σ_i from K corresponds to a bistellar move on $\partial^+ \Sigma_i$. As the projection map is linear so it also corresponds to a bistellar move taking $pr(\partial^+ \Sigma_i)$ to $pr(\partial^+ \Sigma_{i+1})$. Therefore $pr(\partial^+ \Sigma_0) = \beta^s C(\partial K_1)$ is bistellar equivalent to $pr(\partial^+ \Sigma_N) = \beta^s K_1$. Consequently, $\beta^s K_1$ is bistellar equivalent to $\beta^s K_2$ via $\beta^s C(\partial K_1) = \beta^s C(\partial K_2)$. ◀

3 Geometric manifolds

▶ **Definition 3.1.** Let K be a geometric triangulation of a Riemannian manifold M and let L be a subcomplex of K. We call K locally geodesically-flat relative to L if for each simplex A of $K \setminus L$, $st(A, K)$ is simplicially isomorphic to a star-convex flat polyhedron in $\mathbb{R}^n$ by a map which takes geodesics to straight lines.

▶ **Definition 3.2.** Let L be a subcomplex of K containing ∂K and let αK be a subdivision of K which agrees with K on L. Let $\beta_r^\alpha K$ be the subdivision of K such that, if A is a simplex in L or $dim(A) \leq r$, then $\beta_r^\alpha A = \alpha A$. If A is not in L and $dim(A) > r$ then $\beta_r^\alpha A = a \star \beta_r^\alpha \partial \alpha A$, i.e. it is subdivided as the cone on the already defined subdivision of its boundary. Observe that $\beta_n^\alpha K$ is αK while $\beta_0^\alpha K = \beta_L K$ is a derived subdivision of K relative to L.

▶ **Lemma 3.3.** *Let K be a locally geodesically-flat simplicial complex relative to a subcomplex L which contains ∂K. Let αK be a geometric subdivision of K which agrees with K on L.*

Then there exists $s \in \mathbb{N}$ for which $\beta^s \alpha K$ is related to $\beta^s K$ by bistellar moves which keep $\beta^s L$ fixed.

Proof. For A a positive dimensional r-simplex in $K \setminus L$, $st(A, \beta_r^\alpha K)$ is a strictly star-convex subset of $st(A, K)$. As K is locally geodesically-flat relative to L, there exists a geodesic embedding taking $st(A, \beta_r^\alpha K)$ to a strictly star-convex flat polyhedron of $\mathbb{R}^n$. By Theorem 2.3, $\beta^s st(A, \beta_r^\alpha K)$ is bistellar equivalent to $\beta^s C(\partial st(A, \beta_r^\alpha K))$. As A is not in L so no interior simplex of $st(A, \beta_r^\alpha K)$ is in L and consequently these bistellar moves keep $\beta^s L$ fixed. Taking all simplexes A in $K \setminus L$ of dimension $r = n$, we get a sequence of bistellar moves taking $\beta^s \beta_r^\alpha K$ to $\beta^s \beta_{r-1}^\alpha K$. Ranging r from n down to 1, we inductively obtain a sequence of bistellar moves taking $\beta^s \alpha K = \beta^s \beta_n^\alpha K$ to $\beta^s \beta_L K = \beta^s \beta_0^\alpha K$, which keeps $\beta^s L$ fixed.

And finally, arguing as above with the trivial subdivision $\alpha K = K$, we get $\beta^s \beta_L K$ from $\beta^s K$ by bistellar moves which keep $\beta^s L$ fixed. ◀

The following simple observation allows us to treat the star of a simplex in a geometric triangulation as the linear triangulation of a star-convex polytope in $\mathbb{R}^n$ and bistellar moves in the manifold as bistellar moves of the polytope.

▶ **Lemma 3.4.** *Let K be a geometric simplicial triangulation of a spherical, hyperbolic or Euclidean n-manifold M and let L be a subcomplex of K containing ∂K. When M is spherical we require the star of each positive dimensional simplex of $K \setminus L$ to have diameter less than π. Then K is locally geodesically-flat relative to L.*

Proof. Let K be a geometric triangulation of M and let B be the interior of the star of a simplex in $K \setminus L$. As K is simplicial, B is an open n-ball.

When M is hyperbolic, let $\phi : B \to \mathbb{H}^n$ be the lift of B to the hyperbolic space in the Klein model. As geodesics in the Klein model are Euclidean straight lines (as sets) so ϕ is the required embedding.

When M is spherical, let D be the southern hemisphere of $\mathbb{S}^n \subset \mathbb{R}^{n+1}$, let T be the hyperplane $x_{n+1} = -1$ and let $p : D \to T$ be the radial projection map (gnomonic projection) which takes spherical geodesics to Euclidean straight lines. As B is small enough, lift B to D and compose with the projection p to obtain the required embedding ϕ from B to $T \simeq \mathbb{E}^n$.

When B is Euclidean let ϕ be the lift of B to $\mathbb{R}^n$, which is an isometry. ◀

It is known (Theorem 4(c) of [1]) that for simplicial complexes of dimension at least 5 the number of derived subdivisions required to make the link of a vertex combinatorially isomorphic to a convex polyhedron is not (Turing machine) computable. So in particular, the stars of simplexes of a geometric triangulation may not even be combinatorially isomorphic to convex polyhedra, which is why we need to work with star-convex polyhedra instead.

Given a Riemannian manifold M, a *geometric polytopal complex* C of M is a finite collection of geometric convex polytopes whose union is all of M and such that for every $P \in C$, C contains all faces of P and intersection of two polytopes is a face of each of them.

Proof of 1.1. By Lemma 3.4, K_1 and K_2 are locally geodesically flat simplicial complexes. Let C be the geometric polytopal complex obtained by intersecting the simplexes of K_1 and K_2. Then $K = \beta_L C$, the derived subdivision of C relative to L is a common geometric subdivision of K_1 and K_2. By Lemma 3.3 then, there exists $s \in \mathbb{N}$ so that $\beta^s K_1$ and $\beta^s K_2$ are bistellar equivalent via $\beta^s K$ by bistellar moves which leave $\beta^s L$ fixed. ◀

References

1 Karim A. Adiprasito and Ivan Izmestiev. Derived subdivisions make every PL sphere polytopal. *Israel J. Math.*, 208(1):443–450, 2015. URL: https://doi.org/10.1007/s11856-015-1206-4, doi:10.1007/s11856-015-1206-4.

2 Jesús A. De Loera, Jörg Rambau, and Francisco Santos. *Triangulations*, volume 25 of *Algorithms and Computation in Mathematics*. Springer-Verlag, Berlin, 2010. Structures for algorithms and applications. URL: https://doi.org/10.1007/978-3-642-12971-1, doi:10.1007/978-3-642-12971-1.

3 Ivan Izmestiev and Jean-Marc Schlenker. Infinitesimal rigidity of polyhedra with vertices in convex position. *Pacific J. Math.*, 248(1):171–190, 2010. URL: https://doi.org/10.2140/pjm.2010.248.171, doi:10.2140/pjm.2010.248.171.

4 Tejas Kalelkar and Advait Phanse. An upper bound on pachner moves relating geometric triangulations, 2019. arXiv:1902.02163.

5 W. B. R. Lickorish. Unshellable triangulations of spheres. *European J. Combin.*, 12(6):527–530, 1991. URL: https://doi.org/10.1016/S0195-6698(13)80103-5, doi:10.1016/S0195-6698(13)80103-5.

6 Robert Morelli. The birational geometry of toric varieties. *J. Algebraic Geom.*, 5(4):751–782, 1996.

7 Francisco Santos. Geometric bistellar flips: the setting, the context and a construction. In *International Congress of Mathematicians. Vol. III*, pages 931–962. Eur. Math. Soc., Zürich, 2006.

8 Jaroslaw Wlodarczyk. Decomposition of birational toric maps in blow-ups & blow-downs. *Trans. Amer. Math. Soc.*, 349(1):373–411, 1997. URL: https://doi.org/10.1090/S0002-9947-97-01701-7, doi:10.1090/S0002-9947-97-01701-7.

9 E. C. Zeeman. *Seminar on Combinatorial Topology*. Institut Des Hautes Etudes Scientifiques, 1963.

Efficiently stabbing convex polygons and variants of the Hadwiger-Debrunner (p, q)-theorem

Justin Dallant[1] **and Patrick Schnider**[2]

1 **Department of Computer Science, ETH Zürich**
`jdallant@student.ethz.ch`
2 **Department of Computer Science, ETH Zürich**
`patrick.schnider@inf.ethz.ch`

Abstract

Hadwiger and Debrunner showed that for families of convex sets in $\mathbb{R}^d$ with the property that among any p of them some q have a common point, the whole family can be stabbed with $p-q+1$ points if $p \geq q \geq d+1$ and $(d-1)p < d(q-1)$. This generalizes a classical result by Helly. We show how such a stabbing set can be computed for n convex polygons of constant size in the plane in $\mathcal{O}((p-q+1)n^{4/3}\log^{2+\epsilon}(n)+p^3)$ expected time. For convex polyhedra in $\mathbb{R}^3$, the method yields an algorithm running in $\mathcal{O}((p-q+1)n^{13/5+\epsilon}+p^4)$ expected time. We also show that analogous results of the Hadwiger and Debrunner (p, q)-theorem hold in other settings, such as convex sets in $\mathbb{R}^d \times \mathbb{Z}^k$ or abstract convex geometries.

1 Introduction

A classical result in convex geometry by Helly [10] states that if a collection of convex sets in $\mathbb{R}^d$ is such that any $d+1$ sets have a common intersection, then all sets do. In 1957, Hadwiger and Debrunner [9] considered a generalization of this setting. Let $\mathcal{F}$ be a collection of sets in $\mathbb{R}^d$ and let $p \geq q \geq d+1$ be integers. We say that $\mathcal{F}$ has the (p, q)-property if $|\mathcal{F}| \geq p$ and for every choice of p sets in $\mathcal{F}$ there exist q among them which have a common intersection. We further say that a set of points S stabs $\mathcal{F}$ if every set in $\mathcal{F}$ contains at least one point from S. Then the following holds:

▶ **Theorem 1.1** (Hadwiger and Debrunner). *Let $d \geq 1$ be an integer. Let p and q be integers such that $p \geq q \geq d+1$ and $(d-1)p < d(q-1)$, and let $\mathcal{F}$ be a finite family of convex sets in $\mathbb{R}^d$. Suppose that $\mathcal{F}$ has the (p, q)-property. Then $\mathcal{F}$ can be stabbed with $p-q+1$ points in $\mathbb{R}^d$.*

In 1992 Alon and Kleitman [3] proved that for any $p \geq q \geq d+1$, there exists a finite bound on the maximum number of points needed to stab a collection of convex sets with the (p, q)-property. However, the known bounds are probably far from being tight. There is a lot of work in this more general setting, both improving the bounds (e.g. [13]) as well as adapting to generalizations of convex sets (e.g. [12, 16]), and it is an interesting open problem to study algorithmic questions connected to these results. However, in this work we will focus on the setting of Theorem 1.1, where the bound is tight, and show how such a stabbing set can be efficiently computed for convex polytopes in dimensions 2 and 3. To make the presentation simpler, we focus on polytopes of constant size, although the techniques used can be partially adapted to work for polytopes of arbitrary size.

Helly's theorem has also been generalized to many other settings. In general, we say that a set system has *Helly number* h if the following holds: if any h sets in the set system have a common intersection, then the whole set system does. Helly numbers have been shown to exist for many set systems, such as convex sets in $\mathbb{R}^d \times \mathbb{Z}^k$ [4, 11] or abstract convex

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

geometries (see [8] or Chapter III of [14]), which include subtrees of trees and ideals of posets. We will show that under some weak conditions, the existence of a Helly number implies a tight Hadwiger-Debrunner type result.

All of the skipped proofs and details can be found in the full version of this paper [7]

2 Stabbing convex polytopes

2.1 A proof of the Hadwiger-Debrunner (p,q)-theorem

We will first consider a proof of Theorem 1.1, taken from [15], which will naturally lead to an algorithm for finding stabbing points. The proof makes use of a lemma which can also be found in [15]. For a non-empty compact set S, let $lexmin(C)$ denote its lexicographical minimum point. Then we have the following:

▶ **Lemma 2.1** ([15]). *Let $\mathcal{F}$ be family of at least $d+1$ convex compact sets in $\mathbb{R}^d$, such that $I := \bigcap \mathcal{F}$ is non-empty. Let $x := lexmin(I)$. Then, there exist a subfamily $\mathcal{H} \subset \mathcal{F}$ of size d such that $x = lexmin(\bigcap \mathcal{H})$.*

We now sketch the idea of the proof of the Hadwiger-Debrunner theorem.

Proof idea of the Hadwiger-Debrunner theorem. Call a pair of integers (p, q) *admissible* if $p \geq q \geq d+1$ and $(d-1)p < d(q-1)$. Let (p, q) be an admissible pair, and let $\mathcal{F}$ be a family of compact convex sets in $\mathbb{R}^d$ with the (p, q)-property. Construct a point $x^*(\mathcal{F})$ defined as the lexicographically maximum point among all lexicographically minimum points in the intersection of d sets in $\mathcal{F}$. We choose it as one of our stabbing points. Now, remove all the sets stabbed by this point. It can be shown that the remaining sets either satisfy the $(p-d, q-d+1)$-property, where $(p-d, q-d+1)$ is admissible, or it consists of $p-q+k$ sets where some $k+1$ of them have a common intersection. In the first case, we can continue inductively, in the second case we can trivially stab the remaining sets using $p-q$ points. ◀

This proof naturally leads to an algorithm. In the following, we will assume that the convex sets are n polytopes of constant size. Similar ideas still work for general polytopes. Computing $x^*(\mathcal{F})$ can be done in $\mathcal{O}(n^d)$ time by computing all d-wise intersections, and it needs to be computed at most $p-q+1$ times. For the case where there are only $p-q+k$ sets remaining, it can be deduced from the proof the $p-q$ remaining stabbing points can be computed in $\mathcal{O}(p^{d+1})$ time. Thus, in the plane, we get a total runtime of $\mathcal{O}((p-q+1)n^2+p^3)$.

If p (and thus q), is small compared to n, the first term is the bottleneck in the computation time. In the following, we will show how to improve the runtime of this first part. The second term can be improved to $\mathcal{O}(p^2 \log p)$ by adapting the Bentley-Ottmann sweep line algorithm [5]. It is an interesting open problem whether further improvements are possible.

2.2 A more efficient algorithm for the planar case

We will now present an algorithm running in subquadratic time with respect to n. In this whole section, the collection $\mathcal{F}$ consists of n constant-size compact convex polygons in the plane and has the (p, q)-property, for some admissible pair (p, q).

In [6], Chan discovered a simple but remarkably powerful technique to reduce many optimization problems to a corresponding decision problem, with no blow-up in expected runtime. He proves the following lemma (stated in a slightly more general form here):

▶ **Lemma 2.2.** *Let $\alpha < 1$ and r be fixed constants. Suppose $f : \mathcal{P} \to \mathcal{Q}$ is a function that maps inputs to values in a totally ordered set (where elements can be compared in constant time), with the following properties:*

(1) For any input $P \in \mathcal{P}$ of constant size, $f(P)$ can be computed in constant time.

(2) For any input $P \in \mathcal{P}$ of size n and any $t \in \mathcal{Q}$, we can decide $f(P) \leq t$ in time $T(n)$.

(3) For any input $P \in \mathcal{P}$ of size n, we can construct inputs $P_1, \dots, P_r \in \mathcal{P}$ each of size at most $\lceil \alpha n \rceil$, in time no more than $T(n)$, such that $f(P) = max\{f(P_1), \dots, f(P_r)\}$.

Then for any input $P \in \mathcal{P}$ of size n, we can compute $f(P)$ in $\mathcal{O}(T(n))$ expected time, assuming that $T(n)/n^\epsilon$ is monotone increasing for some $\epsilon > 0$.

We can apply this technique to the computation of x^*. Here, each $P \in \mathcal{P}$ is a set of polygons, $\mathcal{Q}$ is the plane with lexicographical order, and f is x^* (which we define as $(-\infty, -\infty)$ if the intersection of the considered polygons is empty). For the sake of simplicity, we will assume that all points defined as the lexicographical minimum in the intersection of a pair of sets have different x-coordinates. We make the following observations:

1. For $\mathcal{F}$ a constant number of polygons, $x^*(\mathcal{F})$ can be computed in constant time by computing the lexicographical minimum of all pairs in $\mathcal{F}$. This verifies property (1).
2. For any $\mathcal{F}$ of size n, we can partition it into 3 disjoint subcollections S_1, S_2, S_3 of size between $\lfloor n/3 \rfloor$ and $\lceil n/3 \rceil$ each. Then, let $\mathcal{F}_1 := S_2 \cup S_3$, $\mathcal{F}_2 := S_1 \cup S_3$ and $\mathcal{F}_3 := S_1 \cup S_2$. Every set $\mathcal{F}_i$ is of size $|\mathcal{F}_i| \leq \lceil n \cdot 2/3 \rceil$. Moreover, every pair of sets of $\mathcal{F}$ appears in one $\mathcal{F}_i$. Thus, $x^*(\mathcal{F})$ is the lexicographical maximum of $\{x^*(\mathcal{F}_1), x^*(\mathcal{F}_2), x^*(\mathcal{F}_3)\}$. These collections can be constructed in $\mathcal{O}(n)$ time. This verifies property (3), assuming that $T(n) \in \Omega(n)$ (which it will be).

Thus, in order to apply Chan's framework, it remains to decide $x^*(\mathcal{F}) \leq_{lex} t$ for any t in subquadratic time. We can rephrase this as deciding whether there exist two intersecting sets in $\mathcal{F}$ whose intersection lies entirely to the right of a vertical line ℓ with x-coordinate t. We can make some simple observations to discard some of the sets in $\mathcal{F}$ in $\mathcal{O}(np)$ time:

- All sets which lie entirely to the left of ℓ can be safely ignored.
- If there are at least p sets lying entirely to the right of l, then by the (p, q)-property some $q > 2$ of them intersect and we can already answer the question in the affirmative.
- If there are fewer than p sets lying entirely to the right of l, then one can test in $\mathcal{O}(pn)$ whether the intersection of any of those with any other set in $\mathcal{F}$ lies entirely to the right of ℓ.

We are then left with answering the following question, which we define as the *Right Intersection Problem*.

▶ **Problem 2.3** (Right Intersection Problem)**.** *Given n compact convex polygons of constant size and a vertical line ℓ intersecting all polygons, decide if there exist two polygons whose intersection is non-empty and lies strictly to the right of ℓ.*

Solving this problem trivially in quadratic time and applying Lemma 2.2 would result in no improvement in the runtime. However, we can solve it in subquadratic time.

▶ **Proposition 2.4.** *The Right Intersection Problem can be decided in $O(n^{4/3} \log^{2+\epsilon}(n))$ time, for any constant $\epsilon > 0$.*

Proof. It was shown in [1] that counting the number of pairwise intersections between n convex polygons of constant size can be done in $\mathcal{O}(n^{4/3}\log^{2+\epsilon}(n))$ time.

For any instance of the Right Intersection Problem, we can cut all polygons along ℓ and discard the parts lying on the left of ℓ in linear time.

The answer to the original instance is positive if and only if there are two polygons which have a non-empty intersection but do not intersect on ℓ. This can be decided by counting the number of pairwise intersecting polygons in $\mathcal{O}(n^{4/3}\log^{2+\epsilon}(n))$ time, counting the number of pairwise intersecting polygons on ℓ (this can be done in $\mathcal{O}(n\log(n))$ time), and then comparing these numbers. They differ if and only if some pair of polygons intersect exclusively to the right of ℓ. ◀

We can thus use Lemma 2.2 to get the following:

▶ **Proposition 2.5.** *Let (p, q) be an admissible pair for $d = 2$ and let $\mathcal{F}$ be a family of n constant size compact convex polygons in the plane with the (p, q)-property. Then we can compute a set of at most $p - q + 1$ points stabbing $\mathcal{F}$ in expected time*

$$\mathcal{O}((p - q + 1)n^{4/3}\log^{2+\epsilon}(n) + p^3).$$

A similar method (using the method found in [1] for counting intersections of 3D convex polyhedra) can be applied for n convex polyhedra in $\mathbb{R}^3$, with a runtime of $\mathcal{O}((p - q + 1)n^{13/5+\epsilon} + p^4)$.

3 Other Hadwiger-Debrunner type results

By taking a close look at our proof for the Hadwiger-Debrunner (p, q)-theorem, we can observe that we made use of relatively few properties of compact convex sets. These properties are (i) closure under intersection, (ii) existence of a lexicographically minimum point, (iii) Helly's theorem as well as (iv) the fact that the set of all points lexicographically smaller than some point y is convex (this last property doesn't appear explicitly but is needed in the proof of Lemma 2.1). We define *Ordered-Helly systems* as set systems with analogue properties:

▶ **Definition 3.1** (Ordered-Helly system)**.** An Ordered-Helly system $\mathfrak{S}$ is a tuple $(\mathcal{B}, \mathcal{C}, \mathcal{D}, h, \preceq)$ consisting of a *base set* $\mathcal{B}$ with a total order $\preceq$, a set $\mathcal{C} \subset \mathcal{P}(\mathcal{B})$, whose members are called *convex sets*, a set $\mathcal{D} \subset \mathcal{C}$, whose members are called *compact sets*, and an integer $h \geq 2$, called the *Helly-number* of $\mathfrak{S}$, with the following properties:

1. $\mathcal{D}$ is closed under intersections (Intersection closure);
2. For all non-empty $S \in \mathcal{D}$, there exists $x \in S$ which is minimal with respect to $\preceq$ (Attainable minimum);
3. If $\mathcal{F} \subset \mathcal{C}$ is a finite family of sets in $\mathcal{C}$ where any h members of $\mathcal{F}$ have non-empty common intersection, then all of $\mathcal{F}$ has non-empty intersection (Helly property);
4. For all $t \in \mathcal{B}$, we have $\{x \in \mathcal{B} \mid x \preceq t \text{ and } x \neq t\} \in \mathcal{C}$ (Convex order).

We can thus carry out an analogue proof and derive (p, q)-theorems in these systems. A similar algorithm to the previous case can also be used to stab such a system, assuming we have access to a few oracles. Let h be the Helly number of an Ordered-Helly system. We say that a pair of integers (p, q) is h-admissible if $p \geq q \geq h$ and $(h - 2)p < (h - 1)(q - 1)$. We then get an analogue of Lemma 2.1 as well as the following:

▶ **Theorem 3.2.** *Let $\mathfrak{S}$ be an Ordered-Helly system. Let (p, q) be an h-admissible pair and let $\mathcal{F}$ be a family of non-empty sets in $\mathcal{D}$ with the (p, q)-property. Then $\mathcal{F}$ can be stabbed with $p - q + 1$ elements of $\mathcal{B}$.*

It should be mentioned that the existence of a Helly number alone is not enough to show such a result, see [2] for an example of a set system with Helly number 2 but no general (p, q)-theorem.

It can be shown that both convex sets in $\mathbb{R}^d \times \mathbb{Z}^k$ as well as abstract convex geometries are Ordered-Helly sytems, so we immediately get (p, q)-theorems for all of them. In the following, we give a non-exhaustive list of results that can be obtained this way:

▶ **Corollary 3.3** (Mixed-Integer Hadwiger-Debrunner)**.** *Let $d, k \geq 0$. Let (p, q) be a $((d+1)2^k)$-admissible pair. Let $\mathcal{F}$ be a finite family of convex sets in $\mathbb{R}^d \times \mathbb{Z}^k$ with the (p, q)-property. Then $\mathcal{F}$ can be stabbed with $p - q + 1$ points in $\mathbb{R}^d \times \mathbb{Z}^k$.*

▶ **Corollary 3.4.** *Let T be a tree and let $\mathcal{F}$ be a collection of subtrees of T (represented as sets of vertices). Let (p, q) be a 2-admissible pair. Let $F \subset \mathcal{N}$ be a family of non-empty subtrees of T with the (p, q)-property. Then $\mathcal{F}$ can be stabbed with $p - q + 1$ vertices.*

For a finite poset $(E, \leq)$, we say that a set $S \subset E$ is an ideal of E if for all $x \in S$ and all $y \in E$, $y \leq w \Rightarrow y \in S$. Let $h(E)$ be the maximum length of an antichain in E.

▶ **Corollary 3.5.** *Let $(E, \leq)$ be a finite poset. Let (p, q) be an $h(E)$-admissible pair and let $\mathcal{F}$ be a family of non-empty ideals of E with the (p, q)-property. Then $\mathcal{F}$ can be stabbed with $p - q + 1$ elements of E.*

References

1. Pankaj K. Agarwal, Mark de Berg, Sariel Har-Peled, Mark H. Overmars, Micha Sharir, and Jan Vahrenhold. Reporting intersecting pairs of convex polytopes in two and three dimensions. *Computational Geometry*, 23(2):195 – 207, 2002. URL: http://www.sciencedirect.com/science/article/pii/S0925772102000494, doi:https://doi.org/10.1016/S0925-7721(02)00049-4.
2. Noga Alon, Gil Kalai, Jiří Matoušek, and Roy Meshulam. Transversal numbers for hypergraphs arising in geometry. *Advances in Applied Mathematics*, 29(1):79–101, 2002.
3. Noga Alon and Daniel J. Kleitman. Piercing convex sets. *Bulletin of the American Mathematical Society*, 27(2):252–257, August 1992. URL: https://doi.org/10.1090/s0273-0979-1992-00304-x, doi:10.1090/s0273-0979-1992-00304-x.
4. Gennadiy Averkov and Robert Weismantel. Transversal numbers over subsets of linear spaces. *Advances in Geometry*, 12, 02 2012. doi:10.1515/advgeom.2011.028.
5. Jon Bentley and Thomas Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Transactions on Computers*, C-28(9):643–647, Sep. 1979. doi:10.1109/TC.1979.1675432.
6. Timothy M. Chan. Geometric applications of a randomized optimization technique. *Discrete & Computational Geometry*, 22(4):547–567, December 1999. URL: https://doi.org/10.1007/pl00009478, doi:10.1007/pl00009478.
7. Justin Dallant and Patrick Schnider. Efficiently stabbing convex polygons and variants of the Hadwiger-Debrunner (p, q)-theorem, 2020. arXiv:2002.06947.
8. Paul H. Edelman and Robert E. Jamison. The theory of convex geometries. *Geometriae Dedicata*, 19(3), December 1985. URL: https://doi.org/10.1007/bf00149365, doi:10.1007/bf00149365.
9. Hugo Hadwiger and Hans Debrunner. Über eine Variante zum Hellyschen Satz. *Archiv der Mathematik*, 8(4):309–313, oct 1957. URL: http://link.springer.com/10.1007/BF01898794, doi:10.1007/BF01898794.

10 Eduard Helly. Über Mengen konvexer Körper mit gemeinschaftlichen Punkten. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 32:175–176, 1923. URL: http://eudml.org/doc/145659.

11 Alan J. Hoffman. Binding constraints and Helly numbers. *Annals of the New York Academy of Sciences*, 319:284 – 288, 12 2006. doi:10.1111/j.1749-6632.1979.tb32803.x.

12 Andreas F Holmsen and Dong-Gyu Lee. Radon numbers and the fractional Helly theorem. *arXiv preprint arXiv:1903.01068*, 2019.

13 Chaya Keller, Shakhar Smorodinsky, and Gábor Tardos. Improved bounds on the Hadwiger–Debrunner numbers. *Israel Journal of Mathematics*, 225(2):925–945, 2018.

14 Bernhard Korte, Rainer Schrader, and László Lovász. *Greedoids*. Springer Berlin Heidelberg, 1991. URL: https://doi.org/10.1007/978-3-642-58191-5, doi:10.1007/978-3-642-58191-5.

15 Jiří Matoušek. *Lectures on Discrete Geometry*, volume 212 of *Graduate Texts in Mathematics*. Springer New York, New York, NY, 2002. URL: http://link.springer.com/10.1007/978-1-4613-0039-7, doi:10.1007/978-1-4613-0039-7.

16 Shay Moran and Amir Yehudayoff. On weak epsilon-nets and the Radon number. In *35th International Symposium on Computational Geometry (SoCG 2019)*, volume 129, page 51. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019.

Weak Unit Disk Contact Representations for Graphs without Embedding

Jonas Cleve[*1]

1 **Institut für Informatik, Freie Universität Berlin**
jonascleve@inf.fu-berlin.de

Abstract

Weak unit disk contact graphs are graphs that admit representing nodes as a collection of internally disjoint unit disks whose boundaries touch if there is an edge between the corresponding nodes. In this work we focus on graphs without embedding, i.e., the neighbor order can be chosen arbitrarily. We give a linear time algorithm to recognize whether a caterpillar, a graph where every node is adjacent to or on a central path, allows a weak unit disk contact representation. On the other hand, we show that it is NP-hard to decide whether a tree allows such a representation.

1 Introduction

A *disk contact graph* $G = (V, E)$ is a graph that has a geometric realization as a collection of internally disjoint disks mapped bijectively to the node set V such that two disks touch if and only if the corresponding nodes are connected by an edge in E. In an attempt to tackle the open problem of recognizing embedded caterpillars for disk contact graphs, *weak* disk contact graphs were introduced, which allow two disks to touch even if they don't share an edge. it was shown in this setting that the problem of recognizing embedded caterpillars is NP-hard by Chiu, Cleve, and Nöllenburg [2]. We continue this line of research by looking at graphs *without* embedding.

2 Recognizing Caterpillars in Linear Time

Similar to the algorithm by Klemz, Nöllenburg and Prutkin [3] we efficiently decide whether a caterpillar $G = (V, E)$, a graph where every node is adjacent to or lies on a central path, admits a weak unit disk contact representation (WUDCR). Let Δ be the maximum degree of G. If $\Delta \geq 7$ it is impossible to find a WUDCR: no unit disk can have more than six other adjacent unit disks. For $\Delta \leq 4$, G can even be realized as a (*strong*) UDCR [3], which is also a WUDCR. For $5 \leq \Delta \leq 6$ some caterpillars can be realized and some cannot; see Figure 1 for an example. This can be formalized as the following

▶ **Lemma 1.** *Let G be a caterpillar, $v_0, v_1, \ldots, v_k, v_{k+1}$ a longest path in G and $d_i = \deg(v_i)$ for all $1 \leq i \leq k$. Then G has a WUDCR iff for all $1 \leq \ell \leq k$: $\sum_{i=1}^{\ell} d_i \leq 4\ell + 2$.*

Proof (by induction). For $k = 1$ we have one node with d_1 leaves. The corresponding disk can have up to 6 neighboring disks and $d_1 \leq 4 + 2 = 6$.

Assuming the hypothesis holds for all $\ell < k$ we show that it holds for k. Look at Figure 2 for a depiction of the cases.

1. $d_k \leq 4$: up to 3 new leaves are added, no overlap with previous disks needed. It follows that $\sum_{i=1}^{k} d_i = \sum_{i=1}^{k-1} d_i + d_k \leq \sum_{i=1}^{k-1} d_i + 4 \overset{\text{IH}}{\leq} [4(k-1)+2] + 4 = 4k + 2$.

* Supported by ERC StG 757609.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

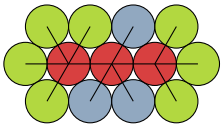


Figure 1 A caterpillar with $\Delta = 5$, red (internal) nodes having degrees 5, 5, and 4. It becomes unrealizable when adding another child to the rightmost internal (red) node, giving it degree 5 as well.

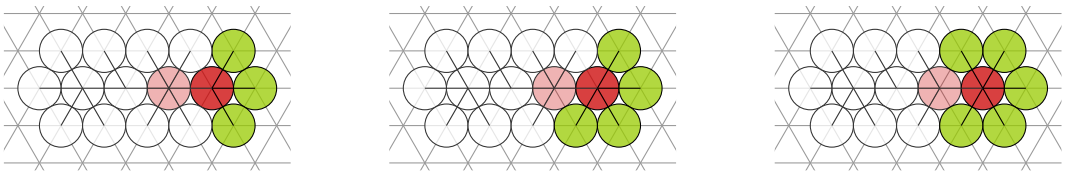


Figure 2 The three different cases from Lemma 1: $d_k \leq 4$ (left), $d_k = 5$ (center), and $d_k = 6$ (right).

2. $d_k = 5$: exactly 4 new leaves are added, taking away one position from the smaller construction, hence for $k-1$ it must hold that $\sum_{i=1}^{k-1} d_i \leq 4(k-1)+1$. It follows that $\sum_{i=1}^{k} d_i = \sum_{i=1}^{k-1} d_i + d_k = \sum_{i=1}^{k-1} d_i + 5 \leq [4(k-1)+1] + 5 = 4k+2$.
3. $d_k = 6$: exactly 5 new leaves are added, taking away two position from the smaller construction, hence for $k-1$ it must hold that $\sum_{i=1}^{k-1} d_i \leq 4(k-1)$. It follows that $\sum_{i=1}^{k} d_i = \sum_{i=1}^{k-1} d_i + d_k = \sum_{i=1}^{k-1} d_i + 6 \leq [4(k-1)] + 6 = 4k+2$. ◀

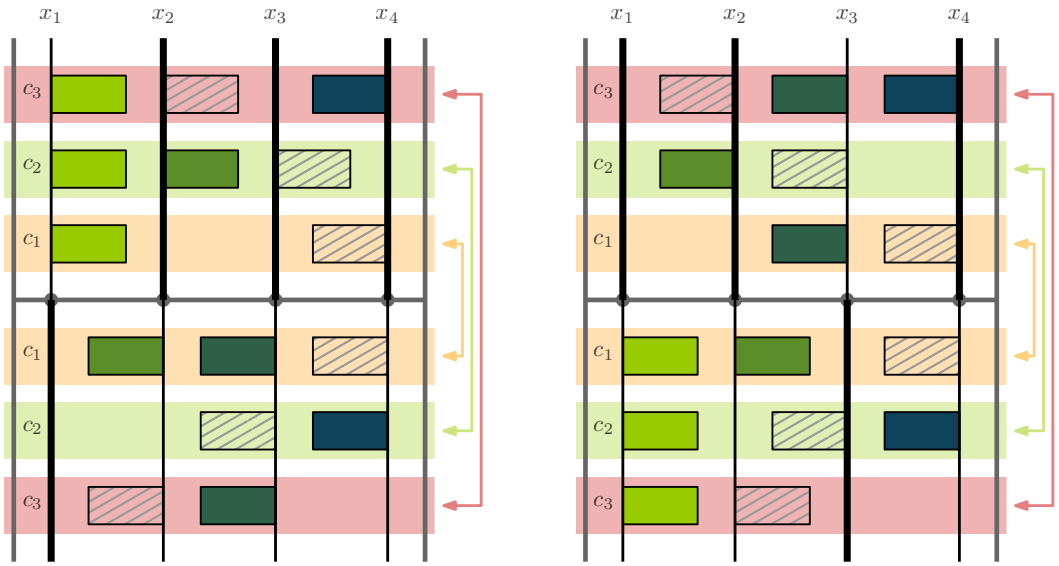
▶ **Theorem 2.** *It can be decided in linear time whether a caterpillar G admits a WUDCR.*

Proof. First determine a longest path $v_0, v_1, \ldots, v_k, v_{k+1}$ in linear time. Then check for all $1 \leq \ell \leq k$ whether $\sum_{i=1}^{\ell} d_i \leq 4\ell + 2$. This linear number of sums is easily checked in linear time. With Lemma 1 this immediately tells us whether a WUDCR for G exists. ◀

3 NP-hardness of Recognizing Trees

Recognizing whether a tree has a WUDCR is NP-hard—we use a reduction from Not-All-Equal-3SAT (NAE3SAT) [4] via a *logic engine construction* [1]. An instance for the NAE3SAT problem is a 3SAT formula and a yes-instance is a formula ϕ for which an assignment exists, which satisfies ϕ and additionally contains at least one false literal per clause. The logic engine construction, see Figure 3, works as follows: Given a formula with variables $x_1, \ldots, x_n$ and clauses $c_1, \ldots, c_m$ we construct an orthogonal drawing representing this formula. We have one horizontal spine to which one pole (consisting of a thick positive and thin negative part) for each variable is attached at its center. Each pole has m levels on the top and m on the bottom, each side representing the m clauses. We add a *flag* to the ith pole on the jth level as follows: **1.** If x_i appears as x_i in c_j we add a flag on the negative part, **2.** if x_i appears as $\neg x_i$ in c_j we add a flag on the positive part, and **3.** if x_i does not appear in c_j we add a flag on both parts (hatched in Figure 3). Two vertical poles are added, one on the left and one on the right. Note that in both realizations of Figure 3 there is one pole which is flipped. Otherwise it would not be drawable without overlap.

The question is now: Can the logic engine be drawn without overlap? For the drawing the variable poles can be flipped along their center and the flags can be drawn either left or right. In a non-overlapping drawing the leftmost pole puts its flags to the right and the



(a) $x_1 = 0$, $x_2 = 1$, $x_3 = 1$, and $x_4 = 1$.

(b) $x_1 = 1$, $x_2 = 1$, $x_3 = 0$, and $x_4 = 1$.

Figure 3 Two different logic engine realizations for the NAE3SAT formula with the three clauses $c_1 = (x_1, x_2, x_3)$, $c_2 = (x_1, \neg x_2, x_4)$, and $c_3 = (x_1, x_3, \neg x_4)$. Shaded flags correspond to literals which do not appear in a clause. For the poles: thicker means positive part, thinner means negative part. Note that in both cases there is one pole which is flipped.

rightmost one puts its flags to the left. Hence, every level j (top *and* bottom) needs at least one pole i *without* a flag on this level. The corresponding literal of x_i appears in c_j, fulfilling c_j. There cannot be a clause c_j with only positive literals—then the jth level on the bottom would have a flag on every variable pole; this is impossible without overlap. The upper part of the drawing finds a positive literal for each clause and the lower part finds a negative literal. Whether a variable pole was flipped for the drawing gives a direct correspondence to the assignment of its corresponding variable to 1 (not flipped) or 0 (flipped). Hence, the logic engine can be drawn without any overlaps if and only if the corresponding NAE3SAT formula is satisfiable. Constructing a logic engine with a WUDCR of trees gives a direct reduction from NAE3SAT and thus shows NP-hardness.

3.1 Rigid Hexagons as Basic Building Blocks

The goal is to model the logic engine structure by weak unit disk representations of trees. The weak model allows us to tightly pack disks, something we use heavily. The whole construction will live on a hexagonal grid with distance 2 (the diameter of a unit disk) between the grid points. The *grid distance* between two grid points is the number of edges on a shortest path—two touching disks have grid distance 1.

We will construct a tree which can only be realized as a hexagon and which can be chained together to form longer and rigid structures. For the chaining we need two special vertices which will always be on opposite corners of the resulting hexagon. In Figure 4 there are examples with various radii r (maximal grid distance to the center) which fulfill this criterion, as will be shown in

▶ **Lemma 3.** *All possible WUDCR of the trees in Figure 4 are hexagons where the (red) paths end on opposite corners of the hexagon.*

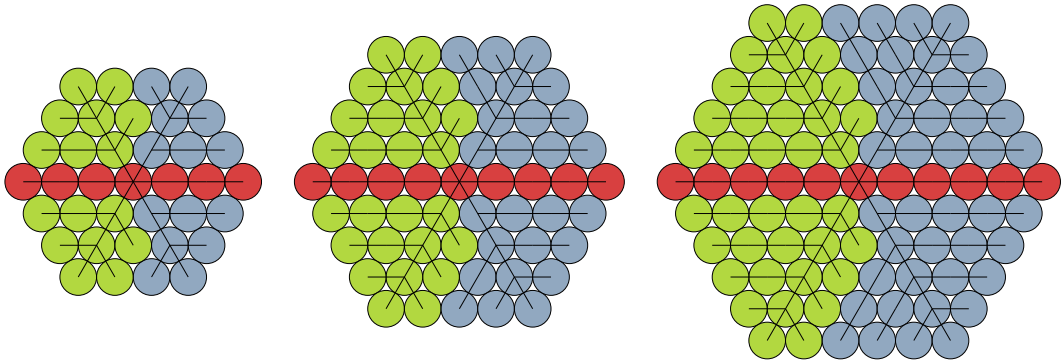
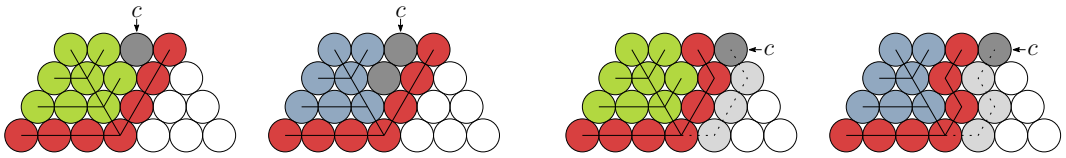


Figure 4 Hexagons with $r = 3, 4, 5$. Hexagons with arbitrary radii $r \geq 3$ can be constructed.



(a) Placing path segments on individual lines leaves at least one uncovered grid position c.

(b) Bending a path segment gives c an unreachable shortest path distance of 4.

Figure 5 Not placing both path segments on a common line leaves unreachable grid positions.

Proof. We show that the trees are always hexagons and that all red nodes lie on a line.

Observe that in Figure 4 a tree node has distance k to the root if and only if its corresponding disks has grid distance k from the center disk. Hence, we have exactly as many nodes with distance k from the root as we have positions with grid distance k from some fixed location. The root node with only its direct children is realized as a disk with six neighboring disks. This is a tight packing and w.l.o.g. we can assume that they lie on a hexagonal grid. Furthermore, all but the last level of the tree have at least one node with 3 children (green nodes in Figure 4); this forces them onto the hexagonal grid: 3 of the 6 neighboring positions are taken by the parent and two siblings, the other three by the children. As a result all nodes on this level are forced onto positions on the hexagonal grid and the result is a tight packing of disks which forms a hexagon.

Assume that not all red nodes are placed on a line. Look at Figure 5 where four such situations are depicted. Due to the structure of the green and blue subtrees, placing the disks as in Figure 5a leaves at least one grid position c empty. Placing the disks as in Figure 5b leaves the corner position c empty. A shortest path from the center disk to c (shown dotted) has distance $k + 1$ if k is c's grid distance to the center. There is no node with depth $k + 1$ in the tree—c is left empty. However, as all grid positions with grid distance up to the tree's height are covered, there is at least one node whose disk cannot be placed without overlap.

We conclude that all red nodes are forced on a line which places the two leaf nodes on opposite corners of the hexagon. ◀

We say that the trees for the hexagons have only one *distinguishable* WUDCR: It means that the placement of the important nodes (where something else will be connected to) does not change, but the placement of the other nodes may.

Plugging two hexagons together at a common endpoint forces them to lie on the same line. With this we are able to construct longer straight paths. Additionally, the connected

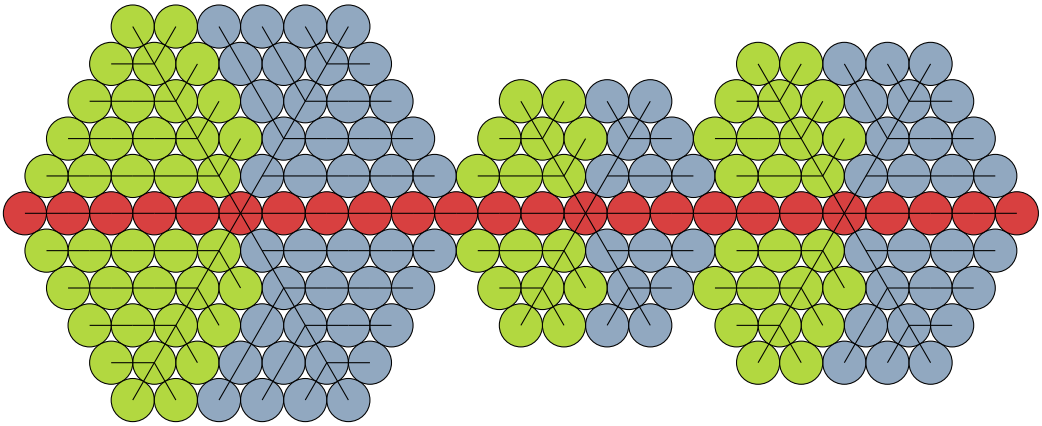


Figure 6 Connecting multiple hexagons, which can have different sizes. Observe, that the hexagons can even overlap more than one disk (center and right).

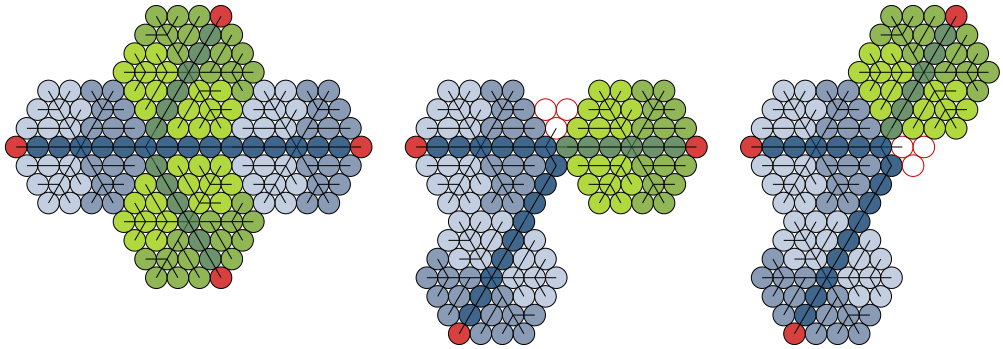


Figure 7 The branching gadget with 60° angle and interchangeable sides (left). Placing the horizontal part non-horizontally does not leave sufficient room for both branches (center, right).

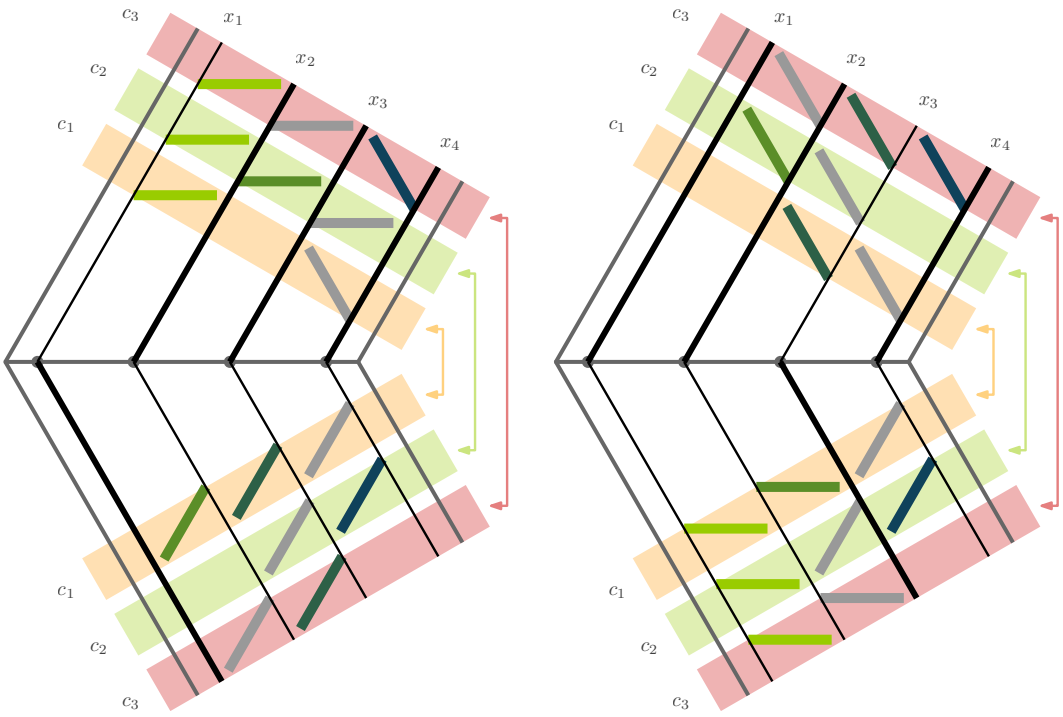
hexagons can differ in size or they can overlap by more than just one disk. See an example of three connected hexagons in Figure 6.

3.2 A Branching Gadget

Apart from going in a straight line, we need to be able to branch off two *branches* from a straight part (the *trunk*) to simulate the variable poles and flags. Additionally, it must allow for both branches to be interchanged to flip the variable poles or flags between two sides. As we could not branch orthogonally in a fully rigid way we will instead introduce a branching gadget which branches off at a 60° angle, see Figure 7.

▶ **Observation 4.** *The branching gadget in Figure 7 (left) has exactly two WUDCR which differ in the placement of the red vertices.*

Proof. From Lemma 3 we know that the four hexagons are rigid. Two hexagons and one path (ending in a hexagon) are connected to the leftmost hexagon. As shown in Figure 7 (left) it is possible to place the path horizontally. Placing the path differently, e.g. as in Figure 7 (center and right), leaves no space to fit both hexagons. Due to symmetry, both branching hexagons can be interchanged in the left case, giving two distinguishable WUDCR. ◀



(a) $x_1 = 0$, $x_2 = 1$, $x_3 = 1$, and $x_4 = 1$.

(b) $x_1 = 1$, $x_2 = 1$, $x_3 = 0$, and $x_4 = 1$.

Figure 8 Two logic engine realizations for the clauses $c_1 = (x_1, x_2, x_3)$, $c_2 = (x_1, \neg x_2, x_4)$, and $c_3 = (x_1, x_3, \neg x_4)$. It is a modification of Figure 3 where the branching angles are 60° instead of 90°.

3.3 Simulating the Logic Engine

The logic engine needs orthogonal branching and we only have branches at at 60° angle. Hence, it is necessary to modify the logic engine in a way to accommodate for such a difference. Figure 8 shows a modification of Figure 3 which only includes 60° angles. Flipping of the poles and flags happens by mirroring them along the line segment they are attached to. As can be seen, the clauses are still orthogonal to the variable poles s.t. two flags in the same free space are forced to overlap.

▶ **Theorem 5.** *It is* NP*-hard to decide whether a tree has weak unit disk contact representation.*

Proof. We model the logic engine from Figure 8 with the hexagon and branching gadgets to reduce NAE3SAT to our problem. See Figure 9 for an example of how a resulting drawing might look like. Apart from the two hexagons with radius 4 near the left and right end of the horizontal spine we only use hexagons with radius 3 and the branching gadget from before (also with radius 3 hexagons).

As shown in Figure 10 the distance between two variable poles and placement of the branching gadgets enforces that no two flags can be placed into the same free space. Furthermore, the left and right frame prevent flags from being drawn to the outside. The tree constructed from a boolean formula has a WUDCR if and only if no overlap occurs. This happens if and only if for every level j / clause c_j there is at least one flag less than the total number of variables; or to put it differently: if and only if not all three variables appearing in c_j place a flag on the bottom and not all on the top. This then gives a satisfying assignment of variables where not alle literals evaluate to 1.

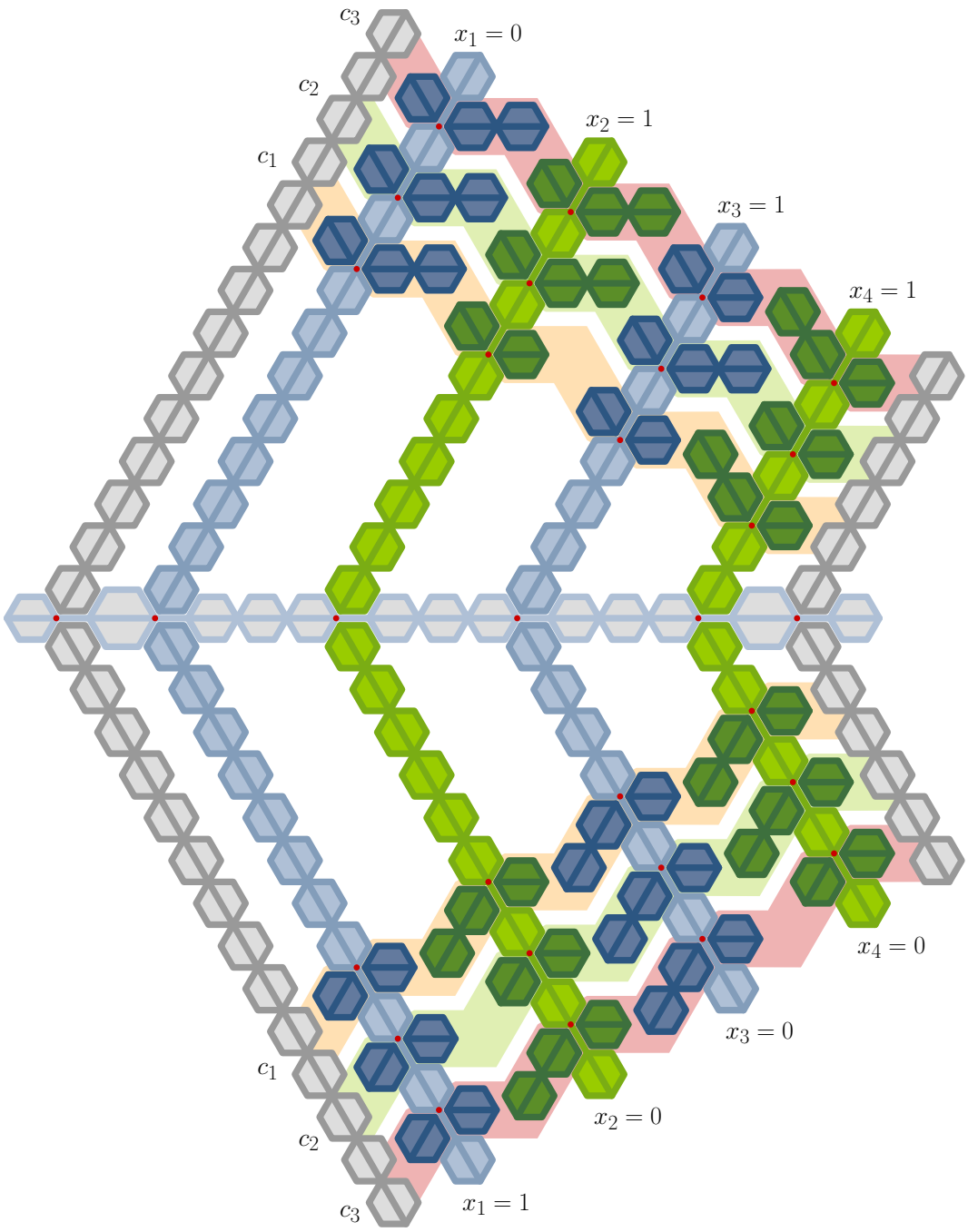


Figure 9 An example of a logic engine for the formula with clauses $c_1 = (x_1, x_2, x_3)$, $c_2 = (x_1, \neg x_2, x_4)$, and $c_3 = (x_1, x_3, \neg x_4)$. The variables are set to $x_1 = 0$, $x_2 = 1$, $x_3 = 1$, and $x_4 = 1$.

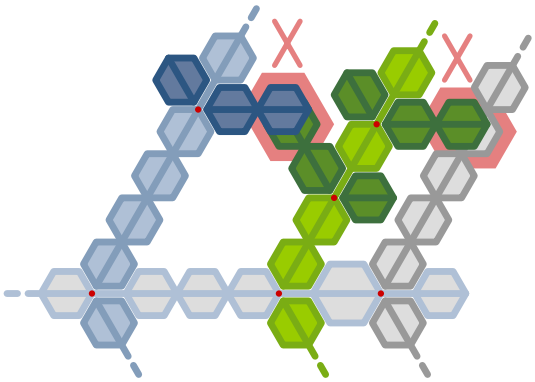


Figure 10 Possible cases of overlaps are highlighted: No two flags can be in the same place (left) and all flags on the outer variable poles must face inside (right).

The size of the construction is polynomial in the number of clauses m and variables n. There is a constant distance between two variable poles, hence, the size of the horizontal spine is $O(n)$. The further left a variable pole is the longer it has to be. The part without branching grows linearly in n (2 more hexagons per step to the left) and the branching part grows linearly in m, since each branching gadget with flag has constant size: the total size of one variable pole is $O(n+m)$. For n variable poles (and the two frames) this gives a total size of $O(n^2+mn)$ for the whole construction and it can be easily constructed in polynomial time. ◀

4 Conclusion

We showed that in linear time we can decide whether a caterpillar graph can be realized as a weak unit disk contact representation. On the other hand, the same problem is NP-hard for trees. The main open question remains whether lobster graphs (every node has distance at most 2 to a central path) can be recognized in polynomial time or whether it is NP-hard to recognize them. This can be generalized to look at trees where each node has a distance at most d from a central path.

Acknowledgments. The author wants to give special thanks to Sujoy Bhore, Man-Kwun Chiu, Soeren Nickel, and Martin Nöllenburg for the discussions and ideas during a research visit in Vienna which laid the groundwork for this paper.

References

1 Sandeep N. Bhatt and Stavros S. Cosmadakis. The complexity of minimizing wire lengths in VLSI layouts. *Information Processing Letters*, 25(4):263–267, 1987. doi:10.1016/0020-0190(87)90173-6.

2 Man-Kwun Chiu, Jonas Cleve, and Martin Nöllenburg. Recognizing embedded caterpillars with weak unit disk contact representations is NP-hard. In *Proceedings of the 35th European Workshop on Computational Geometry (EuroCG)*. URL: http://www.eurocg2019.uu.nl/papers/47.pdf.

3 Boris Klemz, Martin Nöllenburg, and Roman Prutkin. Recognizing weighted disk contact graphs. In Emilio Di Giacomo and Anna Lubiw, editors, *Graph Drawing and Network Visualization - 23rd International Symposium, GD 2015, Los Angeles, CA, USA, September*

24-26, 2015, Revised Selected Papers, volume 9411 of *Lecture Notes in Computer Science*, pages 433–446. Springer, 2015. doi:10.1007/978-3-319-27261-0_36.

4 Thomas J. Schaefer. The complexity of satisfiability problems. In Richard J. Lipton, Walter A. Burkhard, Walter J. Savitch, Emily P. Friedman, and Alfred V. Aho, editors, *Proceedings of the 10th Annual ACM Symposium on Theory of Computing, May 1-3, 1978, San Diego, California, USA*, pages 216–226. ACM, 1978. doi:10.1145/800133.804350.

On Hard Instances of the Minimum-Weight Triangulation Problem

Sándor P. Fekete[1], Andreas Haas[1], Yannic Lieder[1], Eike Niehs[1], Michael Perk[1], Victoria Sack[1], and Christian Scheffer[1]

1 **Department of Computer Science, TU Braunschweig, Germany**
{s.fekete, a.haas, y.lieder, e.niehs, m.perk, v.sack, c.scheffer}@tu-bs.de

Abstract

We present a study on the practical nature of the NP-hard problem of finding a Minimum Weight Triangulation (MWT) of a planar point set: Can we deliberately construct *practically* difficult instances? This requires identifying point sets for which all of a number of previously developed exact and heuristic methods *simultaneously* encounter a combination of pitfalls. We show that for instances of medium size, this seems unlikely, implying that one of several alternative methods may offer a path to an optimal solution. This complements recent work on the practical performance of these heuristic methods for specific classes of large benchmark instances, indicating that MWT problems may indeed be practically easier to solve than implied by its NP-hard complexity.

1 Introduction

The complexity of finding a minimum-weight triangulation (MWT) of a planar point set was a famous open problem for 27 years [8], until Mulzer and Rote [16] gave an NP-hardness proof, based in intricately constructed gadgets of considerable size.

While this shows that finding an MWT is difficult in a well-defined, theoretical sense, it does not necessarily imply that the problem is also intractable for instances of practically relevant size. In recent work, Haas [13] was able to extend, refine and streamline a number of previous ideas to compute provably optimal solutions for point sets of up to $30,000,000$ uniformly distributed points and real-world benchmark instances with up to $744,710$ points. This suggests that the MWT may indeed be much simpler than indicated by its theoretical complexity, at least for standard classes of instances.

We present a complimentary study on the practical nature of the theoretical hardness: Can we deliberately construct practically difficult instances of the MWT problem? This requires identifying point sets for which the previously developed methods *simultaneously* encounter a number pitfalls. We show that for instances of medium size, this seems unlikely, implying that one of several alternative methods may always provide a path to an optimal solution.

1.1 Related Work

There are efficient algorithms for computing optimal MWT solutions for special classes of instances. Independently, Gilbert [9] and Klincsek [15] showed that for simple polygons, the MWT problem can be solved in $O(n^3)$ time with dynamic programming. This can be generalized to polygons with k inner points. Hoffmann and Okamoto [14] showed how to obtain the MWT of such a point set in $O(6^k n^5 \log n)$ time. Grantson et al. [11] improved the algorithm to $O(n^4 4^k k)$ and showed another $O(n^3 k! k)$-time decomposition strategy [12].

For general instances, there are polynomial-time heuristics for including or excluding edges with certain properties from an MWT. Das and Joseph [4] showed that every edge in an MWT has the *diamond property*: For a point set S, an edge e cannot be in its minimum weight triangulation $\mathrm{MWT}(S)$ if both of the two isosceles triangles with base e and base

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

angle $\pi/8$ contain other points of S. Drysdale et al. [7] improved the angle to $\pi/4.6$. This property can exclude large portions of the edge set and works exceedingly well on uniformly distributed point sets, for which only an expected number of $O(n)$ edges remain. Dickerson et al. [5, 6] proposed the *LMT-skeleton heuristic*, which is based on a simple local-minimality criterion fulfilled by every edge in MWT(S). The LMT-skeleton algorithm often yields a connected graph, such that the remaining polygonal faces can be triangulated with dynamic programming to obtain the minimum weight triangulation.

The combination of the diamond property and the LMT-skeleton made it possible to compute the MWT for large, well-behaved point sets. Beirouti and Snoeyink [1] showed an efficient implementation of these two heuristics and reported that their implementation could compute the exact MWT of 40,000 uniformly distributed points in less than 5 minutes and even up to 80,000 points with the improved diamond property.

In more recent work, Haas [13] refined a number of these ideas. Based on a variety of improvements and additional data structures, he could compute provably optimal solutions for instances with up to 30,000,000 uniformly distributed points in less than 4 minutes on commodity hardware; the limiting factor turned out to be memory, not runtime. He achieved the same performance for normally distributed point sets, as well real-world benchmark instances from the TSPLIB [18] and the VLSI library of size up to 744,710 points. This shows that a wide range of huge MWT instances can be solved to provable optimality with the right combination of theoretical insight and algorithm engineering.

1.2 Our Results

We conduct a study of the practical difficulty of arbitrary MWT instances. In addition to the proven methods based on diamond property and LMT-skeleton, we present an integer program that strengthens Haas' toolbox by providing a practically useful alternative for determining optimal triangulation edges in unresolved faces. We also show that with this extended set of methods, any considered instance with up to 300 points can be solved to provable optimality within short time, even point sets deliberately constructed to be difficult.

2 Tools

Solving MWT instances to provably optimality relies on a number of different tools, which we briefly sketch in the following. The cited *Diamond Property* filters out a set of only $O(n)$ edges that *may* be in an MWT. The mentioned *LMT Skeleton* consists of a (possibly large) set of edges that *must* be contained in an MWT, but may still leave a number of untriangulated faces; see Figure 1. These faces can be triangulated with different versions of Dynamic Programming (Section 2.1) or with Integer Programming (Section 2.2).

2.1 Dynamic Programming (DP)

Empty faces of the LMT-skeleton can be triangulated using a dynamic programming approach for simple polygons, in time $\mathcal{O}(n^3)$ for an empty face with $n \in \mathbb{N}$ boundary vertices [10, 15]. For faces containing inner points, one of the following dynamic programming approaches can be used: A non-empty face with $n \in \mathbb{N}$ boundary vertices and $k \in \mathbb{N}$ inner points can be triangulated in $O(n^3 k! k)$ [12] or a non-empty face with $k \in \mathbb{N}$ connected components (resulting from the LMT-skeleton) can be triangulated in $O(n^{k+2})$ [19], respectively.

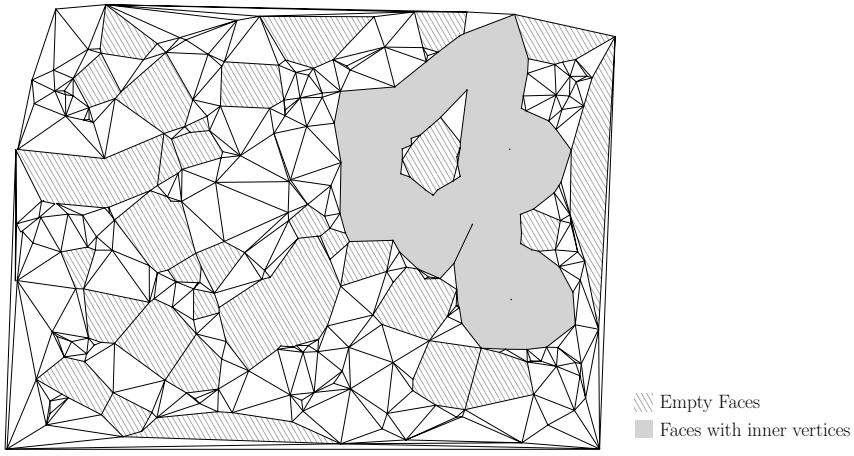


Figure 1 The LMT-skeleton (a subset of MWT(S)) of a point set S may contain untriangulated faces. These can be empty (hatched) or contain points and edges of the LMT-skeleton (filled).

2.2 Integer Programming

Another approach to compute the MWT of the remaining faces of the LMT-skeleton makes use of the following integer program (IP); see Yousefi and Young [20] as well as Dantzig, Hoffman and Hu [3] for related work. The objective function minimizes the sum of the perimeters $||\triangle||$ of all triangles, used in the triangulation. The variables $x_\triangle \in \{0,1\}$ indicate whether $\triangle$ is used in the triangulation. (Note that this description is slightly simplified because of limited space; a practically complete description provides additional adjustments for fixed edges along face boundaries.)

$$\min_{x_\triangle} \quad \sum_{\triangle} ||\triangle|| \cdot x_\triangle \tag{1}$$

$$\text{s.t.} \quad \sum_{\triangle \in \delta(e)} x_\triangle = 1 \qquad \forall e \in \text{boundary component} \tag{2}$$

$$\sum_{\triangle \in \delta^+(e)} x_\triangle = 1 \qquad \forall e \in \text{antennas} \tag{3}$$

$$\sum_{\triangle \in \delta^-(e)} x_\triangle = 1 \qquad \forall e \in \text{antennas} \tag{4}$$

$$\sum_{\triangle \in \delta^+(e)} x_\triangle - \sum_{\triangle \in \delta^-(e)} x_\triangle = 0 \qquad \forall e \in \text{inner edges} \tag{5}$$

$$x_\triangle \in \{0,1\} \tag{6}$$

Given a non-triangulated face, we distinguish three kinds of edges. *Boundary edges* lie on the outer face boundary or inner hole boundaries. Boundary edges are part of exactly one triangle in the face (Equation (2)). *Antenna edges* have the same face on both of its sides (see filled face in Figure 1), so they are part of two triangles in the face (Equation (3) and (4)). The third kind are *inner edges*, i.e., all remaining edges inside a face that are not fixed by the LMT-skeleton and not excluded by the diamond property. For these edges, the difference of the number of triangles on their left side equals the number of triangles on their right side (Equation (5)). The first three constraints imply either zero or one triangle on each side.

Equation 2 and Equation 5 are sufficient to solve the IP for a given polygon, with the boundary edges containing both the outer boundary and hole boundaries. Equation 3 and Equation 4 are auxiliary constraints, fixing edges of the LMT skeleton that have the same face on both sides.

3 Hard Instances

While previous studies on large classes of *specific* MWT instances showed that even huge instances can be solved optimally, this does not imply that there are *no* practically hard ones. For any NP-hard problem, natural candidates for such instances are the ones constructed in an NP-hardness reduction. However, the intricate constructions in the seminal proof by Mulzer and Rote [16] produce instances of tremendous size: While the clause gadgets have dimensions of $250,000 \times 250,000$, the connector gadgets (representing variable-clause incidences in a planar embedding of a 3SAT instance) require 14 points per subsegment of a length less than 28. As a consequence, representing even a Planar 1-in-3SAT instance with a handful of clauses (and thus, three handfuls of connector gadgets) easily requires millions of points. Given that "... modern SAT solvers can often handle problems with millions of constraints [i.e., clauses] and hundreds of thousands of variables" [17], it is clear that insufficient memory becomes a limiting factor long before the algorithmic difficulty of 3SAT.

This motivates the complimentary question to the results of [13]: Can we deliberately construct practically difficult instances of moderate size? It follows from the availability of the tools described in the previous section that such an instance must satisfy the following three properties.

1. It contains at least one complex face; otherwise it can be solved in $O(n^3)$.
2. The complex face must contain a relatively large number of connected components; otherwise, it can be solved in polynomial time with Dynamic Programming.
3. The Linear Programming relaxation of the IP for a face must yield a fractional optimal solution; otherwise, the IP is easy to solve.

We have employed a number of systematic methods to generate such instances. Figure 2 illustrates the workflow of an evolutionary strategy and a local perturbation algorithm. An example of how this leads to more complex instances can be seen in Figure 3.

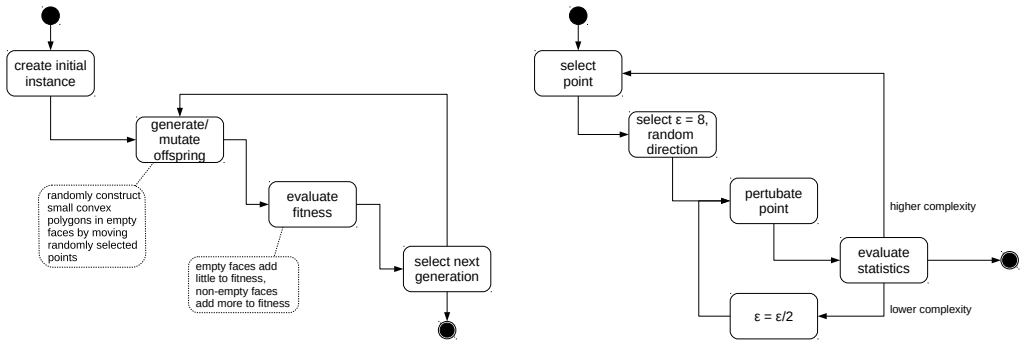


Figure 2 Modifying a point set to produce more complex LMT faces: **(Left)** Evolutionary strategy. **(Right)** Local Perturbation.

There are known classes of instances with a complex face of the LMT-skeleton that contain many connected components; see Figure 4. On the other hand, there are also known classes of

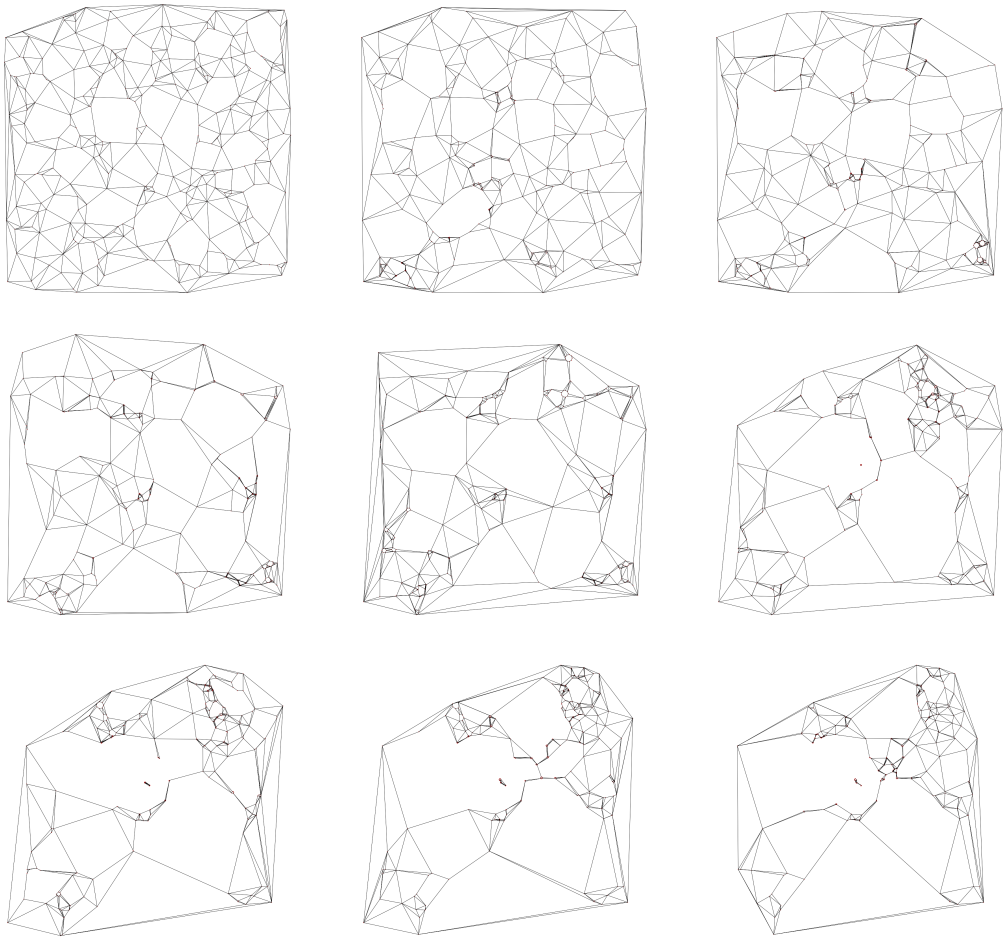


Figure 3 Evolving a point set to produce more complex LMT faces.

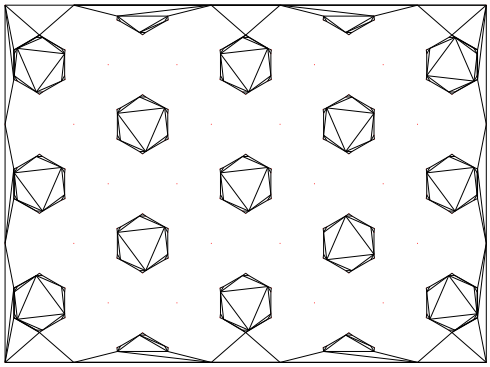


Figure 4 Instances for which the LMT-skeleton has a complex face with many connected components. Adapted from Belleville et al. [2].

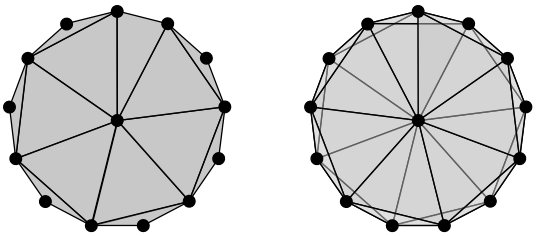


Figure 5 Instances for which the IP has fractional solutions. Adapted from Yousefi and Young [20].

instances with one complex face of the LMT-skeleton that produce fractional solutions when handled by the described integer program; see Figure 5. However, these instances are of quite different nature, so it is not clear that they can be combined for instances of reasonable size.

4 Experimental Results

We investigated the practical solvability of MWT instances, with a focus on constructing hard instances. All experiments were executed with CPLEX 12.9 on an Intel(R) Core(TM) i7-6700K CPU 4.00GHz with 4 cores and 8 threads utilizing an L3 Cache with 8MB, and a maximum amount of 64GB RAM. With the evolutionary strategy shown in Figure 2, we were able to generate many instances that contain at least one complex face. In order to generate the variables of the integer program (i.e. possible empty triangles within the complex face), we used a heuristic that performs well in practical scenarios. The heuristic does not guarantee to find empty triangles in every case. Therefore, we added a callback to the integer programs that verifies that all triangles of an integer solution are empty. In the upcoming figures, the optimization and verification time (callback time) are separated. We chose the instance size to be 306 points, which is comparable to the one shown in Figure 4. The goal was to produce complex faces that require large computational effort during the optimization of the integer program. After generating an instance with a complex face, we applied random local perturbations with respect to certain properties. The generation process was executed for several days, producing around $17,000$ instances.

We first studied the size of the complex boundary, which includes all edges on the boundary of the complex face, as well as hole boundary edges and antennas. Figure 6 (Left)

shows that the number of variables of the IP increases linearly with the size of the complex boundary. Moreover, the time to solve these instances (see Figure 6 (Right)) increases from 0.01 seconds to 0.25 seconds for complex faces with a boundary size of 300 edges.

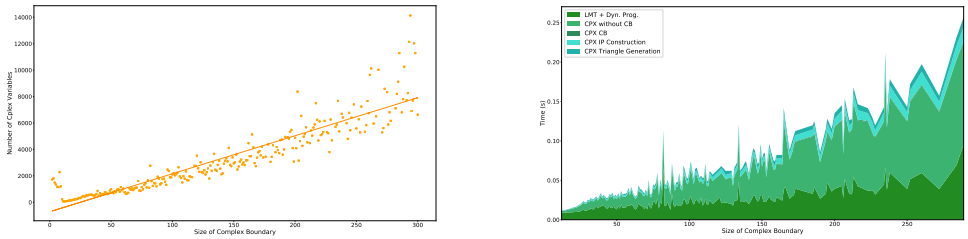


Figure 6 Results on instances that were generated with a large complex boundary size. **(Left)** Number of IP variables as a function of the size of the complex boundary. **(Right)** Runtime of the integer program. *LMT + Dyn. Prog.* refers to the construction of the LMT skeleton and triangulation of the empty faces. *CPX Triangle Generation* and *CPX IP Construction* represents the time that was necessary to generate the variables and constraints of the IP. *CPX (without) CB* refers to the runtime of the optimization and the empty triangle verification.

Next we investigated the number of complex faces in an instance. As shown in Figure 7 (Left), the number of IP variables for empty triangles grows linearly in the number of components. Figure 7 (Right) shows that runtimes for instances with < 5 complex components differ only by 0.1 seconds compared to instances with > 30 components.

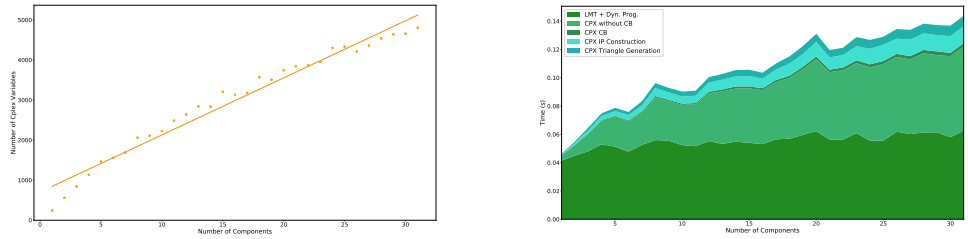


Figure 7 Results on instances that were generated with a large number of components. **(Left)** Number of IP variables as a function of the number of components. **(Right)** Runtime of the integer program. *LMT + Dyn. Prog.* refers to the construction of the LMT skeleton and triangulation of the empty faces. *CPX Triangle Generation* and *CPX IP Construction* represents the time that was necessary to generate the variables and constraints of the IP. *CPX (without) CB* refers to the runtime of the optimization and the empty triangle verification.

Despite the extensive length of the search, no instances with larger complex boundary sizes or more complex components were found. Therefore, we extended the instance from Figure 4 to produce instances with arbitrary numbers of complex boundary edges and connected components. Increasing the complex boundary size to more than $13,000$ edges showed that the runtime of the algorithm increases quadratically, see Figure 8. Further investigation showed that the optimal solution of the LP relaxation of the integer program for the produced instances was integral. Thus, only two of the three necessary malicious properties from the previous section could be established at once, so all instances could be

solved to provable optimality. In particular, only relatively degenerate instances similar to the one from Figure 5 seem to produce complex faces with non-integer LP solutions.

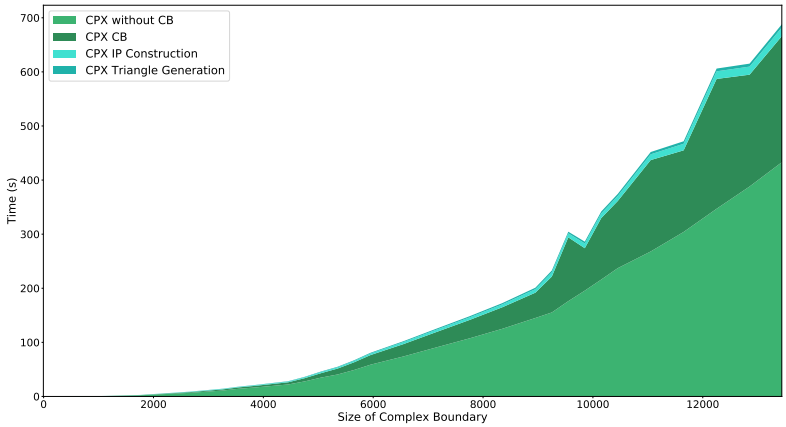


Figure 8 Runtime of the integer program for extensions of the instance in Figure 4. Note the moderate runtime despite the size: The largest IPs have more than 6,000,000 variables. *LMT + Dyn. Prog.* refers to the construction of the LMT skeleton and triangulation of the empty faces. *CPX Triangle Generation* and *CPX IP Construction* represents the time that was necessary to generate the variables and constraints of the IP. *CPX (without) CB* refers to the runtime of the optimization and the empty triangle verification.

5 Conclusions

Our systematic study for constructing practically difficult MWT instances showed that medium-sized point sets that simultaneously have three malicious properties seem hard to come by. This provides further evidence to the observation that the MWT problem is practically easier to solve than indicated by its theoretical complexity, as it shows that this practical solvability does not only depend on benign properties of special classes of instances, but remains intact even when we try to make instances deliberately difficult.

References

1 Ronald Beirouti and Jack Snoeyink. Implementations of the LMT Heuristic for Minimum Weight Triangulation. In *Proc. Symposium on Computational Geometry (SoCG)*, pages 96–105, 1998.

2 Patrice Belleville, Mark Keil, Michael McAllister, and Jack Snoeyink. On computing edges that are in all minimum-weight triangulations. In *Proc. Symposium on Computational Geometry (SoCG)*, pages V7–V8, 1996.

3 George B. Dantzig, Alan J. Hoffmann, and T.C. Hu. Triangulations (tilings) and certain block matrices. *Mathematical Programming*, 31:1–14, 1985.

4 Gautam Das and Deborah Joseph. Which triangulations approximate the complete graph? In *Proc. International Symposium on Optimal Algorithms*, pages 168–192, 1989.

5 Matthew Dickerson, J. Mark Keil, and Mark H. Montague. A Large Subgraph of the Minimum Weight Triangulation. *Discrete & Computational Geometry*, 18(3):289–304, 1997.

6 Matthew Dickerson and Mark H. Montague. A (Usually?) Connected Subgraph of the Minimum Weight Triangulation. In *Proc. Symposium on Computational Geometry (SoCG)*, pages 204–213, 1996.

7 Robert L. Scot Drysdale, Scott A. McElfresh, and Jack Snoeyink. On exclusion regions for optimal triangulations. *Discrete Applied Mathematics*, 109(1-2):49–65, 2001.

8 Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.

9 P. D. Gilbert. New results in planar triangulations. Master's thesis, University Illinois, 1979.

10 Peter D. Gilbert. New results on planar triangulations. Technical report, Illinois University at Urbana-Champaign, 1979.

11 Magdalene Grantson, Christian Borgelt, and Christos Levcopoulos. A Fixed Parameter Algorithm for Minimum Weight Triangulation: Analysis and Experiments. Technical Report LU-CS-TR: 2005-234, Lund University, Sweden, 2005.

12 Magdalene Grantson, Christian Borgelt, and Christos Levcopoulos. Minimum Weight Triangulation by Cutting Out Triangles. In *Proc. International Symposium on Algorithms and Computationa (ISAAC)*, pages 984–994, 2005.

13 Andreas Haas. Solving large-scale minimum-weight triangulation instances to provable optimality. In *Proc. Symposium on Computational Geometry (SoCG)*, pages 44:1–44:14, 2018.

14 Michael Hoffmann and Yoshio Okamoto. The minimum weight triangulation problem with few inner points. *Computational Geometry*, 34(3):149–158, 2006.

15 G.T. Klincsek. Minimal Triangulations of Polygonal Domains. *Annals of Discrete Mathematics*, 9:121–123, 1980.

16 Wolfgang Mulzer and Günter Rote. Minimum-weight triangulation is NP-hard. *Journal of the ACM*, 55(2):11, 2008.

17 Olga Ohrimenko, Peter J. Stuckey, and Michael Codish. Propagation = lazy clause generation. In Christian Bessière, editor, *Principles and Practice of Constraint Programming – CP 2007*, pages 544–558, 2007.

18 G. Reinelt. TSPLIB–A Traveling Salesman Problem Library. *ORSA Journal of Computing*, 3(4):376–384, 1991.

19 Chiu-fai Tsang. *Expected Case Analysis of [beta]-skeletons with Applications to the Construction of Minimum-weight Triangulations*. PhD thesis, Hong Kong University of Science and Technology, 1995.

20 Arman Yousefi and Neal E. Young. On a linear program for minimum-weight triangulation. *SIAM Journal on Computing*, 43(1):25–51, 2014.

Flips in higher order Delaunay triangulations*

Elena Arseneva[1], Prosenjit Bose[2], Pilar Cano[2,3], and Rodrigo I. Silveira[3]

1 **St. Petersburg State University, Russia**
e.arseneva@spbu.ru
2 **Carleton University, Canada**
jit@scs.carleton.ca
3 **Universitat Politècnica de Catalunya, Spain**
{m.pilar.cano, rodrigo.silveira}@upc.edu

Abstract

We study the flip graph of higher order Delaunay triangulations. A triangulation of a set S of n points in the plane is order-k Delaunay if the circumcircle of every triangle encloses at most k points of S in its interior. The *flip graph* of S has one vertex for each possible triangulation of S, and an edge connects two vertices when the two corresponding triangulations can be transformed into each other by a *flip* (i.e., exchanging the diagonal of a convex quadrilateral by the other one). We show that, even though the order-k flip graph might be disconnected for $k \geq 3$, any order-k triangulation can be transformed into some different order-k triangulation by at most $k-1$ flips, such that the intermediate triangulations are of order at most $2k-2$, in the following settings: (1) for any $k \geq 0$ when S is in convex position, and (2) for any point set S when $k \leq 5$.

1 Introduction

Given a set S of points in the plane, a *triangulation* of S is a decomposition of the convex hull of S into triangles, such that each triangle has its three vertices in S. Despite the well-known fact that a point set S in the plane can have many different triangulations [7], most of the time the *Delaunay triangulation* is used, since its triangles are considered "well-shaped". A Delaunay triangulation of S, denoted $DT(S)$, is a triangulation where each triangle satisfies the *empty circle property*: the circumcircle of each triangle does not enclose other points of S (for a survey, see [3, 8]). When no four points of S are co-circular, $DT(S)$ is unique. However, when used to model terrains as a 3D surface, the Delaunay triangulation of points on the surface ignores the elevation information, potentially resulting in poor terrain models where important terrain features, such as valley or ridge lines, are ignored [6, 10]. This motivated Gudmundsson et al. [9] to propose *higher order Delaunay triangulations*. A triangulation T of S is an *order-k Delaunay triangulation*—or, simply, *order-k*—if the circumcircle [1] of each triangle of T contains at most k points of S in its interior. As soon as $k > 0$, one obtains a class of triangulations that, intuition suggests, still has well-shaped triangles for small values of k, but with potentially many triangulations to choose from.

A fundamental operation to locally modify triangulations is the *edge flip*. It consists of removing the edge shared by two triangles that form a convex quadrilateral, and inserting the other diagonal of the quadrilateral. A flip transforms a triangulation T into another

* E.A. was supported by supported by RFBR, project 20-01-00488. P.B. was partially supported by NSERC. P.C. was supported by CONACYT, MX. R.S. was supported by MINECO through the Ramón y Cajal program. P.C. and R.S. were also supported by projects MINECO MTM2015-63791-R and Gen. Cat. 2017SGR1640. This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 734922.

[1] We refer to the *interior of a circumcircle.* as the interior of the disk defined by such circle

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

triangulation T' that differs by exactly one edge and two triangles. The flip operation leads naturally to the definition of the *flip graph* of S. Each triangulation of S is represented by a vertex in this graph, and two vertices are adjacent if their corresponding triangulations differ by exactly one flip. The importance of flips in triangulations comes from the fact that the flip graph is connected [11]. In fact, it is known that $O(n^2)$ flips are enough to convert any triangulation of S into $DT(S)$ [12, 15]. In general, computing the distance in the flip graph between two given triangulations is a difficult problem [13, 14]. This has drawn considerable attention to the study of certain subgraphs of the flip graph, which define the flip graph of certain classes of triangulations. We refer to [5] for a survey.

Almost nothing is known about the flip graph of order-k triangulations, except that it is connected only for $k \leq 2$ [1]. Similarly, Abellanas et al. [2] showed that the flip graph of triangulations of point sets with edges of order k [2] is connected for $k \leq 1$, but can be disconnected for $k \geq 2$. On the other hand, they proved that for point sets in convex position the flip graph is connected provided one allows intermediary triangulations of order at most $3k$ [2]. However, their proof implies an exponential bound on the diameter of the flip graph.

In this paper we present several structural properties of the flip graph of order-k triangulations. For points in convex position, we show that for any $k > 2$ there exists point sets in convex position for which the flip graph is not connected. However, we prove that for any order-k triangulation there exists another order-k triangulation at distance at most $k-1$ in the flip graph of order-$(2k-2)$ triangulations. This shows that, while order-k triangulations are not connected via the flip operation, they become connected if a slightly larger neighborhood is considered. For points in generic (non-convex) position, we prove the same result for up to $k \leq 5$, although we conjecture that it holds for all k. Our results have implications on the flip distance between order-k triangulations, as well as on their efficient algorithmic enumeration.

Due to space limitations, most proofs are omitted.

2 Preliminaries and general observations

Let S be a point set in the plane. The point set S is in *general position* if no three points of S lie on a line and no four points of S lie on a circle. For the rest of the paper we assume that the point set is in general position. Let T be a triangulation of S, and let $\triangle uyv$ be a triangle in T with vertices u,y,v. We will denote by $\bigcirc uyv$ the disk defined by the enclosed area of the *circumcircle* of $\triangle uyv$ (i.e., the unique circle going through u, y, and v), unless stated otherwise. Triangle $\triangle uyv$ is an *order-k triangle* if $\bigcirc uyv$ contains at most k points of S in its interior. A triangulation T where all triangles are order-k is an *order-k (Delaunay) triangulation*. Thus, T is not of order-k if $\bigcirc uxv$ contains more than k points in its interior for some $\triangle uxv$ in T. The set of all order-k triangulations of S will be denoted $\mathcal{T}_k(S)$.

Let $e = uv$ be an edge in T. Edge e is *flippable* if e is incident to two triangles $\triangle uxv$ and $\triangle uyv$ of T and $uxvy$ is a convex quadrilateral. The flippable edge e is *illegal* if $\bigcirc uxv$ contains y in its interior. Note that this happens if and only if $\bigcirc uyv$ contains x in its interior. Otherwise, it is called *legal*. The *angle-vector* $\alpha(T)$ of a triangulation T is the vector whose components are the angles of each triangle in T ordered in increasing order. Let $T' \neq T$ be another triangulation of S. We say that $\alpha(T) > \alpha(T')$ if $\alpha(T)$ is greater than $\alpha(T')$ in lexicographical order. It is well-known that if T' is the triangulation obtained by flipping an

[2] An edge uv is an *order-k edge* of S if there exists a disk that contains u and v on its boundary and at most k points of S in its interior. Observe that the edges of an order-k triangle are order-k edges.

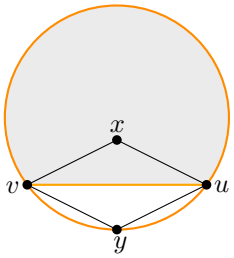


Figure 1 An illegal edge uv, with region $\circlearrowleft_y^{uv}$ in gray.

illegal edge of T, then $\alpha(T') > \alpha(T)$ [8]. Moreover, since $DT(S)$ maximizes the minimum angle, it follows that $DT(S)$ is the only triangulation where all the edges are legal [15]. This also implies that the flip graph is connected, since any triangulation can be transformed into the Delaunay triangulation. We will denote the flip graph of $\mathcal{T}_k(S)$ by $G(\mathcal{T}_k(S))$. Abe and Okamoto [1] observed that $G(\mathcal{T}_2(S))$ is connected as a consequence of the following lemma.

▶ **Lemma 2.1** (Abe and Okamoto [1]). *Let T be a triangulation of S, let uv be an illegal edge of T, and let $\triangle uvx$ and $\triangle uyv$ be the triangles incident to uv in T. If $\triangle uxv$ is of order k, and $\triangle uyv$ is of order l, then triangles $\triangle uxy$ and $\triangle xyv$ have orders k' and l', respectively, for k', l' with $k' + l' \leq k + l - 2$.*

When we refer to points in a certain region, we refer to points of S in that region.

For a triangle $\triangle uyv$, we will use $\circlearrowleft_y^{uv}$ to denote the open region bounded by edge uv and the arc of circle $\partial \bigcirc uyv$ that does not contain y. See Fig 1. Consider a triangulation T of order $k \geq 3$, and an illegal edge uv adjacent to triangles $\triangle uxv$ and $\triangle uyv$. Consider the triangulation T' resulting from flipping uv in T. Using that the interior of $\bigcirc uxv$ and $\bigcirc uyv$ contain at most k points each (including x and y), and the fact that the interior of $\bigcirc uxy$ contains at least $k + 1$ points, a rather simple counting argument implies the following.

▶ **Observation 2.2.** *If $\bigcirc uxy$ contains more than $k \geq 3$ points in its interior then each region $\circlearrowleft_y^{ux} \setminus \bigcirc uxv$ and $\circlearrowleft_x^{uy} \setminus \bigcirc uyv$ contains at least 2 points.*

3 Points in convex position

First, we show that $G(\mathcal{T}_k(S))$ may not be connected and $k - 1$ flips may be necessary to transform a triangulation in $\mathcal{T}_k(S)$ into some different order-k triangulation for $k > 2$.

▶ **Theorem 3.1.** *For any $k > 2$ there is a set S_k of $2k + 2$ points in convex position such that $G(\mathcal{T}_k(S_k))$ is not connected. Moreover, there is a triangulation T_k in $\mathcal{T}_k(S_k)$ that is at least $k - 1$ flips away from any other triangulation in $\mathcal{T}_k(S_k)$.*

Proof sketch. Set S_k is constructed as follows, see Fig. 2.a. Start with a horizontal segment uv and add points $S' = p_1, \ldots, p_k$ above it, and points $S'' = q_1, \ldots, q_k$ below it, such that q_i is the reflection of p_i with respect to the line through uv. Point p_1 is placed close enough to uv, and each next point p_{i+1} for $i = 1, \ldots, k - 1$ is: (1) inside $\bigcirc uq_ip_i$, (2) below the line through $p_{i-1}p_i$, (3) above the line through uv, and (4) outside $\bigcirc up_{i-1}p_i$ (we set $p_0 = v$). The set S_k is $\{u, v\} \cup S' \cup S''$. Triangulation T_k of S_k is formed by all the triangles $\triangle up_ip_{i+1}$ and $\triangle uq_iq_{i+1}$ (where $p_0 = q_0 = v$). It turns out that any $\bigcirc up_ip_{i+1}$ (resp., $\bigcirc uq_iq_{i+1}$) contains exactly the k points of S'' (resp., of S') in its interior and no other point of S_k. Thus T_k is in $\mathcal{T}_k(S_k)$. We observe that any triangulation of S_k containing edge p_ip_t with $k \geq i > t + 1$

is not of order k (the case for q_iq_t is symmetric). Consider $T' \neq T_k$ in $\mathcal{T}_k(S_k)$. Thus, each edge in $T' \setminus T_k$ must have one endpoint in S' and one in S''. Thus, edge uv has to be flipped in order to transform T_k to T'. Triangle $\triangle(uq_1p_1)$ is of order $2k-2$. The second part of the statement follows from the observation that for any i, j with $k \geq i > 0$ and $k \geq j > 0$, the triangle up_iq_j is of order $2k-i-j$. Thus, $k-1$ flips are needed to get T', since some $\triangle up_iq_j$ has to be in T' with $i+j \geq k$. Otherwise uv is in T', a contradiction. ◀

Let S be a point set in convex position. Let T be an order-k triangulation of S. We say that T is *minimal* if flipping any illegal edge in T results in a triangulation that is not of order k. Let uv be a diagonal in T and let $\triangle uxv$ and $\triangle uyv$ be the triangles adjacent to it. Since S is in convex position, the diagonal uv in T partitions the triangulation T into two sub-triangulations that only share edge uv. Let T^x_{uv} (respectively, T^y_{uv}) denote the sub-triangulation that contains triangle $\triangle uxv$ (respectively, $\triangle uyv$). See Fig. 2.b.

We show that any order-k triangulation different from DT can be transformed into some other order-k triangulation by performing at most $k-1$ flips of illegal edges such that all the intermediate triangulations are of order $2k-2$.

▶ **Theorem 3.2.** *Let S be a point set in convex position and let $k \geq 2$. Let $T \neq DT(S)$ be in $\mathcal{T}_k(S)$. Then, there exists a triangulation T' in $\mathcal{T}_k(S)$ at flip distance at most $k-1$ in $G(\mathcal{T}_{2k-2}(S))$ from T such that $\alpha(T') > \alpha(T)$.*

Proof sketch. For $k = 2$ the theorem follows trivially, since $G(\mathcal{T}_2(S))$ is connected. Thus, we assume $k \geq 3$. Note that if T is not minimal, then there exists an illegal edge e such that the resulting triangulation T' after flipping e is a triangulation of order k with the property that $\alpha(T') > \alpha(T)$. Thus, we assume that T is a minimal triangulation. We observe that there must exist an illegal edge ac adjacent to triangle $\triangle abc$ such that T^b_{ac} consists of only legal edges: Since T is not an order-0 triangulation, there is an illegal edge in T. Note that if a triangle has two edges in the convex hull of S, its third edge must be legal in T, otherwise T would not be minimal. Since triangulations of polygons have at least two such triangles (often called *ears*), then there exists an edge ac in $\triangle abc$ such that T^b_{ac} consists of legal edges.

Let uv be an illegal edge in $\triangle uxv$ such that T^x_{uv} consists of legal edges. Let $\triangle uyv$ be the other triangle in T adjacent to uv. Consider the triangulation $T_1 = (T \setminus \{uv\}) \cup \{xy\}$. Note that $\alpha(T_1) > \alpha(T)$. Since T is minimal, T_1 is not an order-k triangulation. Thus, the only triangles that cannot be of order k in T_1 are the new triangles $\triangle uxy$ and $\triangle xyv$. Without loss of generality assume that $\triangle uxy$ is not of order k. By Lemma 2.1, it follows that $\triangle uxy$ is the only triangle that is not of order k. In addition, $\bigcirc uxy$ contains at most $2k-2$ points in its interior. By Obs. 2.2 it follows that $\circlearrowleft^{ux}_{y}$ has at least 2 points and at most $k-1$.

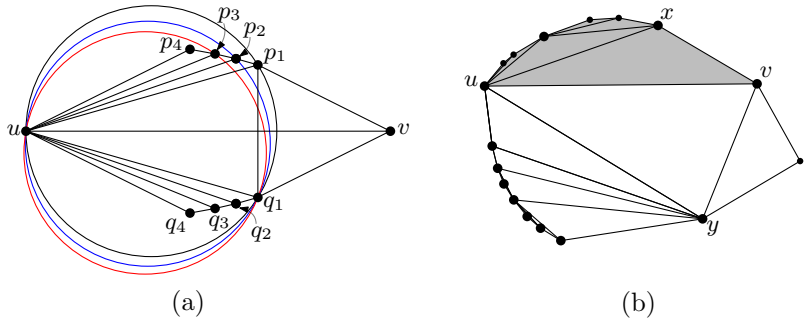


Figure 2 (a) An order-k triangulation at distance at least $k-1$ from other order-k triangulations (k=4). (b) The gray area corresponds to T^x_{uv}.

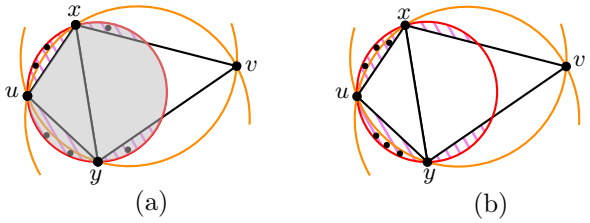


Figure 3 In both cases, $k = 5$. (a) There are four points in the gray region $\bigcirc uxy \setminus \oslash_y^{ux}$. (b) There are exactly three points in $\oslash_y^{ux}$ and exactly three points in $\oslash_x^{uy}$.

By induction on the number of points in $\oslash_y^{ux}$ we show that T_1 can be transformed into an order-k triangulation T' by flipping at most $k-2$ illegal edges. Hence, $\alpha(T') > \alpha(T_1) > \alpha(T)$. Moreover, the triangulations from T_1 to T' are of order $2k-2$, implying our result. ◀

If T is an order-k triangulation, then the edges of T have order k. There are $O(kn)$ edges of order k [2, 9]. It follows from Theorem 3.2 that T can be transformed into $DT(S)$ by a sequence of at most $O((2k-2)n) = O(kn)$ flips, since all the flipped edges are illegal and of order $2k-2$, which implies that no order-$(2k-2)$ edge is flipped twice.

4 General point sets

Consider a general point set S. Using a much more involved approach than the one for convex point sets, we can obtain an analogous result for triangulations of order $k = 3, 4$ or 5.

In order to prove this result, we consider a triangulation $T \in \mathcal{T}_k(S)$. If there is an illegal edge in T whose flip results in a new order-k triangulation, we are done. If not, $k > 2$ and T is a minimal triangulation. Thus, for any illegal edge uv in T, flipping uv produces a new and unique triangle $\triangle uxy$ that is not of order k. Since $k = 3, 4, 5$, we notice that there are only two cases to consider for the number of points in each region of $\bigcirc uxy$. For $k = 3, 4$, since $\triangle uxy$ is not of order k, by Obs. 2.2 it follows that one of the regions $\bigcirc uxy \setminus \oslash_y^{ux}$ and $\bigcirc uxy \setminus \oslash_x^{uy}$ contains $k-1$ points in its interior. For $k = 5$, if none of the regions $\bigcirc uxy \setminus \oslash_y^{ux}$ and $\bigcirc uxy \setminus \oslash_x^{uy}$ contains $k-1$ points of S in its interior then, by Obs. 2.2 and the fact that T is of order k, it follows that each region of $\oslash_y^{ux}$ and $\oslash_x^{uy}$ contains 3 points of S. See Fig. 3. Finally, for each of these two cases we show that the statement holds using the fact that when flipping certain illegal edges, the circumcircles of the new triangles lie in the union $\bigcirc uxv \cup \bigcirc uyv$.

In addition, since there are $O(kn)$ order-k edges (see [2, 9]), it follows that for $k \leq 5$, any order-k triangulation can be transformed into $DT(S)$ by a sequence of at most $O(kn)$ triangulations of order $2k-2$. Moreover, using the reverse search framework of Avis and Fukuda [4] and the pre-processing method of order-k triangulations given by Silveira and van Kreveld [16], all order-k triangulations can be enumerated in polynomial expected time per triangulation.

References

1 Yusuke Abe and Yoshio Okamoto. On algorithmic enumeration of higher-order Delaunay triangulations. In *Proceedings of the 11th Japan-Korea Joint Workshop on Algorithms and Computation, Seoul, Korea*, pages 19–20, 2008.

2 Manuel Abellanas, Prosenjit Bose, Jesús García, Ferran Hurtado, Carlos M Nicolás, and Pedro Ramos. On structural and graph theoretic properties of higher order Delaunay graphs. *Internat. J. Comput. Geom. Appl.*, 19(06):595–615, 2009.

3 Franz Aurenhammer, Rolf Klein, and Der-Tsai Lee. *Voronoi diagrams and Delaunay triangulations*. World Scientific Publishing Company, 2013.

4 David Avis and Komei Fukuda. Reverse search for enumeration. *Discrete Appl. Math.*, 65(1-3):21–46, 1996.

5 Prosenjit Bose and Ferran Hurtado. Flips in planar graphs. *Comput. Geom.*, 42(1):60–80, 2009.

6 Leila De Floriani. Surface representations based on triangular grids. *The Visual Computer*, 3(1):27–50, 1987.

7 Adrian Dumitrescu, André Schulz, Adam Sheffer, and Csaba D. Tóth. Bounds on the maximum multiplicity of some common geometric graphs. *SIAM J. Discrete Math.*, 27(2):802–826, 2013.

8 Steven Fortune. Voronoi diagrams and Delaunay triangulations. pages 225–265. World Scientific, 1995.

9 Joachim Gudmundsson, Mikael Hammar, and Marc van Kreveld. Higher order Delaunay triangulations. *Comput. Geom.*, 23(1):85–98, 2002.

10 Joachim Gudmundsson, Herman J Haverkort, and Marc Van Kreveld. Constrained higher order Delaunay triangulations. *Comput. Geom.*, 30(3):271–277, 2005.

11 Charles L. Lawson. Transforming triangulations. *Discrete Math.*, 3(4):365 – 372, 1972.

12 Charles L Lawson. Software for C1 surface interpolation. In *Mathematical software*, pages 161–194. Elsevier, 1977.

13 Anna Lubiw and Vinayak Pathak. Flip distance between two triangulations of a point set is NP-complete. *Comput. Geom.*, 49:17–23, 2015.

14 Alexander Pilz. Flip distance between triangulations of a planar point set is apx-hard. *Comput. Geom.*, 47(5):589–604, 2014.

15 Robin Sibson. Locally equiangular triangulations. *Comput. J.*, 21(3):243–245, 1978.

16 Rodrigo I Silveira and Marc van Kreveld. Optimal higher order Delaunay triangulations of polygons. *Comput. Geom.*, 42(8):803–813, 2009.

Distance Measures for Embedded Graphs - Optimal Graph Mappings

Maike Buchin[1] **and Bernhard Kilgus**[2]

1 **Department of Mathematics, Ruhr University Bochum, Bochum, Germany**
`Maike.Buchin@rub.de`
2 **Department of Mathematics, Ruhr University Bochum, Bochum, Germany**
`Bernhard.Kilgus@rub.de`

Abstract

We want to compare embedded graphs at a global and a local scale. The graph distances presented in [5] compare two graphs by mapping one graph onto the other and taking the maximum Fréchet distance between edges and their mappings. For this, the graph mapping is chosen such that the bottleneck distance is minimized. Here, we present two approaches to compute graph mappings subject to additional optimization criteria in order to improve distances locally.

1 Introduction

Motivation We are interested in comparing two embedded graphs. There are many applications that work with graphs embedded in an Euclidean space, such as road networks. For instance, by comparing two road networks one can assess the quality of map construction algorithms [3, 4]. Recently, graph distances based on graph mappings [5] were presented suitable for this task. However, these distances are bottleneck distances and a mapping realizing the bottleneck graph distance might be far from optimal for a single edge or any subgraph. See Figure 1 for an example. In this paper, we want to improve these mappings such that they express local distances more accurately.

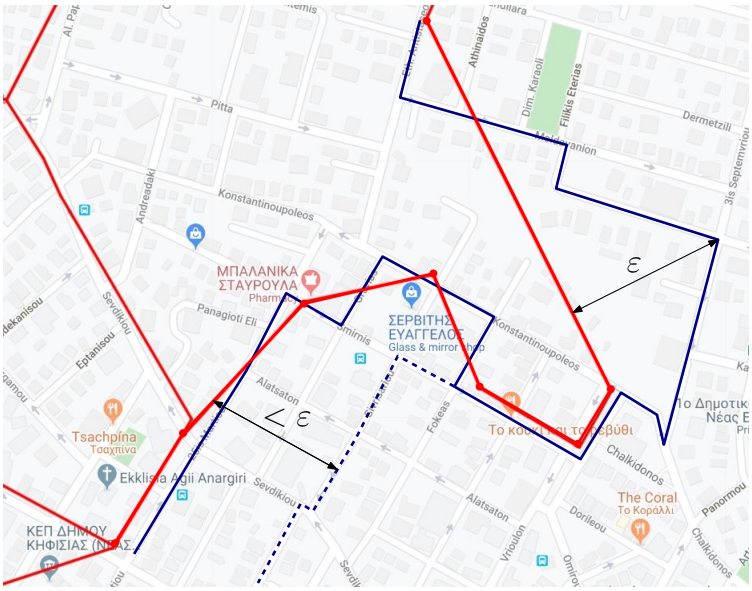


Figure 1 A partial map reconstruction R of the street map of Athens (in red). The graph distance between R and the ground truth is ε. The blue dashed and solid mappings are both valid, although the latter captures the local distance between the reconstruction and the ground truth better.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Related Work Several approaches have been proposed for comparing embedded graphs. These include an edit distance [8], algorithms that compare all paths [1] or random samples of shortest paths [9], traversal distance [6], and local persistent homology distance [2]. However,as argued in [5], (most of) these capture only the geometry or only the topology of the graphs. The distances presented in [5] capture both and are based on an explicit mapping between the graphs. Here, we extend these measures by looking for locally good matchings. To obtain locally good matchings between two polygonal curves, Buchin et al. introduced locally correct matchings [7] and Rote suggested lexicographic Fréchet matchings [10].

The traversal distance is a bottleneck distance but given the optimal traversals, local optimality is obtained by computing lexicographic Fréchet matchings. This approach can also be applied when computing the path-based graph distances. However, such a local traversal or path-based distance still suffers the shortcoming of the original measures, namely that it reduces the graphs to one or several paths. The local persistent homology distance allows for computing and visualization of local distances whereas expressing local distances with the edit distance is limited as some edges and vertices might be deleted during transformation.

Definition and Previous Results Here, we summarize the definition of the graph distances, the general algorithmic approach to compute the distances and the computational complexity for several settings (general graphs, planar embedded graphs, trees) as described in detail in [5]. Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two undirected graphs with vertices embedded as points in $\mathbb{R}^d$ (typically in the plane) that are connected by straight-line edges. We consider a mapping $s\colon G_1 \to G_2$ that maps each vertex $v \in V_1$ to a point $s(v)$ on G_2 (not necessarily a vertex) and that maps each edge $\{u, v\} \in E_1$ to a simple path in G_2 with endpoints $s(u)$ and $s(v)$. The directed graph distances $\vec{\delta}_{(w)G}$ are defined as

$$\vec{\delta}_{(w)G}(G_1, G_2) = \inf_{s\colon G_1 \to G_2} \max_{e \in E_1} \delta_{(w)F}(e, s(e))^1,$$

where $\delta_{(w)F}$ denotes the (weak) Fréchet distance, s ranges over all graph mappings from G_1 to G_2, and e and its image $s(e)$ are interpreted as curves in the plane. These distances are not symmetric and the difference between $\vec{\delta}_{(w)G}(G_1, G_2)$ and $\vec{\delta}_{(w)G}(G_2, G_1)$ can be arbitrarily large. The undirected graph distances $\delta_{(w)G}(G_1, G_2)$ are defined as the maximum of $\vec{\delta}_{(w)G}(G_1, G_2)$ and $\vec{\delta}_{(w)G}(G_2, G_1)$. Note that in this extended abstract, we only consider optimizing the directed distances. The undirected distances can be defined analogously as the maximum of the two directions.

An *ε-placement* of a vertex v is a maximally connected component of G_2 restricted to the ε-ball $B_\varepsilon(v)$ around v. A (weak) *ε-placement* of an edge $e = \{u, v\} \in E_1$ is a path P in G_2 with endpoints on ε-placements C_u of u and C_v of v such that $\delta_{(w)F}(e, P) \leq \varepsilon$. In that case, we say that C_u and C_v are *reachable* from each other. An ε-placement C_v of v is *(weakly) valid* if for every neighbor u of v, there exists an ε-placement C_u of u such that C_v and C_u are *reachable* from each other.

The general algorithmic approach to solve the decision problem of the directed (weak) graph distances for a given value $\varepsilon > 0$ consists of the following steps [5] **(1)** Compute all ε-placements of vertices and **(2)** of edges. Subsequently, **(3)** prune all invalid placements and **(4)** decide whether $\vec{\delta}_{(w)G}(G_1, G_2) \leq \varepsilon$ based on the remaining valid ε-placements. We call a graph mapping s that realizes $\vec{\delta}_{(w)G}(G_1, G_2) \leq \varepsilon$ a (weakly) valid mapping.

[1] We use the (w)-notation to simultaneously address the weak Fréchet distance and the Fréchet distance and the weak graph distance and the graph distance, respectively.

Deciding the directed (weak) graph distance is NP-hard for general graphs, but we can compute the (weakly) valid ε-placements in polynomial time. If there is a vertex with no (weakly) valid ε-placement, it follows that $\vec{\delta}_{(w)G}(G_1, G_2) > \varepsilon$. Conversely, the existence of a (weakly) valid ε-placement for each vertex ensures $\vec{\delta}_{(w)G}(G_1, G_2) \leq \varepsilon$ for several cases, namely if G_1 is a tree (both graph distances) and if G_1 and G_2 are plane graphs (weak graph distances). Thus, the distances are decidable in polynomial time in these cases. Deciding whether $\vec{\delta}_G(G_1, G_2) \leq \varepsilon$ remains NP-hard, if G_1 and G_2 are plane graphs [5].

Contribution The graph distances are bottleneck distances and a valid mapping might be far from optimal for a specific edge. To improve the local distance, we introduce additional optimization criteria for the graph mappings. First observe that in contrast to the (weak) Fréchet distance for polygonal paths [7, 10], we cannot expect to find a valid mapping $s_1 \colon G_1 \to G_2$ that is locally minimal in the sense that for any other valid mapping s_2, $\delta_{(w)F}(e, s_1(e)) \leq \delta_{(w)F}(e, s_2(e))$ for each edge e of G_1. See Figure 2 for an example.

We formulate the following optimization criteria: One natural goal is to minimize the (weighted) sum of the (weak) Fréchet distance between edges and their images; we denote this goal by *(weak) min-sum graph distance* and describe how to compute the (weak) min-sum graph distance for the setting where G_1 is a tree in Section 2.

Another goal is to refine the minmax optimization goal of the definition of $\vec{\delta}_{wG}(G_1, G_2)$. Intuitively, in addition to the largest value, we want to minimize the second largest value with respect to the largest value and so on. We denote this goal by *lexicographic graph distance*. Note that this approach only applies for the weak graph distance. We show how to compute the lexicographic graph distance for the setting where both graphs are planar in Section 3.

2 Min-Sum Graph Distance for Trees

Let G_1 be a tree. First, we compute the bottleneck distance $\varepsilon = \vec{\delta}_{(w)G}(G_1, G_2)$. Subsequently, we choose an optimal mapping based on ε. That is, we compute $\min_s \sum_{e \in E_1} \delta_{(w)F}(e, s(e))$, where $s \colon G_1 \to G_2$ ranges over all valid graph mappings with respect to ε.

Note that it might be possible to decrease this value by choosing an initial value $\varepsilon' > \vec{\delta}_{(w)G}(G_1, G_2)$. Mappings with small distance to most of the edges of G_1 are possibly declared invalid with respect to ε if the bottleneck distance for these mappings is large. Therefore, an alternative initial value can be used if we allow an (arbitrarily) large bottleneck distance for the min-sum graph distance. However, computing the min-sum graph distance with respect to ε is an optimal mapping respecting the global directed graph distance.

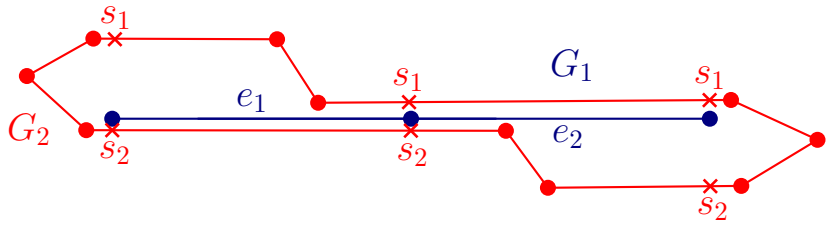


Figure 2 The graph mapping s_1 is locally optimal for e_2 but not for e_1, whereas the graph mapping s_2 is locally optimal for e_1 but not for e_2.

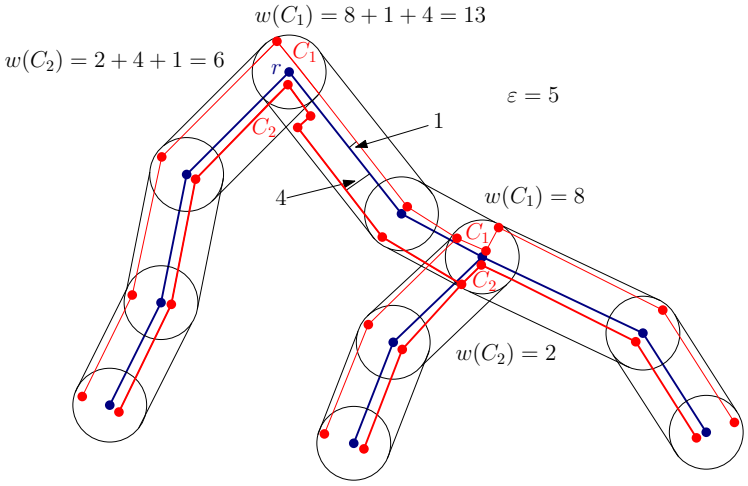


Figure 3 The weights of the vertex placements of the root r consist of the sum of the weights of the vertex placements of the children of r in $\overline{G}_1$ and the weights of the paths between the placements. The min-sum graph mapping of G_1 (in blue) onto G_2 (in red) is marked with bold lines.

Description of the Search Structure For each edge $e = (u, v) \in G_1$ and each pair of ε-placements C_u, C_v, we compute the minimum (weak) Fréchet distance $\Delta(C_u, C_v)$ of e and an edge-placement of e connecting C_u and C_v. We store a list L_e with entries $(C_u, C_v, \Delta(C_u, C_v))$ for each combination of vertex placements. That is, we augment step 2 of the algorithm by explicitly computing and storing the (weak) Fréchet distance of an edge and its placements. This can be done in $O(n_1 m_2^2 \log(m_2))$ time and $O(n_1 m_2^2)$ space for both the weak Fréchet distance and the Fréchet distance. We denote the set of placements of a vertex u by $P(u)$.

We will compute the min-sum graph distance *bottom-up*, maintaining the invariant that subgraphs have been optimally placed for any vertex-placement. For this, we consider G_1 as directed tree with arbitrary root r. The (abstract) *reachability graph* H of the vertex placements has one vertex for each vertex placement of a vertex of G_1. Two vertices C_u, C_v are adjacent in H if the corresponding vertices u, v of G_1 are adjacent and if the two placements are reachable from each other. The edges of H are weighted with the minimum (weak) Fréchet distance between the corresponding edge of G_1 and an edge placement connecting C_u and C_v in G_2. Note that alternatively to the Fréchet bottleneck distances one might, for instance, use the minimum area between an edge and a valid path as weights. As each vertex $u \in V_1$ has $O(m_2)$ vertex placements the vertex set V_H of H has size $O(n_1 m_2)$ and the set of edges E_H has size $O(n_1 m_2^2)$. We consider the edges of H to be directed according to the direction of the edges of G_1. Obviously, H is a directed acyclic graph (DAG).

For simplicity, let $\overline{G}_1$ be the graph obtained by replacing all paths of internal degree-2 vertices in G_1 by one edge between the start and endpoint of the path. Then, for all vertices v of $\overline{G}_1$, $deg(v) \geq 3$. See Figure 3 for a small example of a min-sum graph mapping.

Description of the Algorithm First, we set the weight $w(p) = 0$ for all placements p of vertices of $\overline{G}_1$. Let $u \in \overline{V}_1$ be a vertex whose children are all leaves. We compute

$$w(C_u) = \sum_{u' : u' \text{is child of } u} \min_{C_{u'} \in P(u')} w(C_{u'}) + w_H(C_u, C_{u'}), \tag{1}$$

where $w_H(C_u, C_{u'})$ is the weight of a minimum weight shortest path P between C_u and $C_{u'}$ in H. We store the mapping s which realizes $w(C_u)$, delete the subtree of $\overline{G}_1$ rooted at u,

and proceed with the next vertex of the updated graph $\overline{G}_1$ with leaf-children only until we encounter the root r. After having processed r, the following theorem holds:

▶ **Theorem 1.** *We can compute a valid mapping s in $O(n_1 m_2^3)$ time and $O(n_1 m_2^2)$ space such that for any other valid mapping $s' : G_1 \to G_2$ we have*

$$\sum_{e \in E_1} \delta_{(w)F}(e, s'(e)) \geq \sum_{e \in E_1} \delta_{(w)F}(e, s(e)).$$

Proof. The graph H can be computed in $O(n_1 m_2^2 \log(m_2))$ time and uses $O(n_1 m_2^2)$ space. The algorithm maintains the invariant of optimally mapped subtrees and thus terminates with a min-sum mapping. The complexity of the subgraph of H for computing $w_H(C_u, C_{u'})$ for all children u' of u in $\overline{G}_1$ is $O(l m_2^2)$, where l is the length of the path between u and u' in G_1. Since H is a DAG, we can compute equation (1) in $(deg(u) - 1)O(l m_2^2)$ time using topological sorting. As u has up to m_2 placements, the runtime for processing one vertex of $\overline{G}_1$ is $O(deg(u) l m_2^3)$. With $\sum_{u \in \overline{V}_1} deg(u) l = 2m_1 = 2(n_1 - 1)$, the runtime follows. ◀

▶ Remark. While computing the directed graph distance for planar embedded graphs is NP-hard, one can compute the directed weak graph distance in polynomial time [5]. It remains open whether the weak min-sum graph distance can be computed in polynomial time for planar embedded graphs, but we conjecture that the problem is NP-hard.

3 Lexicographic Graph Distance

First, we formally define the lexicographic graph distance based on weak Fréchet distance.

▶ **Definition 2.** Let $s \colon G_1 \to G_2$ be a mapping. We say that s is a *mapping realizing the lexicographic graph distance*, if it has the following property: Given an arbitrary subdivision D of the graph G_1 (with finite number l of edges). Let $e_1, e_2, \ldots, e_l$ be a numbering of the edges of D such that $\varepsilon_1 \geq \varepsilon_2 \geq \cdots \geq \varepsilon_l$, where $\varepsilon_i = \delta_{wF}(e_i, s(e_i))$. If there exists another mapping $\hat{s}$, such that $\delta_{wF}(e_i, \hat{s}(e_i)) < \delta_{wF}(e_i, s(e_i))$ for some $i \in \{1, 2, \ldots, l\}$, then there exists an index $j < i$ such that $\delta_{wF}(e_j, \hat{s}(e_j)) > \delta_{wF}(e_j, s(e_j))$.

A mapping that realizes the lexicographic graph distance is a *lexicographic graph mapping*.

Note that here we use the term lexicographic with a slight abuse of the standard notation of a lexicographic order. This is due to the fact that the entities that must be ordered differ for each mapping from G_1 to G_2. Intuitively, any mapping with a locally smaller distance in comparison with a lexicographic graph mapping increases the value of some larger weak Fréchet distance of an edge and its image.

In the following, we first assume that no pair of edges (e_1, e_2), where $e_1 \in E_1$ and e_2 in E_2, is parallel. Hence we can assume that the weak Fréchet distance between an edge $e \in E_1$ and a path $s(e)$ in G_2 is characterized by the maximum M of the maximum distance between a vertex $v \in V_2$ on $s(e)$ and the edge e and the distances of the endpoints of e and $s(e)$. Second, we assume that the length of the perpendiculars of edges of G_2 and vertices of G_1 are unique. Furthermore, we assume that M is uniquely defined by a vertex on $s(e)$ or by one of the endpoints of $s(e)$. Last, we assume that for any valid mapping s, the weak Fréchet distance between an edge e and $s(e)$ are unique. These assumptions imply:

▶ **Lemma 3.** *Let $\varepsilon = \vec{\delta}_{wG}(G_1, G_2)$. If $\delta_{wF}(e, s(e)) = \varepsilon$ for an edge $e = (u, v) \in E_1$ and a valid mapping $s \colon G_1 \to G_2$. Then, exactly one of the following cases occurs:*

- ***Case 1:** There is a vertex $w \in V_2$ on $s(e)$ with $dist(e, w) = \varepsilon$. Then, for each valid mapping $\hat{s} \neq s$, $\hat{s}(e)$ contains w and therefore $\delta_{wF}(e, \hat{s}(e)) = \varepsilon$.*

- ***Case 2:*** *$dist(u, s(u)) = \varepsilon$. In this case, u has exactly one valid placement.*
- ***Case 3:*** *$dist(u, s(v)) = \varepsilon$. In this case, v has exactly one valid placement.*

Description of the Algorithm First we compute the directed weak graph distance $\varepsilon = \vec{\delta}_{wG}(G_1, G_2)$ and store a copy of G_2 restricted to the ε-surrounding of e for each edge $e \in E_1$. We denote this subgraph by $G_2(e)$. In each step of the algorithm we compute $\varepsilon = \vec{\delta}_{wG}(G_1, G_2)$, where the subgraphs $G_2(e)$ are used for all graph explorations in step **(1)** and **(2)**. We consider the unique edge e with $\delta_F(e, s(e)) = \varepsilon$ and the unique point w on $s(e)$ with distance ε to e and remove this bottleneck by updating the graph as follows: Snap w to w' on e at distance ε and update all incident edges of w in $G_2(w)$ accordingly. Proceed until $\vec{\delta}_{wG}(G_1, G_2) = 0$. Figure 4 illustrates the iterations of the algorithm.

Note that the updated graphs are not necessarily plane, but planarity is not needed for computing valid placements and pruning invalid placements. To compute a mapping for two adjacent cycles, the corresponding placements in the plane graph G_2 are used.

▶ **Theorem 4.** *Given plane graphs G_1, G_2, the algorithm described above computes a lexicographic graph mapping $s\colon G_1 \to G_2$ in $O(n_1^2 n_2^2 \log(n_1 + n_2))$ time using $O(n_1 n_2)$ space.*

Proof. Let $G_2(e = (u, v))$ be the graph updated in iteration i and let $G_2(e)_{prev}$ be the graph before the update. Furthermore, let $\varepsilon = \vec{\delta}_{wG}(G_1, G_2(e))$ be the weak directed graph distance in iteration $i + 1$ and $\varepsilon_{prev} = \vec{\delta}_{wG}(G_1, G_2(e)_{prev})$. Now Lemma 3 implies that for a valid mapping $s\colon G_1 \to G_2(e)$ with respect to ε, the corresponding re-transformed mapping is a valid mapping from G_1 onto $G_2(e)_{prev}$ with respect to ε_{prev}. That is, when the algorithm terminates, we can easily compute a valid mapping from G_1 onto G_2 by a series of re-transformations of the graph G_2. The obtained mapping is a lexicographic graph mapping as in each step the algorithm identifies the current bottleneck distance. In each iteration, either a vertex of G_2 is snapped onto an edge e of G_1, or a point of an edge of G_2 is snapped onto an endpoint of e. The latter case can only happen once for each endpoint of e. Therefore, at most $n_2 + 2$ points of G_2 are snapped onto e until the distance between e and $s(e)$ is zero for any valid mapping s. Thus, after a maximum of $m_1(n_2 + 2) = O(n_1 n_2)$ iterations, the algorithm terminates. Hence, the total runtime is $O(n_1^2 n_2^2 \log(n_1 + n_2))$. ◀

▶ Remark. The definition and algorithmic approach cannot be directly transferred to the Fréchet distance instead of the weak Fréchet distance The Fréchet distance depends on the specific subdivision of an edge. In general, the distance increases for larger subdivisions and is maximal between the whole edge and the corresponding path.

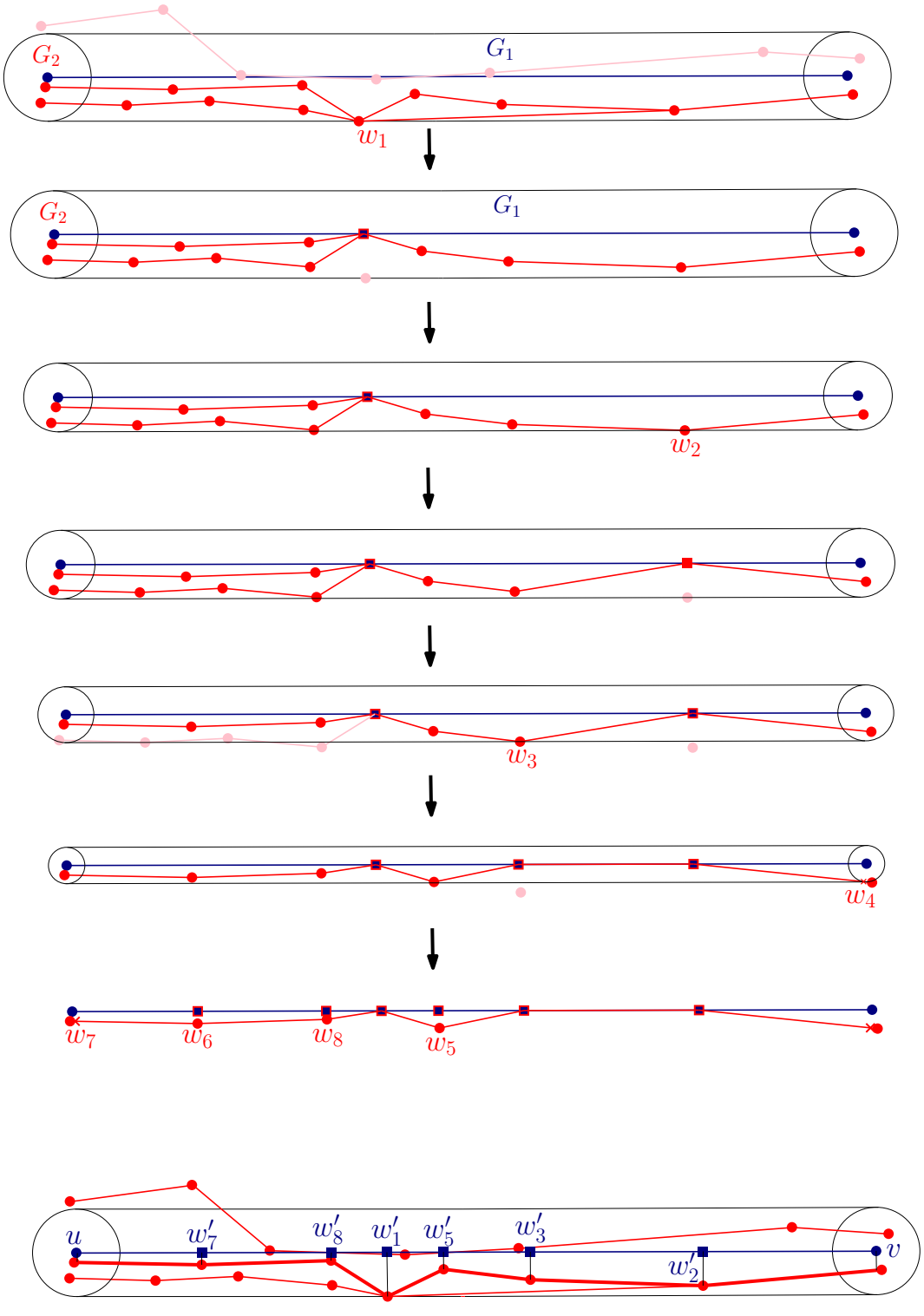


Figure 4 Iteratively updating G_2. The lexicographic graph mapping (bold line) is uniquely defined by the vertices w_i and u, v, w'_i: $s_{opt}(u) = w_7$, $s_{opt}(v) = w_4$, $s_{opt}(w'_7) = w_6$, $s_{opt}(w'_8) = w_8$, $s_{opt}(w'_1) = w_1$, $s_{opt}(w'_5) = w_5$, $s_{opt}(w'_3) = w_3$, $s_{opt}(w'_2) = w_2$, $s_{opt}(v) = w_4$.

References

1 Mahmuda Ahmed, Brittany Terese Fasy, Kyle S. Hickmann, and Carola Wenk. Path-based distance for street map comparison. *ACM Transactions on Spatial Algorithms and Systems*, 28 pages, 2015.

2 Mahmuda Ahmed, Brittany Terese Fasy, and Carola Wenk. Local persistent homology based distance between maps. In *22nd ACM SIGSPATIAL GIS*, pages 43–52, 2014.

3 Mahmuda Ahmed, Sophia Karagiorgou, Dieter Pfoser, and Carola Wenk. A comparison and evaluation of map construction algorithms using vehicle tracking data. *GeoInformatica*, 19(3):601–632, 2015.

4 Mahmuda Ahmed, Sophia Karagiorgou, Dieter Pfoser, and Carola Wenk. *Map Construction Algorithms*. Springer, 2015.

5 Hugo A. Akitaya, Maike Buchin, Bernhard Kilgus, Stef Sijben, and Carola Wenk. Distance Measures for Embedded Graphs. In Pinyan Lu and Guochuan Zhang, editors, *30th International Symposium on Algorithms and Computation (ISAAC 2019)*, volume 149 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 55:1–55:15, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: `http://drops.dagstuhl.de/opus/volltexte/2019/11551`, `doi:10.4230/LIPIcs.ISAAC.2019.55`.

6 Helmut Alt, Alon Efrat, Günter Rote, and Carola Wenk. Matching planar maps. *Journal of Algorithms*, 49(2):262 – 283, 2003.

7 Kevin Buchin, Maike Buchin, Wouter Meulemans, and Bettina Speckmann. Locally correct Fréchet matchings. In *Proceedings of the 20th Annual European Conference on Algorithms*, ESA'12, pages 229–240, Berlin, Heidelberg, 2012. Springer-Verlag. URL: `http://dx.doi.org/10.1007/978-3-642-33090-2_21`, `doi:10.1007/978-3-642-33090-2_21`.

8 Otfried Cheong, Joachim Gudmundsson, Hyo-Sil Kim, Daria Schymura, and Fabian Stehn. Measuring the similarity of geometric graphs. In *International Symposium on Experimental Algorithms*, pages 101–112, 2009.

9 Sophia Karagiorgou and Dieter Pfoser. On vehicle tracking data-based road network generation. In *20th ACM SIGSPATIAL GIS*, pages 89–98, 2012.

10 Günter Rote. Lexicographic Fréchet matchings. In *Proc. 30rd European Workshop on Computational Geometry (EuroCG)*, 2014.

Reconfiguring sliding squares in-place by flooding*

Joel Moreno[1] and Vera Sacristán[1]

1 **Universitat Politècnica de Catalunya**
joel.moreno97@gmail.com, vera.sacristan@upc.edu

Abstract

We present a new algorithm that reconfigures between any two edge-connected configurations of n sliding squares within their bounding boxes. The algorithm achieves the reconfiguration by means of $\Theta(n^2)$ slide moves. A visual simulator and a set of experiments allows us to compare the performance over different shapes, showing that in many practical cases the number of slide moves grows significantly slower than in others as n increases.

1 Introduction

As defined in [13], "Modular self-reconfigurable (MSR) robots are robots composed of a large number of repeated modules that can rearrange their connectedness to form a large variety of structures. An MSR system can change its shape to suit the task, whether it is climbing through a hole, rolling like a hoop, or assembling a complex structure with many arms."

Their versatility, though, comes with a drawback: the complexity of the algorithms required to control MSR systems and make them walk, self-repair or, more generally, reconfigure. Reconfiguring modular robots without a thoughtful method can generate an excessive number of moves (and therefore too much time and power consumption), disconnections of the robot, collisions between its modules, dead-locks and other complex situations.

In this paper we propose a new and efficient in-place universal reconfiguration algorithm for a class of lattice-based modular robots.

1.1 Basic Definitions

A *connected robot configuration* (in short, a *robot* or a *configuration*) is any edge-connected set of squares (*modules* of the robot) located in a 2-dimensional square lattice. In other words, it is a polyomino. Within the *sliding-square* setting, robot modules (represented as squares) move relative to each other by sliding along their shared edges, as shown in Figure 1.

Figure 1 The sliding move allows straight transitions (left) as well as convex transitions (right).

A *hole* in a configuration is any vertex-connected component of empty lattice cells. A *pseudo-hole* is any edge-connected component of empty lattice cells. See Figure 2 for an illustration. Notice that, with this definitions, the unbounded component is considered to be a hole/pseudo-hole. Notice also that pseudo-holes are subsets of holes.

* V.S. was partially supported by MTM2015-63791-R (MINECO/FEDER) and Gen. Cat. DGR 2017SGR1640.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

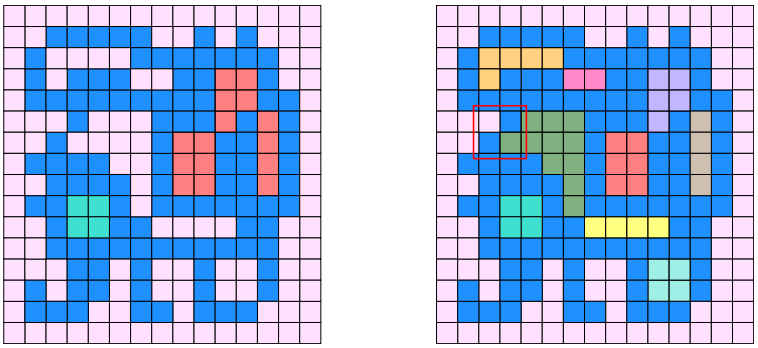


Figure 2 Example of a configuration (in blue) with 3 holes and 10 pseudo-holes. Left: Empty cells marked with the same color belong to the same hole. Right: Empty cells marked with the same color belong to the same pseudo-hole. The red square indicates one of the many critical pairs.

We call the *boundary* of a hole (respectively, pseudo-hole) the cycle of vertices and edges simultaneously incident to the hole (pseudo-hole) and the robot, and *boundary traversal* the cycle of hole (pseudo-hole) cells incident to its boundary.

A *critical pair* is a pair of vertex-adjacent robot modules such that no module exists that is edge adjacent to both. They can be found in pseudo-holes that are not holes. Figure 2 shows an example.

1.2 Problem Statement and Results

Given any two configurations C and C' with n modules each, we present an efficient algorithm that reconfigures C into C' in-place, without disconnecting the robot at any moment throughout the reconfiguration. The algorithm is centralized and sequential, i.e., it slides one module at a time. It is efficient in the sense that the overall slide moves performed along the reconfiguration is $\Theta(n^2)$, which is optimal if constant force and velocity are assumed [2]. Furthermore, if B and B' respectively are the 1-cell offset of the bounding boxes of C and C', and B and B' share their left-bottom corner cell, then space used throughout the reconfiguration is enclosed in $B \cup B'$.

Furthermore, we provide an on-line simulator of our algorithm, together with a first analysis of its performance in practice on a variety of configurations of different sizes.

1.3 Related Work

The sliding-square/cube model was introduced in [4] as a geometric abstraction for a variety of modular robots, such as Metamorphic [5], Vertical [8], Molecube [14], M-TRAN [10], EM-cube [3], or Smart [11]. Since then, several algorithms have been proposed to solve different (but somehow related) problems such as locomotion [4, 7], self-repair [9], and reconfiguration [6, 1].

In particular, [6] proposes the first algorithm to reconfigure between any two edge-connected shapes of sliding squares in the plane with the same number of modules. The algorithm is sequential and reconfiguration is done through an intermediate shape: a strip grown from one extremal module. As a consequence, the overall reconfiguration takes place in the disjoint union of the 1-offset of the bounding boxes of the initial and final configurations.

The algorithm we present builds on the ideas from [6], but differs from it in that it reconfigures in-place, i.e., within the 1-offset of the union of the two bounding boxes. This

adds further difficulties, as in our case the final destination of a module may be located in a hole or a pseudo-hole, and not only in the outer boundary as in [6]. We further elaborate on this issue in the following section.

2 Algorithm overview

Given two configurations C and C', our strategy to reconfigure C into C' consists of three steps, illustrated in Figure 3.

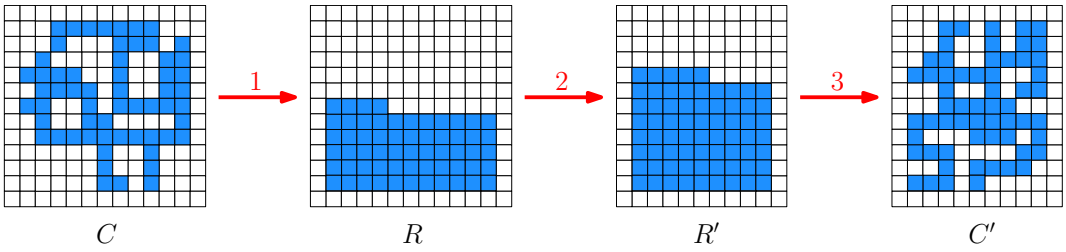


Figure 3 Reconfiguration steps.

1. First, we *flood* the bounding box of our initial configuration C. This process consists of sequentially finding modules that can slide without disconnecting the configuration, and sending them along a boundary traversal to fill the bounding box of C by rows from bottom to top, filling each row from left to right. The result is a rectangle, except for its topmost row, that may be incomplete. We call this shape R.
2. Then, we reconfigure R into R', which is the analogous almost-rectangular shape corresponding to C'.
3. Finally, we reconfigure R' into C'.

The algorithm for step 2 is straightforward, and that of step 3 consists of reversing the procedure of step 1. Therefore, in the remaining of this paper we concentrate on step 1.

As already mentioned, when flooding, the final destination of a module may be located in a hole or a pseudo-hole, and not only in the outer boundary as in [6]. In other words, while in [6] the algorithm consists in finding a feasible sequence of moves (of possibly several modules) that ends up freeing a module in the outer boundary, our algorithm needs to produce sequences that first go outwards from a pseudo-hole towards the outer boundary and then may have to go inwards to a different pseudo-hole in order to fill the goal position. Figure 4 illustrates this process.

We solve this issue using a hierarchy of pseudo-holes that is built by traversing all the boundaries of the configuration in a preprocessing step. Starting at the outer one, the boundary traversal detects all critical pairs. Each critical pair is a gate for a pseudo-hole that is a descendant in the hierarchy. Gates between pseudo-holes contained in holes different than the outer one are considered siblings in the hierarchy. Figure 5 illustrates this.

Once the hierarchy has been built, the algorithm looks for a module that is able to directly move to the goal cell, i.e., a module that is not a cut vertex of the edge-adjacency graph of the configuration, and is adjacent to the same pseudo-hole boundary as the next cell position to be filled. If such module does not exist, the algorithm searches through the pseudo-holes hierarchy for a module that can be moved, and sends it to connect a critical pair in the path to the goal pseudo-hole. This creates a cycle in the adjacency graph of C enclosing the goal cell, which, at its turn, allows finding a new module that can be moved inside the cycle. The process continues until a movable module can be found in the goal pseudo-hole. This, at last,

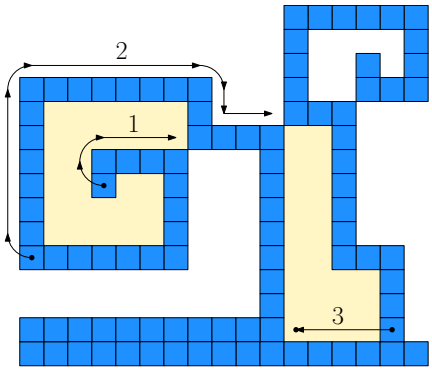


Figure 4 Filling the next free position in the second row. Notice how the sequence of moves starts in a pseudo-hole, continues in the outer boundary, and ends back in a (different) pseudo-hole.

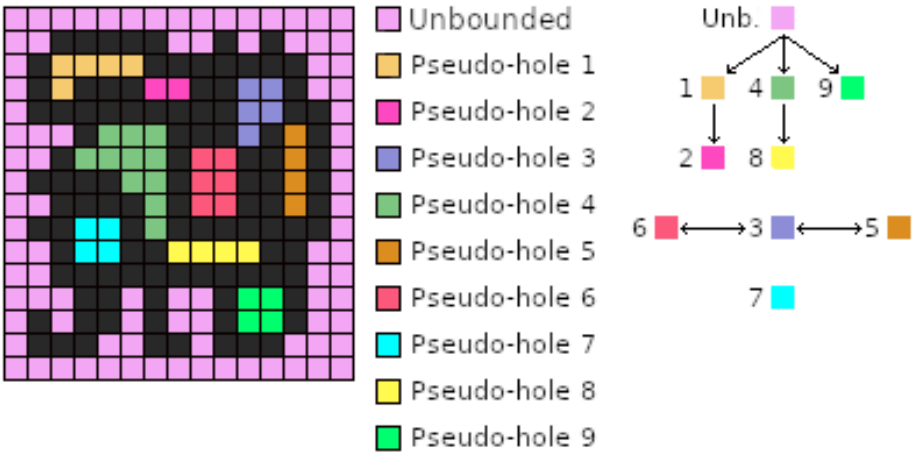


Figure 5 A Hierarchy of pseudo-holes.

fills the goal cell. Naturally, every time a modules leaves a position in the configuration, and every time it stops either because it reaches the goal cell or because it closes a critical pair, the hierarchy is updated accordingly.

This procedure is repeated until the configuration has been completely flooded.

3 Correctness and complexity

We denote by G the edge-adjacency graph of the configuration C, and by T the cactus graph associated WITH G, i.e., the tree of maximal cycles of G. We start by proving two lemmata.

▶ **Lemma 3.1.** *As long as not all the modules are in their final destination in R, there always exists a module that can slide without disconnecting the configuration.*

Proof. Since T is a tree, it must have a leaf. If such a leaf is a module, it must be movable since it is a leaf in G. Otherwise, if the leaf is a cycle, it must contain a module that is not a cut vertex of T. Such a module is movable since it cannot be a cut vertex in G. ◀

▶ **Lemma 3.2.** *If a movable module cannot reach the goal cell, g, it can connect a critical pair reducing the size of the cycle containing g.*

Proof. Let h_g be the pseudo-hole containing g, and let h_m be the pseudo-hole containing a movable module m whose existence is guaranteed by Lemma 3.1. A path must exist in the pseudo-hole hierarchy connecting h_g to h_m, since every pseudo-hole must belong to a hole, each hole is bounded by a cycle in T, and each cycle in T must contain a module that is not a cut vertex. ◀

▶ **Theorem 3.3.** *Let C be any configuration with n modules and R a configuration with the same number of modules filling the bounding box of C from bottom to top, left to right. Our algorithm reconfigures C into R using $\Theta(n^2)$ sliding moves.*

Proof. Each goal cell is filled after $O(n)$ slide moves. This is evident when the moving module reaches its destination without having to stop at a critical pair, since any boundary traversal has size $O(n)$. When it does stop, it can be proved that the overall number of slide moves performed by the sequence of moving modules before filling the empty goal position is $O(n)$. Details are omitted due to space restrictions. Therefore, the entire flooding takes $O(n^2)$ slide moves. It is worth noticing that the second step of the algorithm (reconfiguring between R and R') can be trivially done with $O(n^2)$ slide moves. Therefore, the entire reconfiguration uses $O(n^2)$ slide moves, which is optimal within this context [2]. ◀

4 Simulator and practical experiments

In addition to the theoretical results, we have also implemented an on-line simulator of our reconfiguration algorithm, see [12]. The simulator allows the user to define the initial and the final configurations in a 2-dimensional grid both by uploading a predefined file or by direct interaction. Then, it visualizes all the slide moves that the robot would be undertaking, step by step from the initial configuration C to the final one, C'. The reconfiguration can be stopped at any moment and backtracking is also allowed.

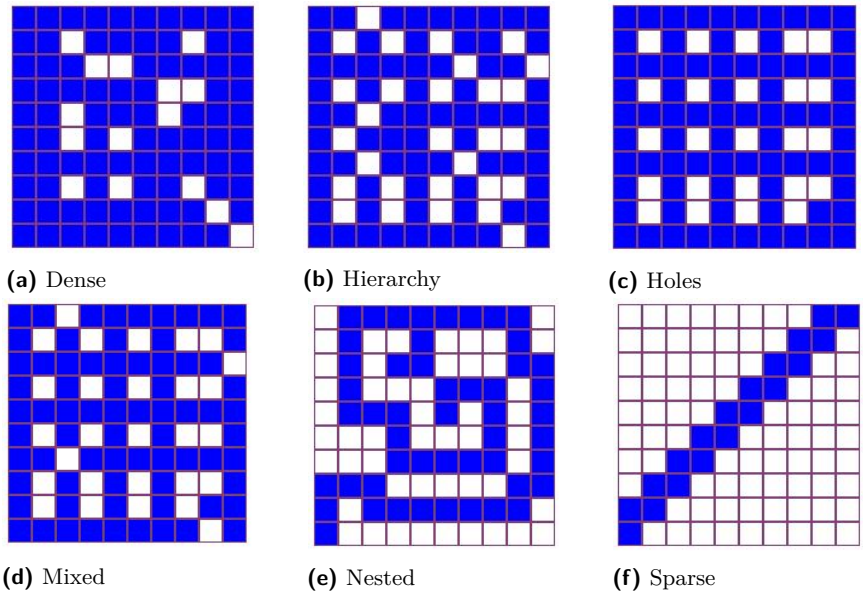
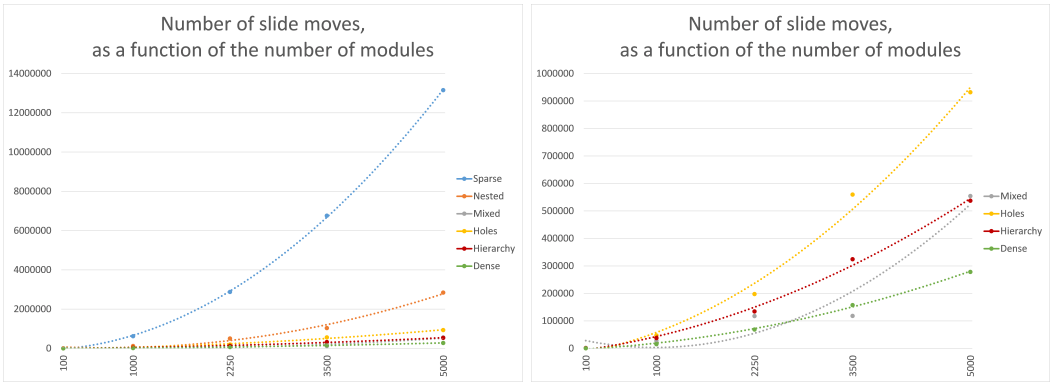


Figure 6 Examples of the configuration shapes used in our experiments.

We have used our simulator to test our algorithm on a variety of configurations, different in shape and size. The chosen shapes have been the following: Dense (configurations whose

bounding box is almost completely filled with modules), Hierarchy (configurations with several and large hierarchies of pseudo-holes), Holes (configurations without hierarchies but with multiple holes), Mixed (configurations combining large hierarchies of pseudo-holes with multiple holes), Nested (configurations requiring to connect critical pairs in order to reconfigure), Sparse (configurations with maximal ratio between the size of their bounding box and their number of modules). Figure 6 shows examples of these shape types.

The results of the experiments show that the sparser a configuration is, more slide moves its reconfiguration requires. This is a consequence of the fact that sparsity usually obliges the modules to slide over a great number of other modules in order to reach the intermediate rectangular shape. Denser configurations require a smaller number of moves because many modules are already located in their final grid positions. Figure 7 shows the results of the experiments we run.



Number of slide moves,
as a function of the number of modules

		Number of modules				
		100	**1000**	**2250**	**3500**	**5000**
Shape	**Dense**	422	15554	68765	157662	278332
	Hierarchy	1415	37582	134395	324502	537519
	Holes	433	46908	197741	559821	931817
	Mixed	808	27491	117543	118319	554274
	Nested	3061	122339	505382	1034488	2843997
	Sparse	7806	621082	2877678	6760828	13156460

Figure 7 Number of slide moves used to reconfigure the different configuration types into their corresponding rectangles.

Notice that the number of slide moves indicated in the table and charts corresponds to reconfiguring each initial configuration C into its corresponding rectangle R. The number of slide moves corresponding to the overall reconfiguration of an initial shape C into a final shape C' can be obtained by adding up the corresponding values from the table together with the number of steps required to reconfigure R into R', which is straightforward to compute.

5 Conclusions and open problems

We have presented an algorithm that reconfigures between any two edge-connected configurations of n sliding squares within their bounding boxes by means of $\Theta(n^2)$ slide moves, and we have made available an on-line simulator. We believe that extending our strategy to the hexagonal and triangular grids should be straightforward.

Our algorithm is intrinsically sequential. An interesting open problem is to obtain a new

strategy that allows to move several modules in parallel, giving rise to a faster reconfiguration, and to distribute it in order to obtain a more scalable algorithm. Extending the results to the 3-dimensional setting is also desirable.

6 Acknowledgments

We thank Diane Souvaine and Matias Korman for our preliminary discussions on a similar problem.

References

1 Z. Abel and S.D. Kominers. Universal reconfiguration of (hyper-)cubic robots. *CoRR*, abs/0802.3414, 2008. URL: `http://arxiv.org/abs/0802.3414`, `arXiv:0802.3414`.

2 G. Aloupis, S. Collette, M. Damian, E. D. Demaine, R. Flatland, S. Langerman, J. O'Rourke, V. Pinciu, S. Ramaswami, V. Sacristán, and S. Wuhrer. Efficient constant-velocity reconfiguration of crystalline robots. *Robotica*, 29(1):59–71, 2011.

3 B.K. An. EM-cube: cube-shaped, self-reconfigurable robots sliding on structure surfaces. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 3149–3155, 2008.

4 Z. Butler, K. Kotay, D. Rus, and K. Tomita. Generic decentralized control for a class of self-reconfigurable robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, page 809–816, 2002.

5 G.S. Chirikjian. Kinematics of a metamorphic robotic system. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 449–455, 1994.

6 A. Dumitrescu and J. Pach. Pushing squares around. *Graphs and Combinatorics*, 22:37–50, 2006.

7 R. Fitch and Z. Butler. Million module march: Scalable locomotion for large self-reconfiguring robots. *The International Journal of Robotics Research*, 27(3–4):331–343, 2008.

8 K. Hosokawa, T. Tsujimori, T. Fujii, H. Kaetsu, H. Asama, Y. Kuroda, and I. Endo. Self-organizing collective robots with morphogenesis in a vertical plane. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, volume 4, pages 2858–2863, 1998.

9 K. Kotay and D. Rus. Generic distributed assembly and repair algorithms for self-reconfiguring robots. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 2362–2369, 2004.

10 H. Kurokawa, K. Tomita, A. Kamimura, S. Kokaji, T. Hasuo, and S. Murata. Distributed self-reconfiguration of M-TRAN III modular robotic system. *International Journal of Robotics Research*, 27(3-4):373–386, 2008.

11 S. Mobes, G.J. Laurent, C. Clevy, N. Le Fort-Piat, B. Piranda, and J. Bourgeois. Toward a 2D modular and self-reconfigurable robot for conveying microparts. In *Proc. Second Workshop on Design, Control and Software Implementation for Distributed MEMS (dMEMS)*, pages 7–13, 2012.

12 J. Moreno. Reconfiguring sliding squares almost in-place. https://dccg.upc.edu/people/vera/teaching/tfm-tfg/flooding/.

13 M. Yim, P. White, M. Park, and J. Sastra. Modular self-reconfigurable robots. In R.A. Meyers, editor, *Encyclopedia of Complexity and Systems Science*, pages 5618–5631. Springer New York, 2009.

14 V. Zykov, A. Chan, and H. Lipson. Molecubes: An open-source modular robotic kit. In *IROS-2007 Self-Reconfigurable Robotics Workshop*, 2007.

Computational Complexity of the α-Ham-Sandwich Problem*

Man-Kwun Chiu[1], Aruni Choudhary[1], and Wolfgang Mulzer[1]

1 **Institut für Informatik, Freie Universität Berlin, Berlin, Germany**
[chiumk,arunich,mulzer]@inf.fu-berlin.de

Abstract

A variant of the Ham-Sandwich Theorem by Bárány, Hubard, and Jerónimo [DCG 2008] states that given any d measurable sets in $\mathbb{R}^d$ that are convex and *well-separated*, and any given $\alpha_1, \ldots, \alpha_d \in [0, 1]$, there is a unique oriented hyperplane that cuts off a respective fraction $\alpha_1, \ldots, \alpha_d$ from each set. Steiger and Zhao [DCG 2010] proved a discrete analogue, which we call the *α-Ham-Sandwich theorem*. They gave an algorithm to find the hyperplane in time $O(n(\log n)^{d-3})$, where n is the total number of input points. The computational complexity of this search problem in high dimensions is open, unlike that of the Ham-Sandwich problem, which is now known to be PPA-complete (Filos-Ratsikas and Goldberg [STOC 2019]).

Recently, Fearley, Gordon, Mehta, and Savani [ICALP 2019] introduced a new sub-class of CLS (Continuous Local Search) called *Unique End-of-Potential Line* (UEOPL). This class captures problems in CLS that have unique solutions. We show that for the α-Ham-Sandwich theorem, the search problem of finding the dividing hyperplane lies in UEOPL. This gives the first non-trivial containment of the problem in a complexity class and places it in the company of several classic search problems.

1 Introduction and preliminaries

The classic Ham-Sandwich theorem [7, 8, 12] states that for any d measurable sets in $\mathbb{R}^d$, there is a hyperplane that bisects them simultaneously. Bárány et al. [2] proved a variant of this classic theorem that aims at dividing sets into arbitrary given ratios instead of simply bisecting them. The sets $S_1, \ldots, S_d \subset \mathbb{R}^d$ are *well-separated* if every selection of the sets can be strictly separated from the others by a hyperplane. If the sets are well-separated and convex, then for any given choice $\alpha_1, \ldots, \alpha_d \in [0, 1]$, there is a unique oriented hyperplane that divides $S_1, \ldots, S_d$ in the ratios $\alpha_1, \ldots, \alpha_d$, respectively.

Steiger and Zhao [11] gave a discrete version of [2] and called their result the *Generalized Ham-Sandwich Theorem*, yet it is not a strict generalization of the classic Ham-Sandwich Theorem. Their result requires that the point sets obey well-separation and weak general position, while the classic theorem always holds without these assumptions. Therefore, we call this result the *α-Ham-Sandwich theorem*, for a clearer distinction. Formally, given d finite point sets $P_1, \ldots, P_d \subset \mathbb{R}^d$ and any set of positive integers $\{\alpha_1, \ldots, \alpha_d\}$ satisfying $1 \leq \alpha_i \leq |P_i|$, for all $i \in [d]$, where $[d]$ denotes the set $\{1, \ldots, d\}$, an $(\alpha_1, \ldots, \alpha_d)$-*cut* is an oriented hyperplane H that contains one point from each set and satisfies $|H^+ \cap P_i| = \alpha_i$ for all $i \in [d]$, where H^+ is the closed positive half-space bounded by H.

▶ **Theorem 1.1** (α-Ham-Sandwich Theorem [11]). *Let $P_1, \ldots, P_d$ be finite, well-separated point sets in $\mathbb{R}^d$. Let $\alpha = (\alpha_1, \ldots, \alpha_d)$ be a vector, where $\alpha_i \in [|P_i|]$ for all $i \in [d]$.*
1. *If an α-cut exists, then it is unique.*

* Supported in part by ERC StG 757609.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

2. If P is in a sufficiently general position, then a cut exists for each choice of α.
This statement does not necessarily hold if the sets are not well-separated, see Figure 1.

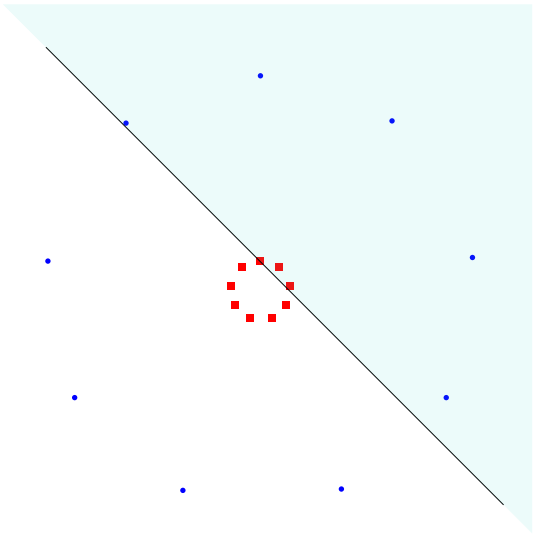


Figure 1 The red (square) and the blue (round) point sets are not well-separated. There is no halfplane that contains exactly three red and three blue points.

We call the associated computational search problem of finding the dividing hyperplane ALPHA-HS. Set $n = \sum_{i\in[d]} |P_i|$. Steiger and Zhao gave an algorithm that computes the dividing hyperplane in $O\left(n(\log n)^{d-3}\right)$ time, which is exponential in d. Later, Bereg [3] improved this algorithm to achieve a running time of $n2^{O(d)}$, which is linear in n but still exponential in d. No polynomial algorithms are known for ALPHA-HS if d is not fixed. Despite their superficial similarity, it is not immediately apparent whether the classic Ham-Sandwich theorem problem and ALPHA-HS are comparable in terms of their complexity. Due to the additional requirements on an input for ALPHA-HS, an instance of the Ham-sandwich problem may not be reducible to ALPHA-HS in general.

ALPHA-HS is a total search problem and is modeled by the complexity class TFNP (Total Function Nondeterministic Polynomial) of NP-search problems that always admit a solution. A noteworthy sub-class is CLS (continuous local search), that was introduced by Daskalakis and Papadimitriou [4]. It models optimization problems that can be solved by local search over a continuous domain using a continuous potential function. Recently there have been increasing efforts towards mapping the complexity landscape of existence theorems in high-dimensional discrete geometry in such classes. It was shown in [6] that the search problem for the Ham-Sandwich theorem is complete for PPA. Finding a solution to the Colorful Carathéodory problem [1] was shown to lie in the intersection PPAD $\cap$ PLS [9, 10]. Here, PPAD $\subseteq$ PPA, CLS $\subseteq$ PLS $\cap$ PPAD are other sub-classes of TFNP.

Recently, Fearley et al. [5] defined a sub-class of CLS by the name *Unique End of Potential Line* that represents problems in CLS with unique solutions. They define it through a canonical complete problem UNIQUEEOPL:

▶ **Definition 1.2** (from [5]). Let n, m be positive integers. The input consists of
- a pair of Boolean circuits $\mathbb{S}, \mathbb{P} : \{0,1\}^n \to \{0,1\}^n$ such that $\mathbb{P}(0^n) = 0^n \neq \mathbb{S}(0^n)$, and
- a Boolean circuit $\mathbb{V} : \{0,1\}^n \to \{0,1,\ldots,2^m-1\}$ such that $\mathbb{V}(0^n) = 0$,

each circuit having poly(n, m) size. The UNIQUEEOPL problem is to report one of the following:

(U1). A point $v \in \{0,1\}^n$ such that $\mathbb{P}(\mathbb{S}(v)) \neq v$.

(UV1). A point $v \in \{0,1\}^n$ such that $\mathbb{S}(v) \neq v$, $\mathbb{P}(\mathbb{S}(v)) = v$, and $\mathbb{V}(\mathbb{S}(v)) - \mathbb{V}(v) \leq 0$.
(UV2). A point $v \in \{0,1\}^n$ such that $\mathbb{S}(\mathbb{P}(v)) \neq v \neq 0^n$.
(UV3). Two points $v, u \in \{0,1\}^n$ such that $v \neq u$, $\mathbb{S}(v) \neq v$, $\mathbb{S}(u) \neq u$, and either $\mathbb{V}(v) = \mathbb{V}(u)$ or $\mathbb{V}(v) < \mathbb{V}(u) < \mathbb{V}(\mathbb{S}(v))$.

The problem defines a graph G with up to 2^n vertices. Informally, $\mathbb{S}(\cdot), \mathbb{P}(\cdot), \mathbb{V}(\cdot)$ represent the *successor*, *predecessor* and *potential* functions that act on the vertices. There is an edge $(u, v) \in G$ if and only if $\mathbb{S}(u) = v$, $\mathbb{P}(v) = u$ and $\mathbb{V}(u) < \mathbb{V}(v)$. Thus, G is a directed path (line) along which the potential strictly increases. $\mathbb{S}(\mathbb{P}(x)) \neq x$ represents a start of a line, $\mathbb{P}(\mathbb{S}(x)) \neq x$ represents the end, $\mathbb{P}(\mathbb{S}(x)) = x$ otherwise, and 0^n is a given starting vertex.

(U1) is a solution representing the end of a path. **(UV1)**, **(UV2)** and **(UV3)** are violations. **(UV1)** gives a violation of our assumption that $\mathbb{V}$ increases strictly along the path. **(UV2)** gives a start of a path that is not 0^n. **(UV3)** shows that G has more than one path. If there are no violations, G is a single path starting at 0^n and ending at **(U1)**. UniqueEOPL is formulated in the non-promise setting, placing it in TFNP. UEOPL contains three classical problems [5], including finding the fixed point of a contraction map.

A notion of *promise-preserving* reductions is also defined in [5]. A reduction from problem X to Y is said to be promise-preserving, if whenever it is promised that X has no violations, then the reduced instance of Y is free of violations. Such a reduction would imply that whenever the original problem is free of violations, then the reduced instance always has a single line that ends at a valid solution.

Contributions. We formalize the search problem for Alpha-HS in a non-promise setting:

▶ **Definition 1.3** (Alpha-HS)**.** Given d finite point sets $P = P_1 \cup \cdots \cup P_d \subset \mathbb{R}^d$ each interpreted as a different color, and a vector $(\alpha_1, \ldots, \alpha_d)$ of positive integers such that $\alpha_i \leq |P_i|$ for $i \in [d]$, the Alpha-HS problem is to find one of the following:

(G1). A $(\alpha_1, \ldots, \alpha_d)$-cut.

(GV1). A subset of P of size $d+1$ and at least $d-1$ colors that lies on a hyperplane.
(GV2). A disjoint pair of sets $I, J \subset [d]$ such that $\text{conv}(\{\cup_{i \in I} P_i\}) \cap \text{conv}(\{\cup_{j \in J} P_j\}) \neq \emptyset$.

(G1) corresponds to a solution representing a valid cut, while **(GV1)** and **(GV2)** refer to violations of weak general position and well-separation, respectively. From Theorem 1.1 we see that **(G1)** is guaranteed if no violations are presented, so that Alpha-HS is a total search problem. We give the first non-trivial complexity-theoretic upper bound for Alpha-HS:

▶ **Theorem 1.4.** *There is a* poly(n, d)*-time promise-preserving reduction from* Alpha-HS *to* UniqueEOPL*, so that* Alpha-HS $\in$ UEOPL $\subseteq$ CLS*.*

It is not surprising to discover that Alpha-HS $\in$ PPAD, since the proof of the continuous version [2] was based on Brouwer's Fixed Point Theorem. The observation that it also lies in PLS is new and noteworthy, putting Alpha-HS into the reach of local search algorithms. See Figure 2 for a pictorial view.

2 Alpha-HS is in UEOPL

For space reasons, we cannot provide much technical detail. Instead, we give a broad overview and some difficulties we encountered. We call a hyperplane *colorful* if it passes through exactly d colorful points $p_1, \ldots, p_d \subset P$. Otherwise, we call the hyperplane *non-colorful*. We

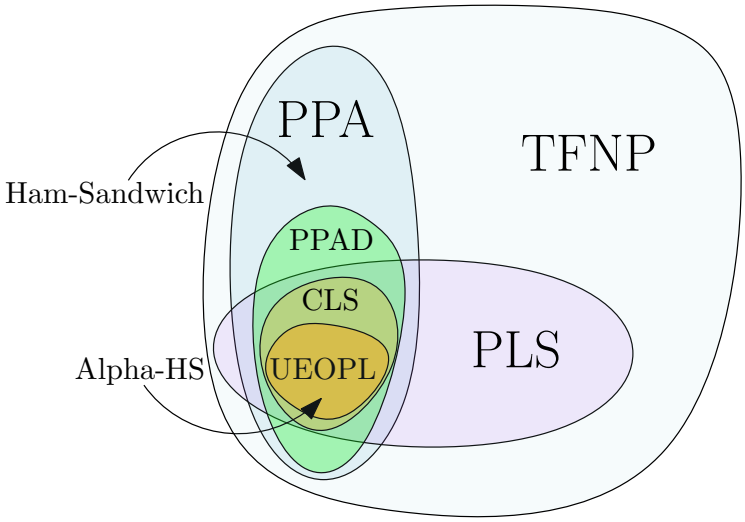


Figure 2 The hierarchy of complexity classes.

follow the notation of [11] to define the orientation of hyperplanes. If a hyperplane is colorful, the orientation is determined by the d colorful points. If a hyperplane is non-colorful, we design a deterministic way to pick a point in the intersection of the convex of the missing color with the hyperplane to define the orientation (see Figure 3). The *α-vector* of any oriented hyperplane H is a d-tuple $(\alpha_1, \ldots, \alpha_d)$ of integers where α_i is the number of points of P_i in the closed halfspace H^+ for $i \in [d]$.

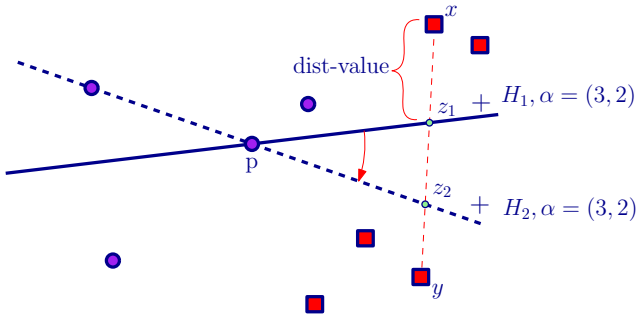


Figure 3 Purple (disk) is the first color and red (square) is the second color. H_2 is a hyperplane that rotates from H_1 at the anchor p. x, y are the highest ranked points of red color on each side of H_1, H_2 under a given order. The orientations of H_1, H_2 are determined by p and z_1, z_2 respectively.

Our intuition is based on rotating a colorful hyperplane H to another colorful hyperplane H' through a sequence of local changes of the points on the hyperplanes such that the α-vector of H' increases in some coordinate by one from that of H. The hyperplane rotates about an *anchor*, which is a colorful $(d-1)$-tuple of P that spans a $(d-2)$-flat. Whenever the non-colorful hyperplane hits a new point of a repeated color, the point in the anchor of the same color is swapped with it and continues the rotation until a point of the missing color is hit (see Figure 4). Roughly speaking, the colorful hyperplanes represent the vertices of the UniqueEOPL instance and the rotations determine the edges. We first describe our approach assuming that both well-separation and sufficient general position hold. We then describe how to handle the cases when these assumptions are violated.

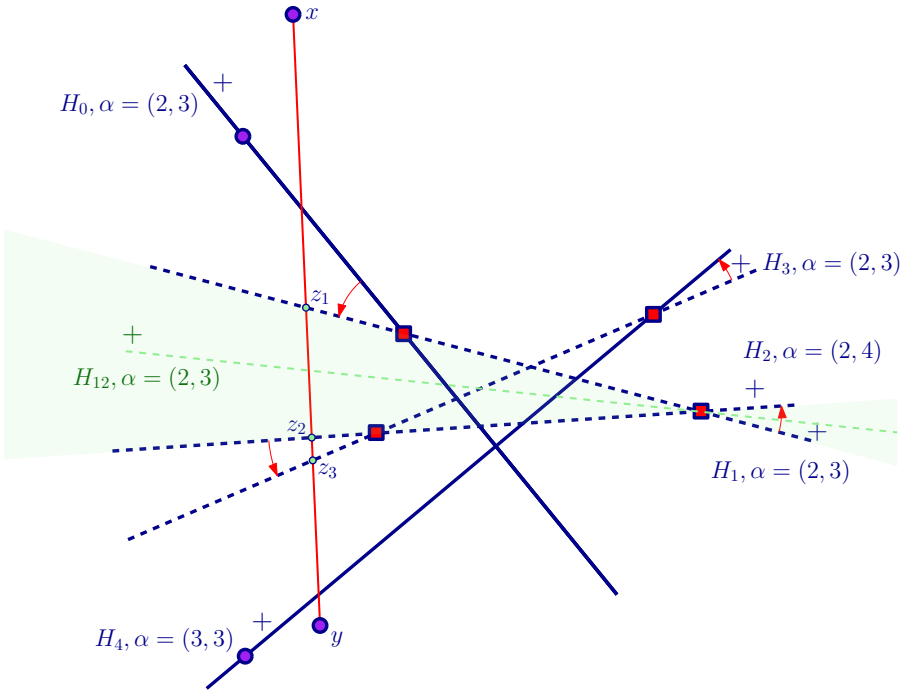


Figure 4 An example showing a sequence of rotations from H_0 to H_4 through H_1, H_2, H_3. Purple (disk) is the first color and red (square) is the second color. This sequence represents a path between two vertices in the UniqueEOPL graph that is generated in the reduction. The shaded region represents a rotation and H_{12} is its angular bisector. The segment xy is used to define the orientations of H_1, H_2, H_3, H_{12}.

Canonical path. Each colorful hyperplane H is incident to a colorful set of d points. This set of points defines d possible anchors, and each anchor can be used to rotate H in a different fashion. To define a unique sequence of rotations, we pick a specific order as follows: first, we assume that the colorful hyperplane H whose α-vector is $(1, \ldots, 1)$ is given (we show later how this assumption can be removed). We start at H and pick the anchor that excludes the first color, then apply a sequence of rotations until we hit another colorful hyperplane with α-vector $(2, 1, \ldots, 1)$. Similarly, we move to a colorful hyperplane with α-vector $(3, 1, \ldots, 1)$ and so on until we reach $(\alpha_1, 1, \ldots, 1)$. Then, we repeat this for the other colors in order to reach $(\alpha_1, \alpha_2, 1, \ldots, 1)$ and so on until we reach the target α-vector. This pattern of α-vectors helps in defining a potential function that strictly increases along the path. We can encode this sequence of rotations as a unique path in the UniqueEOPL instance, and we call it *canonical path*.

Distance parameter and potential function. The α-vector is not sufficient to define the potential function, since the sequence of rotations between two colorful hyperplanes may have the same α-vector. For instance, the angular bisectors of the rotations in $H_0, \ldots, H_3$ in Figure 4 all have the same α-vector. Hence, we need an additional measurement in order to determine the direction of rotation that increases the α-vector. Similar to how we define the orientation for a non-colorful hyperplane H, we deterministically select a directed segment xy that intersects H. We define a distance parameter called *dist-value* of H to be the distance from x to the intersection point (see Figure 3). We define a potential value for each vertex on the canonical path in UniqueEOPL using the sum of weighed components of α-vector

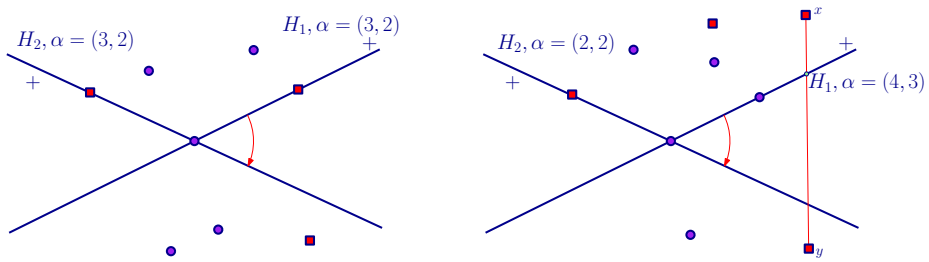


Figure 5 The examples show two sets of points that are not well-separated. Purple (circle) represents the first color and red (square) represents the second color. In both examples the rotation procedure does not increase the α-vector. Both examples show that the orientation of the hyperplane may be flipped after the rotation, so the resulting α-vector can go wrong.

and dist-value for the tie-breaker.

We do not need to know the vertex with α-vector $(1, \ldots, 1)$ in advance. We split the problem into two sub-problems: in the first we start with a copy of G and any arbitrary vertex. We reverse the direction determined by the potential and construct a Alpha-HS instance for which the vertex with α-vector $(1, \ldots, 1)$ is the solution. In the second, we use this vertex as the input to the main Alpha-HS instance. If the input is free of violations, then both sub-problems give valid solutions and together they answer the original question.

Handling violations. We show that if there are no violations, then the reduced instance of UniqueEOPL only gives a **(U1)** solution, which readily translates to a **(G1)** solution, so our reduction is promise-preserving, and this can be done in poly(n, d) time.

If P violates well-separation or weak general position, there may be multiple solutions for the same α-cut (see Figure 5, left), and no solutions for other cuts. Many nice properties of rotations are destroyed because the orientation of the rotating hyperplane may flip. For instance, the α-vector may fail to increment (see Figure 5, right). From the point of view of the canonical path we create, the path may be split into several pieces, which fails the assumption of the unique line. The vertex that corresponds to the target α-vector may not exist.

We design our reduction in such a way that any violations on the canonical path can be captured from the violations of the UniqueEOPL instance. After we obtain a violation solution from the reduced instance, we can process it to generate a certificate that witnesses a violation of Alpha-HS. When weak general position fails, then the hyperplanes may have additional points of P. These give rise to many different d-tuples (each corresponding to some vertex in the UniqueEOPL graph G) that represent the same hyperplane. We join these vertices to form a cycle in G. For some other case, we show that when two hyperplanes have the same α-vector (and dist-value for non-colorful), we can compute a witness for the violation of well-separation. To summarize, we show how to compute a

- **(GV1)** solution from a **(UV1)** solution.
- **(GV1)** or **(GV2)** solution, given a **(UV2)** or **(UV3)** solution.
- **(GV1)** or **(GV2)** solution, that occurs with a **(U1)** solution with the incorrect α-vector.

We show that converting these solutions always takes poly(n, d) time. See Figure 6 for an example.

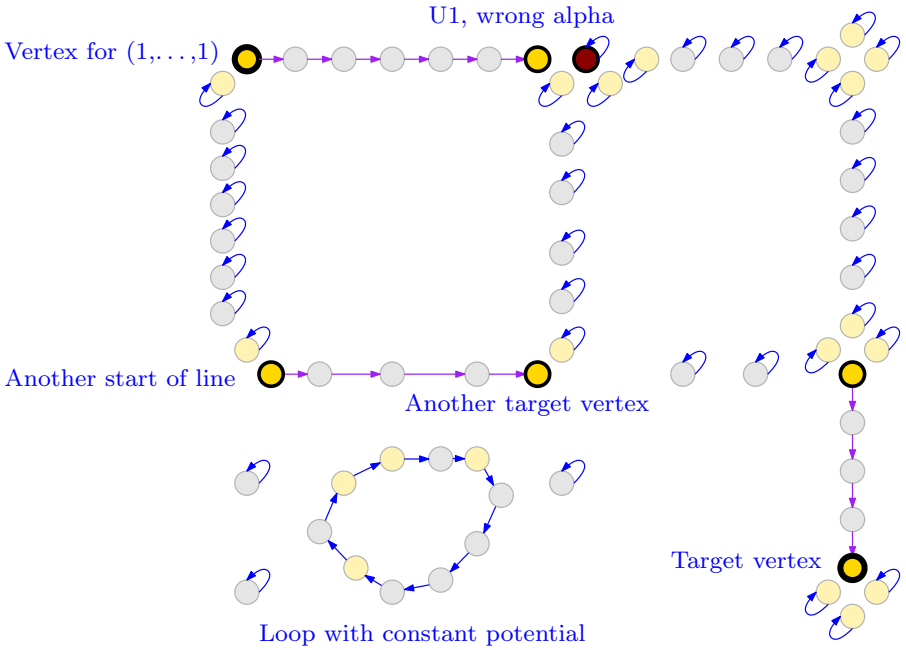


Figure 6 A subgraph with multiple violations. The vertices that are not on the canonical path are isolated by self-loops. Some vertex that witnesses a violation splits the canonical path into two. Since the orientations are not consistent, there may exist multiple paths that contain vertices with the same α-vector.

3 Conclusion and future work

We gave an upper bound on the complexity of Alpha-HS. The next question is determining if the problem is hard for UEOPL. One challenge is that UniqueEOPL is formulated as Boolean circuits, whereas Alpha-HS is purely geometric. Emulating circuits using purely geometric arguments is highly non-trivial. It could be worthwhile to investigate if the techniques used in [6] can prove useful in answering this question.

References

1 Imre Bárány. A generalization of Carathéodory's theorem. *Discrete Mathematics*, 40(2-3):141–152, 1982.

2 Imre Bárány, Alfredo Hubard, and Jesús Jerónimo. Slicing convex sets and measures by a hyperplane. *Discrete Comput. Geom.*, 39(1):67–75, 2008.

3 Sergey Bereg. Computing generalized ham-sandwich cuts. *Inform. Process. Lett.*, 112(13):532–534, 2012.

4 Constantinos Daskalakis and Christos H. Papadimitriou. Continuous Local Search. In *Proc. 22nd Annu. ACM-SIAM Sympos. Discrete Algorithms (SODA)*, pages 790–804, 2011.

5 John Fearnley, Spencer Gordon, Ruta Mehta, and Rahul Savani. Unique End of Potential Line. In *Proc. 46th Internat. Colloq. Automata Lang. Program. (ICALP)*, pages 56:1–56:15, 2019.

6 Aris Filos-Ratsikas and Paul W. Goldberg. The complexity of splitting Necklaces and bisecting Ham sandwiches. In *Proc. 51st Annu. ACM Sympos. Theory Comput. (STOC)*, pages 638–649, 2019.

7 Chi-Yuan Lo, Jiří Matoušek, and William L. Steiger. Algorithms for Ham-sandwich cuts. *Discrete Comput. Geom.*, 11:433–452, 1994.

8 Jiří Matoušek. *Using the Borsuk-Ulam theorem.* Springer-Verlag Berlin Heidelberg, 2003.

9 Frédéric Meunier, Wolfgang Mulzer, Pauline Sarrabezolles, and Yannik Stein. The rainbow at the end of the line: A PPAD formulation of the Colorful Carathéodory theorem with applications. In *Proc. 28th Annu. ACM-SIAM Sympos. Discrete Algorithms (SODA)*, pages 1342–1351, 2017.

10 Wolfgang Mulzer and Yannik Stein. Computational aspects of the Colorful carathéodory theorem. *Discrete Comput. Geom.*, 60(3):720–755, 2018.

11 William Steiger and Jihui Zhao. Generalized Ham-sandwich cuts. *Discrete Comput. Geom.*, 44(3):535–545, 2010.

12 Arthur H. Stone and John W. Tukey. Generalized "Sandwich" theorems. *Duke Mathematical Journal*, 9(2):356–359, 06 1942.

Graph Planarity Testing with Hierarchical Embedding Constraints*

Giuseppe Liotta[1], Ignaz Rutter[2], and Alessandra Tappini[1]

1 **Dipartimento di Ingegneria, Università degli Studi di Perugia, Italy**
giuseppe.liotta@unipg.it, alessandra.tappini@studenti.unipg.it
2 **Department of Computer Science and Mathematics, University of Passau, Germany**
rutter@fim.uni-passau.de

Abstract

Hierarchical embedding constraints define a set of allowed cyclic orders for the edges incident to the vertices of a graph. These constraints are expressed in terms of FPQ-trees. FPQ-trees are a variant of PQ-trees that includes F-nodes in addition to P-nodes and to Q-nodes: An F-node represents a permutation that is fixed, i.e., it cannot be reversed. Let G be a graph such that every vertex of G is equipped with a set of FPQ-trees encoding hierarchical embedding constraints for its incident edges. We study the problem of testing whether G admits a planar embedding such that, for each vertex v of G, the cyclic order of the edges incident to v is described by at least one of the FPQ-trees associated with v. We prove that the problem is fixed-parameter tractable for biconnected graphs, where the parameters are the treewidth of G and the number of FPQ-trees associated with every vertex. We also show that the problem is NP-complete if parameterized by the number of FPQ-trees only, and W[1]-hard if parameterized by the treewidth only.

1 Introduction

The study of graph planarity testing and of its variants is at the heart of graph algorithms and of their applications. This paper is inspired by a work of Gutwenger et al. [7], who study the graph planarity testing problem subject to hierarchical embedding constraints. Hierarchical embedding constraints specify for each vertex v of G which cyclic orders of the edges incident to v are admissible in a constrained planar embedding of G. For example, Fig. 1 shows the edges incident to a vertex v and a set of hierarchical embedding constraints on these edges. Edges are partitioned into four sets, denoted as E_1, E_2, E_3, and E_4; the constraints allow only two distinct clockwise cyclic orders for these edge-sets, namely either $E_1E_2E_3E_4$ (Fig. 1a) or $E_1E_3E_2E_4$ (Fig. 1b). Within each set, the constraints of Fig. 1 allow the edges of E_1, E_2, and E_3 to be arbitrarily permuted, while the edges of E_4 are partitioned into three subsets E_4', E_4'', and E_4''' such that E_4'' must appear between E_4' and E_4''' in the clockwise order around v. The edges of E_4' can be arbitrarily permuted, while the edges of E_4'' and the edges of E_4''' have only two possible orders that are the reverse of one another.

Hierarchical embedding constraints can be encoded by using FPQ-trees, a variant of PQ-trees that includes F-nodes in addition to P-nodes and to Q-nodes. An F-node encodes a permutation that cannot be reversed. For example, the hierarchical embedding constraints of Fig. 1 can be represented by two FPQ-trees denoted as T and T' in Fig. 1a and 1b.

Gutwenger et al. [7] study the planarity testing problem with hierarchical embedding

* Work partially supported by: ***(i)*** MIUR, under grant 20174LF3T8 "AHeAD: efficient Algorithms for HArnessing networked Data"; ***(ii)*** Dipartimento di Ingegneria - Università degli Studi di Perugia, under grant RICBA19FM: "Modelli, algoritmi e sistemi per la visualizzazione di grafi e reti"; ***(iii)*** German Science Foundation (DFG), under grant Ru 1903/3-1.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

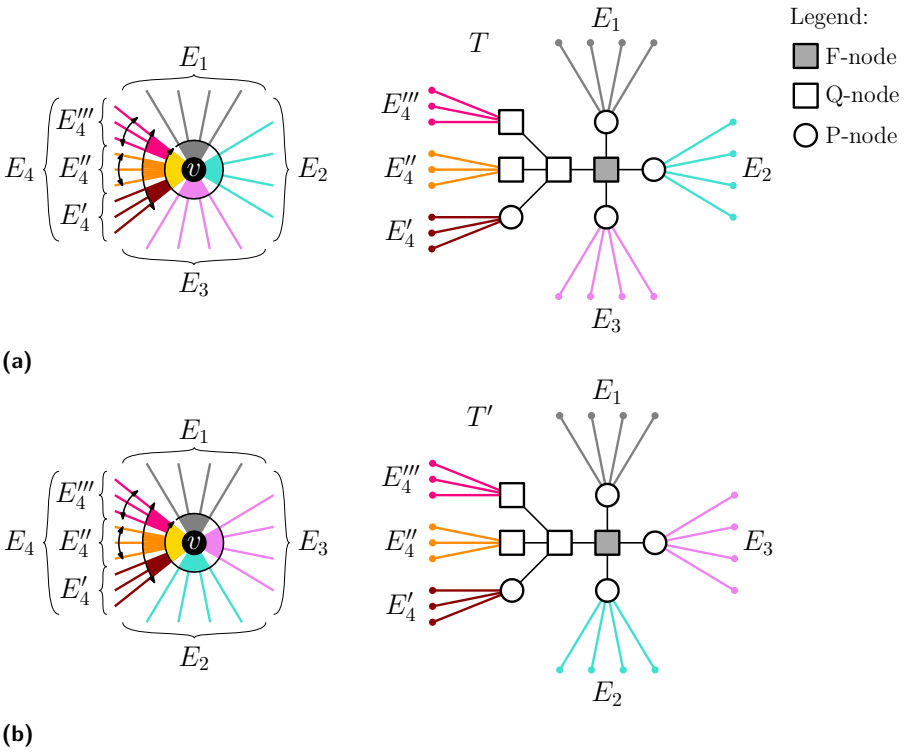


Figure 1 Two examples of a vertex v with hierarchical embedding constraints and the corresponding FPQ-trees. F-nodes are shaded boxes, Q-nodes are white boxes, and P-nodes are circles.

constraints by allowing *at most one* FPQ-tree per vertex. We generalize their study and allow *more than one* FPQ-tree associated with each vertex. Our main results are the following.

- We show that FPQ-Choosable Planarity Testing is NP-complete even if the number of FPQ-trees associated with each vertex is bounded by a constant greater than 1, and it remains NP-complete even if the FPQ-trees only contain P-nodes. This contrasts with the result of Gutwenger et al. [7] who prove that FPQ-Choosable Planarity Testing can be solved in linear time when each vertex is equipped with at most one FPQ-tree.
- We prove that FPQ-Choosable Planarity Testing is W[1]-hard if parameterized by treewidth, and that it remains W[1]-hard even when the FPQ-trees only contain P-nodes.
- The above results imply that FPQ-Choosable Planarity Testing is not fixed-parameter tractable if parameterized by treewidth only or by the number of FPQ-trees per vertex only. For a contrast, we show that FPQ-Choosable Planarity Testing becomes fixed-parameter tractable for biconnected graphs when parameterized by both the treewidth and the number of FPQ-trees associated with every vertex.

Proofs and details omitted from this extended abstract can be found in the full version [8].

Preliminaries. We assume familiarity with graph theory and algorithms, and with the concepts of PQ-tree, SPQR-decomposition tree, branchwidth, treewidth and sphere-cut decomposition of a graph [3, 4, 5, 6, 9]. We only briefly recall some of the basic concepts that will be used extensively in the rest of the paper (see also [1]).

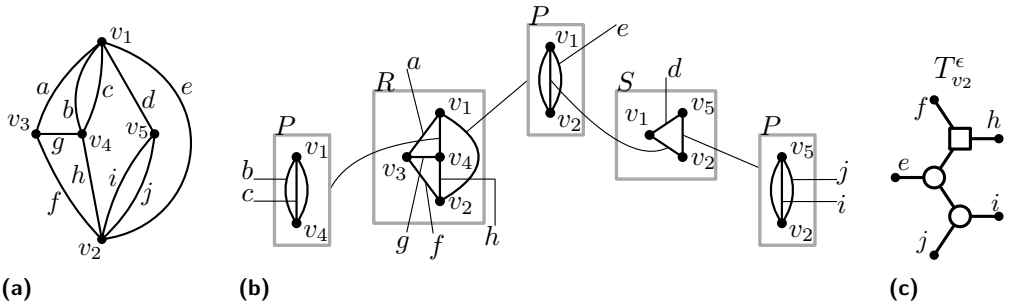


Figure 2 (a) A biconnected planar graph G. (b) An SPQR-decomposition tree of G. (c) The embedding tree of v_2.

Given a graph G together with a fixed combinatorial embedding, we can associate with each vertex v a PQ-tree T_v whose leaves represent the edges incident to v. The tree T_v encodes a set of permutations for its leaves: Each of these permutations is in a bijection with a cyclic order of the edges incident to v. If there is a permutation π_v of the leaves of T_v that is in a bijection with a cyclic order σ_v of the edges incident to v, we say that T_v *represents* σ_v, or equivalently that σ_v *is represented by* T_v. An *FPQ-tree* is a PQ-tree where, for some of the Q-nodes, the reversal of the permutation described by their children is not allowed. To distinguish these Q-nodes from the regular ones, we call them *F-nodes*.

The planar combinatorial embeddings that are given by the SPQR-decomposition tree of a biconnected graph G give constraints on the cyclic order of edges around each vertex of G. These constraints can be encoded by associating a PQ-tree with each vertex v of G, called the *embedding tree* of v and denoted by T^{ϵ}_v (see, e.g., [2]). For example, Fig. 2c shows the embedding tree $T^{\epsilon}_{v_2}$ of the vertex v_2 in Fig. 2a. Note that edges f and h (i and j, resp.) belong to an R-node (a P-node, resp.) in the SPQR-decomposition tree of G (Fig. 2b), hence the corresponding leaves are connected to a Q-node (a P-node, resp.) in $T^{\epsilon}_{v_2}$.

2 The FPQ-choosable Planarity Testing Problem

Let $G = (V, E)$ be a (multi-)graph, let $v \in V$, and let T_v be an FPQ-tree whose leaf set is $E(v)$. We define *consistent*(T_v) as the set of cyclic orders of the edges incident to v in a planar embedding $\mathcal{E}$ of G that are represented by the FPQ-tree T_v. An *FPQ-choosable graph* is a pair (G, D) where $G = (V, E)$ is a (multi-)graph, and D is a mapping that associates each vertex $v \in V$ with a set $D(v)$ of FPQ-trees whose leaf set is $E(v)$. Given a planar embedding $\mathcal{E}$ of G, we denote by $\mathcal{E}(v)$ the cyclic order of edges incident to v in $\mathcal{E}$. An *assignment* A is a function that assigns to each vertex $v \in V$ an FPQ-tree in $D(v)$. We say that A is *compatible with* G if there exists a planar embedding $\mathcal{E}$ of G such that $\mathcal{E}(v) \in$ *consistent*$(A(v))$ for all $v \in V$. In this case, we also say that $\mathcal{E}$ is *consistent with* A. An FPQ-choosable graph (G, D) is *FPQ-choosable planar* if there exists an assignment that is compatible with G. Refer to Fig. 3 for an example.

The FPQ-Choosable Planarity Testing problem receives as input an FPQ-choosable graph (G, D) and it asks whether (G, D) is FPQ-choosable planar. In the rest of the paper we assume that G is a biconnected (multi-)graph. Clearly G must be planar or else the problem becomes trivial. Also, any assignment that is compatible with G must define a planar embedding of G among those described by an SPQR-decomposition tree of G. Therefore, a preliminary step for an algorithm that tests whether (G, D) is FPQ-choosable

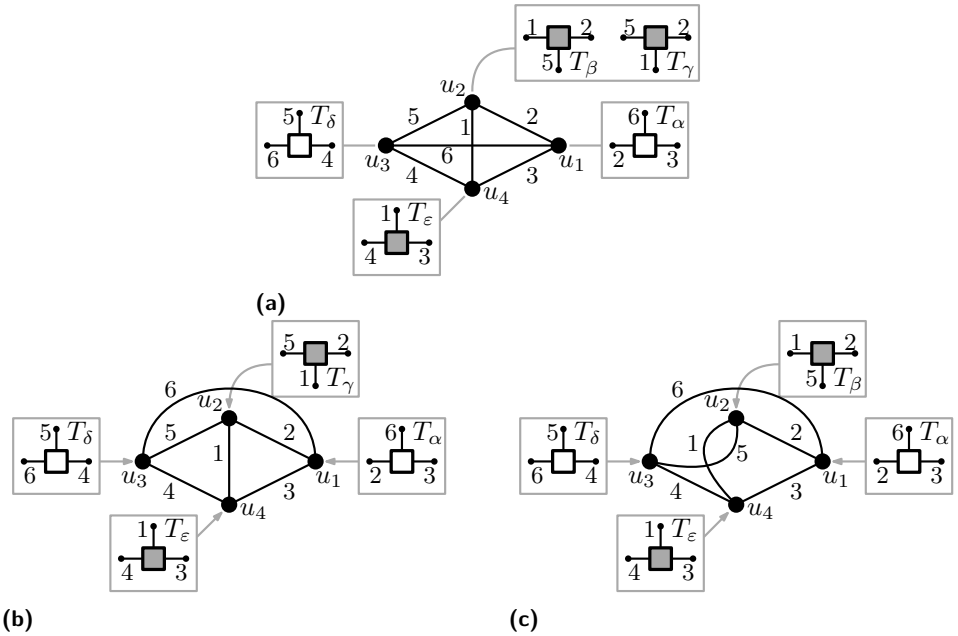


Figure 3 (a) An FPQ-choosable planar graph (G, D). (b) A planar embedding of G that is consistent with assignment $\{A(u_1) = T_\alpha, A(u_2) = T_\gamma, A(u_3) = T_\delta, A(u_4) = T_\varepsilon\}$; the assignment is compatible with G. (c) A non-planar embedding of G that is consistent with assignment $\{A(u_1) = T_\alpha, A(u_2) = T_\beta, A(u_3) = T_\delta, A(u_4) = T_\varepsilon\}$; there is no planar embedding that is consistent with A.

planar is to intersect each FPQ-tree $T_v \in D(v)$ with the embedding tree T_v^ϵ of v, so that the cyclic order of the edges incident to v satisfies both the constraints given by T_v and the ones given by T_v^ϵ. (See, e.g., [2] for details about the operation of intersection between two PQ-trees, whose extension to the case of FPQ-trees is straightforward). We assume that the FPQ-trees of D have been intersected with the corresponding embedding trees and we still denote by $D(v)$ the set of FPQ-trees associated with v and resulting from the intersection.

3 Complexity of FPQ-choosable Planarity Testing

As we are going to show, FPQ-Choosable Planarity Testing is fixed-parameter tractable when parameterized by treewidth and number of FPQ-trees per vertex. One may wonder whether the problem remains FPT if parameterized by the treewidth only or by the number of FPQ-trees per vertex only. The following theorems answer this question in the negative.

▶ **Theorem 3.1.** FPQ-Choosable Planarity Testing *with a bounded number of FPQ-trees per vertex is NP-complete. It is NP-complete even if the FPQ-trees have only P-nodes.*

▶ **Theorem 3.2.** FPQ-Choosable Planarity Testing *parameterized by treewidth is W[1]-hard. It is W[1]-hard even if the FPQ-trees have only P-nodes.*

4 Fixed Parameter Tractability of FPQ-choosable Planarity Testing

In this section, we introduce some concepts that are fundamental to the description of the algorithm and we present a polynomial-time testing algorithm for graphs having bounded branchwidth and such that the number of FPQ-trees associated with each vertex is bounded

by a constant. Note that, for a graph G with treewidth t and branchwidth $b > 1$, it holds that $b - 1 \leq t \leq \lfloor \frac{3}{2} b \rfloor - 1$ [9].

Let T be an FPQ-tree, let leaves(T) denote the set of its leaves, and let L be a proper subset of leaves(T). We denote by σ a cyclic order of the leaves of an FPQ-tree, and we say that $\sigma \in$ *consistent*(T) if the FPQ-tree T represents σ. We say that L is a *consecutive set* if the leaves in L are consecutive in every cyclic order represented by T. Let e be an edge of T, and let T' and T'' be the two subtrees obtained by removing e from T. If either leaves(T') or leaves(T'') are a subset of a consecutive set L, then we say that e is a *split edge for* L. The subtree that contains the leaves in L is the *split subtree* of e for L. A split edge e is *maximal* for L if there exists no split edge e' such that the split subtree of e' contains e.

▶ **Lemma 4.1.** *Let T be an FPQ-tree, let L be a consecutive proper subset of* leaves(T)*, and let S be the set of maximal split edges for L. Then either $|S| = 1$, or $|S| > 1$ and there exists a Q-node (or an F-node) χ of T such that χ has degree at least $|S| + 2$ and the elements of S appear consecutive around χ.*

If $|S| = 1$, the split edge in S is called the *boundary of* L. If $|S| > 1$, the Q-node (or F-node) χ defined in the statement of Lemma 4.1 is the *boundary of* L. Since F-nodes are a more constrained version of Q-nodes, when we refer to boundary Q-nodes we also take into account the case of F-nodes. Fig. 4a shows an FPQ-choosable graph (G, D) and two FPQ-trees $T_u \in D(u)$ and $T_v \in D(v)$. The three red edges b, c, and d of G define a consecutive set L_u in T_u; the edges e and f define a consecutive set L_v in T_v. The boundary of L_u in T_u is a Q-node, while the boundary of L_v in T_v is an edge.

We denote by $\mathcal{B}(L)$ the boundary of a set of leaves L. If $\mathcal{B}(L)$ is a Q-node, we associate $\mathcal{B}(L)$ with a default orientation that arbitrarily defines one of the two possible permutations of its children. This default orientation is called the *clockwise orientation* of $\mathcal{B}(L)$, while the other possible permutation of the children of $\mathcal{B}(L)$ is the *counter-clockwise orientation*.

Let $L' = L \cup \{\ell\}$, where ℓ is a new element. Let $\sigma \in$ *consistent*(T), and let $\sigma|_{L'}$ be a cyclic order obtained from σ by replacing the elements of the consecutive set leaves(T)$\setminus L$ by the single element ℓ. We say that a cyclic order σ' of L' is *extensible* if there exists a cyclic order $\sigma \in$ *consistent*(T) with $\sigma|_{L'} = \sigma'$ (and σ is an *extension of* σ'). An extensible order σ is *clockwise* if the orientation of χ is clockwise; σ is *counter-clockwise* otherwise. If the boundary of L is an edge, we consider any extensible order as both clockwise and counter-clockwise.

Let (G, D) be an FPQ-choosable graph, let $\mathcal{T}$ be an SPQR-decomposition tree of G and let v be a pole of a node μ of $\mathcal{T}$, let $T_v \in D(v)$ be an FPQ-tree associated with v, let E_{ext} be the set of edges that are incident to v and not contained in the pertinent graph G_μ, and let $E^\star_\mu(v) = E(v) \setminus E_{\text{ext}}$. Note that there is a bijection between the edges $E(v)$ of G and the leaves of T_v, hence we shall refer to the set of leaves of T_v as $E(v)$. Also note that $E^\star_\mu(v)$ is represented by a consecutive set of leaves in T_v, because in every planar embedding of G the edges in $E^\star_\mu(v)$ must appear consecutively in the cyclic order of the edges incident to v.

The *pertinent FPQ-tree* of T_v, denoted as $\text{Pert}_\mu(T_v)$, is the FPQ-tree obtained from T_v by replacing the consecutive set E_{ext} with a single leaf ℓ. Informally, the pertinent FPQ-tree of v describes the hierarchical embedding constraints for v within G_μ. For example, in Fig. 4b a pertinent graph G_μ with poles u and v is highlighted by a shaded region; the pertinent FPQ-tree $\text{Pert}_\mu(T_u)$ of T_u and the pertinent FPQ-tree $\text{Pert}_\mu(T_v)$ of T_v are obtained by the FPQ-trees T_u and T_v of Fig. 4a.

Let $\nu_1, \ldots, \nu_k$ be the children of μ in $\mathcal{T}$. Observe that the edges $E^\star_{\nu_i}(v)$ of each G_{ν_i} $(1 \leq i \leq k)$ form a consecutive set of leaves of $A_\mu(v) = \text{Pert}_\mu(T_v)$. The *skeletal FPQ-tree* of $\text{Pert}_\mu(T_v)$, denoted by $\text{Skel}_\mu(T_v)$, is the tree obtained from $\text{Pert}_\mu(T_v)$ by replacing each of the

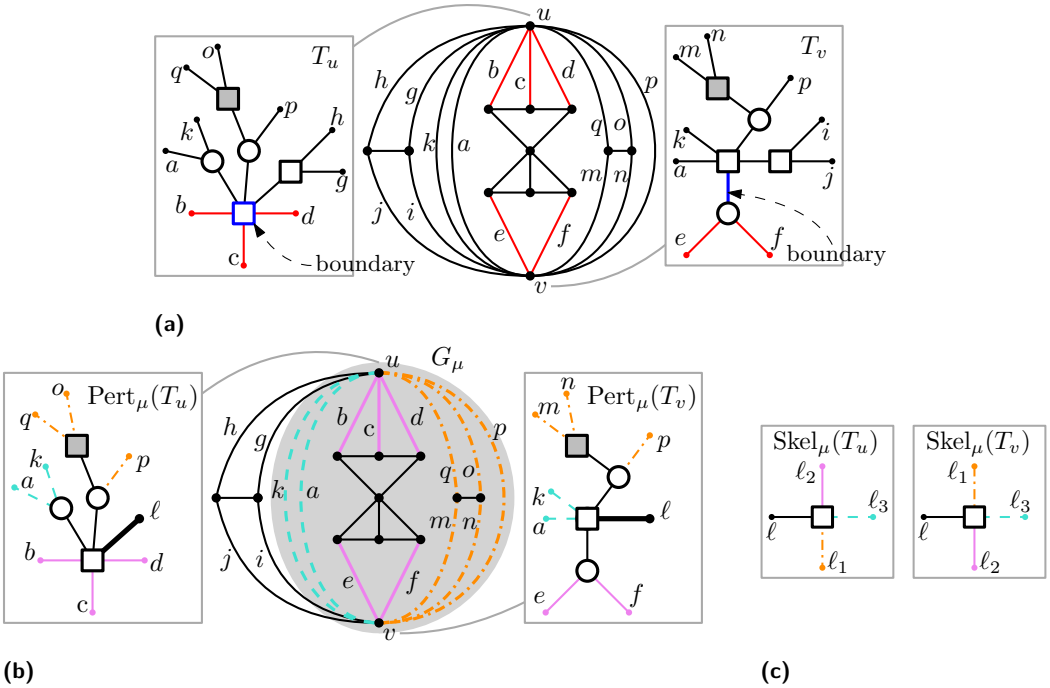


Figure 4 (a) A boundary Q-node in T_u and a boundary edge in T_v. (b) Pertinent FPQ-trees $\mathrm{Pert}_\mu(T_u)$ and $\mathrm{Pert}_\mu(T_v)$. (c) Skeletal FPQ-trees $\mathrm{Skel}_\mu(T_u)$ of $\mathrm{Pert}_\mu(T_u)$ and $\mathrm{Skel}_\mu(T_v)$ of $\mathrm{Pert}_\mu(T_v)$.

consecutive sets $E^\star_{\nu_i}(v)$ $(1 \le i \le k)$ by a single leaf ℓ_i (see Fig. 4c). Note that each Q-node of $\mathrm{Skel}_\mu(T_u)$ corresponds to a Q-node of $\mathrm{Pert}_\mu(T_u)$, and thus to a Q-node of T_u; also, distinct Q-nodes of $\mathrm{Skel}_\mu(T_u)$ correspond to distinct Q-nodes of $\mathrm{Pert}_\mu(T_u)$, and thus to distinct Q-nodes of T_u. For each Q-node χ of T_u that is a boundary of μ or of one of its children, there is a corresponding Q-node in $\mathrm{Skel}_\mu(T_u)$ that inherits its default orientation from T_u.

Let (G, D) be an FPQ-choosable graph, let $\mathcal{T}$ be an SPQR-decomposition tree of G, let μ be a node of $\mathcal{T}$, and let u and v be the poles of μ. We denote by (G_μ, D_μ) the FPQ-choosable graph consisting of the pertinent graph G_μ and the set D_μ that is defined as follows: $D_\mu(z) = D(z)$ for each vertex z of G_μ that is not a pole, and $D_\mu(v) = \{\mathrm{Pert}_\mu(T_v) \mid T_v \in D(v)\}$ if v is a pole of μ. A tuple $\langle T_u, T_v, o_u, o_v\rangle \in D(u) \times D(v) \times \{0,1\} \times \{0,1\}$ is *admissible for* G_μ if there exist an assignment A_μ of (G_μ, D_μ) and a planar embedding $\mathcal{E}_\mu$ of G_μ consistent with A_μ such that $A_\mu(u) = \mathrm{Pert}_\mu(T_u)$, $A_\mu(v) = \mathrm{Pert}_\mu(T_v)$, $\mathcal{B}(E^\star_\mu(u))$ is clockwise (counter-clockwise) in T_u if $o_u = 0$ $(o_u = 1)$, and $\mathcal{B}(E^\star_\mu(v))$ is clockwise (counter-clockwise) in T_v if $o_v = 0$ $(o_v = 1)$. A tuple is *admissible for* μ if it is admissible for G_μ. $\Psi(\mu)$ is the set of admissible tuples for G_μ.

FPT Algorithm: In order to test if (G, D) is FPQ-choosable planar, we root the SPQR-decomposition tree $\mathcal{T}$ at an arbitrary Q-node and we visit $\mathcal{T}$ from the leaves to the root. At each step of the visit, we equip the current node μ with the set $\Psi(\mu)$. If we encounter a node μ such that $\Psi(\mu) = \emptyset$, we return that (G, D) is not FPQ-choosable planar; otherwise the planarity test returns an affirmative answer. If the currently visited node μ is a leaf of $\mathcal{T}$, we set $\Psi(\mu) = D(u) \times D(v) \times \{0,1\} \times \{0,1\}$, because its pertinent graph is a single edge. If μ is an internal node, $\Psi(\mu)$ is computed from the sets of admissible tuples of the children of μ and depending on whether μ is an S-, P-, or R-node. In the case of R-nodes, we compute the set of admissible tuples by executing the sphere-cut decomposition of the skeleton of μ and by exploiting the fact that it has branchwidth at most b, where b is the branchwidth of G.

▶ **Theorem 4.2.** *Let (G, D) be a biconnected FPQ-choosable (multi-)graph such that $G = (V, E)$ and $|V| = n$. Let $D(v)$ be the set of FPQ-trees associated with vertex $v \in V$. There exists an $O(D_{\max}^{\frac{3}{2}b} \cdot n^2 + n^3)$-time algorithm to test whether (G, D) is FPQ-choosable planar, where b is the branchwidth of G and $D_{\max} = \max_{v \in V} |D(v)|$.*

As future work, it would be nice to extend Theorem 4.2 to simply connected graphs. Indeed, our proof is based on the SPQR-decomposition that assumes the biconnectivity of the input graph.

References

1 Subramanian Arumugam, Andreas Brandstädt, Takao Nishizeki, and Krishnaiyan Thulasiraman. *Handbook of graph theory, combinatorial optimization, and algorithms*. Chapman and Hall/CRC, 2016.

2 Thomas Bläsius and Ignaz Rutter. Simultaneous PQ-ordering with applications to constrained embedding problems. *ACM Trans. Algorithms*, 12(2):16:1–16:46, 2016.

3 Kellogg S. Booth and George S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *J. Comput. Syst. Sci.*, 13(3):335–379, 1976.

4 G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing*. Prentice Hall, Upper Saddle River, NJ, 1999.

5 Frederic Dorn, Eelko Penninkx, Hans L. Bodlaender, and Fedor V. Fomin. Efficient exact algorithms on planar graphs: Exploiting sphere cut decompositions. *Algorithmica*, 58(3):790–810, 2010.

6 Qian-Ping Gu and Hisao Tamaki. Optimal branch-decomposition of planar graphs in $O(n^3)$ time. *ACM Trans. Algorithms*, 4(3):30:1–30:13, 2008.

7 Carsten Gutwenger, Karsten Klein, and Petra Mutzel. Planarity testing and optimal edge insertion with embedding constraints. *J. Graph Algorithms Appl.*, 12(1):73–95, 2008.

8 Giuseppe Liotta, Ignaz Rutter, and Alessandra Tappini. Graph planarity testing with hierarchical embedding constraints. *CoRR*, abs/1904.12596, 2019. `arXiv:1904.12596`.

9 Neil Robertson and Paul D. Seymour. Graph minors. X. obstructions to tree-decomposition. *J. Comb. Theory, Ser. B*, 52(2):153–190, 1991.

On the edge-length ratio of 2-trees*

Václav Blažej[1], Jiří Fiala[2], and Giuseppe Liotta[3]

1 **Faculty of Information Technology, Czech Technical University in Prague, Prague, Czech Republic**
2 **Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic**
3 **Dipartimento di Ingegneria, Università degli Studi di Perugia, Italy**

Abstract

We study planar straight-line drawings of graphs that minimize the ratio between the length of the longest and the shortest edge. We answer a question of Lazard et al. [Theor. Comput. Sci. **770** (2019), 88–94] and, for any given constant r, we provide a 2-tree which does not admit a planar straight-line drawing with a ratio bounded by r. When the ratio is restricted to adjacent edges only, we prove that any 2-tree admits a planar straight-line drawing whose edge-length ratio is at most $4 + \varepsilon$ for any arbitrarily small $\varepsilon > 0$.

1 Introduction

Straight-line drawings of planar graphs are thoroughly studied both for their theoretical interest and their applications in a variety of disciplines (see, e.g., [7, 13]). Different quality measures for planar straight-line drawings have been considered in the literature, including area, angular resolution, slope number, average edge length, and total edge length (see, e.g., [9, 10, 12]).

This paper studies the problem of computing planar straight-line drawings of graphs where the length ratio of the longest to the shortest edge is as small as possible. We recall that the problem of deciding whether a graph admits a planar straight-line drawing with specified edge lengths is NP-complete even when restricted to 3-connected planar graphs [8] and the completeness persists in the case when all given lengths are equal [5]. In addition, deciding whether a degree-4 tree has a planar drawing such that all edges have the same length and the vertices are at integer grid points is NP-complete [1].

In the attempt of relaxing the edge length conditions which make the problem hard, Hoffmann et al. [10] propose to minimize the ratio between the longest and the shortest edges among all straight-line drawings of a graph. While the problem remains hard for general graphs (through approximation of unit disk graphs [6]), Lazard et al. prove [11] that any outerplanar graph admits a planar straight-line drawing such that the length ratio of the longest to the shortest edges is strictly less than 2. This result is tight in the sense that for any $\varepsilon > 0$ there are outerplanar graphs that cannot be drawn with an edge-length ratio smaller than $2 - \varepsilon$. Lazard et al. also ask whether their construction could be extended to the class of series-parallel graphs.

We answer this question in the negative sense, by showing that a subclass of series-parallel graphs, called 2-trees, does not allow any planar straight-line drawing of bounded edge-length ratio. (The class of 2-trees is defined constructively: an edge is a 2-tree, and

* The research was initiated during workshop Homonolo 2018. Research partially supported by MIUR, the Italian Ministry of Education, University and Research, under Grant 20174LF3T8 AHeAD: efficient Algorithms for HArnessing networked Data. V. Blažej acknowledges the support of the OP VVV MEYS funded project CZ.02.1.01/0.0/0.0/16_019/0000765 "Research Center for Informatics". The work of J. Fiala was supported by the grant 19-17314J of the GA ČR.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

modifying a 2-tree by adding a new vertex connected to two neighboring vertices is also a 2-tree.) In fact, a corollary of our main result is the existence of an $\Omega(\log n)$ lower bound for the edge-length ratio of planar straight-line drawings of 2-trees. Motivated by this negative result, we consider a local measure of edge-length ratio and prove that when the ratio is restricted only to the adjacent edges, any series-parallel graph admits a planar straight-line drawing with local edge-length ratio at most $4+\varepsilon$, for any arbitrarily small $\varepsilon > 0$. The proof of this upper bound is constructive, and it gives rise to a linear-time algorithm assuming a real RAM model of computation. (The omitted proofs can be found in [2].)

It is worth noticing that Borrazzo and Frati recently showed that any 2-tree on n vertices can be drawn with edge-length ratio $O(n^{0.695})$ [3]. This, together with our $\Omega(\log n)$ result, defines a non-trivial gap between upper and lower bound on the edge-length ratio of planar straight-line drawings of partial 2-trees.

2 Preliminaries

We consider finite nonempty planar graphs and their planar straight-line drawings. Once a straight-line drawing of G is given, with a slight abuse of notation we use the same symbol for a vertex U and the point U representing the vertex U in the drawing, as well as for an edge UV and the corresponding segment UV of the drawing. For points U and V, let $|UV|$ denote the Euclidean distance between U and V. For three mutually adjacent vertices U, V and W of a graph G, the symbol $\triangle UVW$ denotes the triangle of the corresponding drawing of G. For a polygon Q, we denote its perimeter by $P(Q)$ and its area by $A(Q)$.

The *edge-length ratio* of a planar straight-line drawing of a graph G is the ratio between the length of the longest and the shortest edge of the drawing.

▶ **Definition 2.1.** The *edge-length ratio* $\rho(G)$ of a planar graph G is the infimum edge-length ratio taken over all planar straight-line drawings of G.

A vertex is called simplicial when its neighborhood forms a clique. A complete graph on $k+1$ vertices is a k-tree; a graph constructed from a k-tree by adding a simplicial vertex to a clique of size k is also a k-tree. A partial k-tree is a subgraph of a k-tree.

3 Edge-length ratio of 2-trees

We recall that 2-trees are planar graphs. The main result of this section is the following.

▶ **Theorem 3.1.** *For any $r \geq 1$, there exists a 2-tree G whose edge-length ratio $\rho(G) \geq r$.*

To prove Theorem 3.1, for a given r we argue that a sufficiently large 2-tree, drawn with the longest edge having length r, contains a triangle with area at most $\frac{1}{2}$ (Lemma 3.2). Then, inside this triangle of small area we build a sequence of triangles with perimeters decreasing by $\frac{1}{2}$ in each step (Lemmas 3.8 and 3.9), which results in a triangle with an edge of length less than 1.

We consider a special subclass $\mathcal{G} = \{G_0, G_1, \dots\}$ of 2-trees with labeled vertices and edges constructed as follows: G_0 is the complete graph K_3 whose vertices and edges are given the label 0. The graph G_{i+1} is obtained by adding five simplicial vertices to each edge of label i of G_i. Each newly created vertex and edge gets label $i+1$. See Fig. 1 for an example where the black vertices and edges have label 0, the blue ones have label 1, and the red ones have label 2.

A *separating triangle of level i* in a straight-line drawing of a 2-tree G is an unordered triple $\{U, V, W\}$ of its mutually adjacent vertices such that the vertex W of label i was added

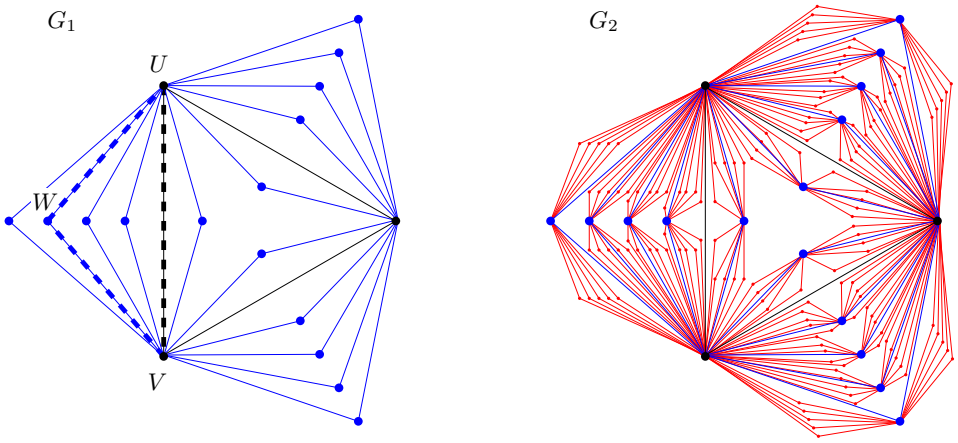


Figure 1 The 2-trees G_1 and G_2. Black color corresponds to label 0, blue to 1, and red to 2. Separating triangle Δ_1 is emphasized by a dashed line in G_1.

as a simplicial vertex to the edge UV in the recursive construction of G and the triangle $\triangle UVW$ splits the plane into two regions, each containing at least two other vertices with label i which are simplicial to the edge UV. In particular, the triangle $\triangle UVW$ contains two vertices of G with label i in its interior. For example, in Fig. 1 (a) vertices $\{U, V, W\}$ form a separating triangle.

▶ **Lemma 3.2.** *For any $k > i \geq 1$, any planar straight-line drawing of the graph G_k and an edge e of G_k labeled by i there exists a separating triangle of level $i + 1$ containing the endpoints of e.*

We proceed to show that any drawing of G_k contains a triangle of sufficiently small area. To this aim, we construct a sequence of nested triangles such that each element's area is half of the previous element's area. We denote as Δ_i a separating triangle in an embedding of G_k such that all its edges have labels at most i, with $i \leq k - 1$.

▶ **Lemma 3.3.** *For any $k \geq 1$, any planar straight-line drawing of G_k contains a sequence of triangles $\Delta_1, \Delta_2, \ldots, \Delta_k$, where for any $i \in \{1, \ldots, k\}$ the triangle Δ_i is a separating triangle of level i, and for each $i > 1$, in addition, Δ_i is in the interior of Δ_{i-1} and $A(\Delta_i) \leq \frac{1}{2}A(\Delta_{i-1})$.*

▶ **Corollary 3.4.** *For any given $r > 1$, there is a k such that every planar straight-line drawing of G_k with edge lengths at most r contains a separating triangle of area at most $\frac{1}{4}$.*

We call *thin* any triangle with edges of length at least 1 and area at most $\frac{1}{4}$. Any thin triangle has height at most $\frac{1}{2}$ and hence one obtuse angle of size at least $\frac{2\pi}{3}$ and two acute angles, each of size at most $\frac{\pi}{6}$.

▶ **Lemma 3.5.** *Let $\triangle UVW$ be a thin triangle, where the longest edge is UV and let $Z \in \triangle UVW$ be such that $|ZW| \geq 1$. Then one of the angles $\angle UWZ$ or $\angle VWZ$ is obtuse.*

Now we focus our attention on the perimeters of the considered triangles.

▶ **Observation 3.6.** Let a triangle $\triangle UVW$ be placed in the interior of a polygon Q. Then the perimeter of the triangle is bounded by the perimeter of the polygon, i.e., $P(\triangle UVW) \leq P(Q)$.

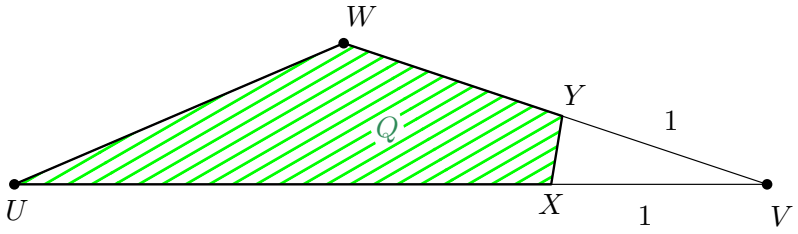


Figure 2 Cutting-off the triangle $\triangle XVY$.

▶ **Lemma 3.7.** *Let $\triangle UVW$ be a thin triangle, where the longest edge is UV. Then the polygon Q, created by cutting off an isosceles triangle $\triangle XVY$ with both edges XV and VY of length 1, has perimeter $P(Q) \leq P(\triangle UVW) - 1$.*

See Fig. 2 for an example of cutting off. We now show that a separating triangle with a small area is guaranteed to contain a separating triangle of a significantly smaller perimeter.

▶ **Lemma 3.8.** *Let G_k have a planar straight-line drawing with edge length at least 1 and let $\triangle UVW$ be a thin separating triangle of level $i \leq k-1$. Assume that the edge UV is of level $i-1$ and that it is incident to the obtuse angle of $\triangle UVW$. Then $\triangle UVW$ contains a separating thin triangle Q of level $i+1$ whose perimeter satisfies $P(Q) \leq P(\triangle UVW) - 1$.*

▶ **Lemma 3.9.** *Let G_k have a planar straight-line drawing with edge length at least 1 and let $\triangle UVW$ be a thin separating triangle of level $i \leq k-2$. Assume that the edge UV is of level $i-1$ and that it is* not incident *to the obtuse angle of $\triangle UVW$. Then $\triangle UVW$ contains a separating thin triangle Q of level at most $i+2$ whose perimeter satisfies $P(Q) \leq P(\triangle UVW) - 1$.*

Now we combine all lemmas together to complete the proof of Theorem 3.1.

Proof of Theorem 3.1. For given r we choose $k = \log_2\left(\frac{\sqrt{3}}{4}r^2\right) + 3$ and consider the graph G_{k+4r}. Assume for a contradiction that G_{k+4r} allows a drawing of edge-length ratio at most r. Without loss of generality assume that the longest edge of such drawing has length r and hence the shortest has length at least 1. In the drawing of the graph G_{k+4r} consider a sequence of separating triangles $\Delta_1, \ldots, \Delta_{k+4r}$ where $\Delta_1, \ldots, \Delta_k$ are chosen as shown in Lemma 3.3.

By Corollary 3.4, the triangle Δ_k is thin. Observe that the side-length and area constraints imply that it could be drawn inside a rectangle $r \times \frac{1}{8}$, hence it has perimeter at most $2r + \frac{1}{4}$ by Observation 3.6.

For any $i \in \{0, 1, \ldots, 2r-1\}$ either the edge of level $k+2i-1$ in Δ_{k+2i} is incident to the obtuse angle of Δ_{k+2i} or not:

- In the first case we apply Lemma 3.8 to get $P(\Delta_{k+2i+1}) \leq P(\Delta_{k+2i}) - 1$. As Δ_{k+2i+2} is inside Δ_{k+2i+1}, we get $P(\Delta_{k+2i+2}) \leq P(\Delta_{k+2i}) - 1$.
- Otherwise we apply Lemma 3.9 to derive $P(\Delta_{k+2i+2}) \leq P(\Delta_{k+2i}) - 1$ directly.

Therefore, $P(\Delta_{k+4r}) \leq P(\Delta_k) - 2r \leq 2r + \frac{1}{4} - 2r = \frac{1}{4}$, a contradiction to the assumption that all triangles of G_{k+4r} have all sides of length at least one. ◀

Note that the graph G_{k+4r} has $O^*((5^4)^r)$ vertices. The dependency between the edge-length ratio and the number of vertices could be rephrased as follows:

▶ **Corollary 3.10.** *The edge-length ratio over the class of n-vertex 2-trees is $\Omega(\log n)$.*

We recall that Borrazzo and Frati prove that every partial 2-tree with n vertices admits a planar straight-line drawing whose edge-length ratio is in $O(n^{0.695})$ (Corollary 1 of [4]).

4 Local edge-length ratio of 2-trees

The aesthetic criterion studied in the previous section took into account any pair of edges. By our construction of nested triangles, it might happen that two edges attaining the maximum length ratio might be far in the graph distance (in the Euclidean distance they are close as the triangles are nested). This observation leads us to the question, whether the two edges might be forced to appear close or whether 2-trees allow drawings where the length ratio of any two adjacent edges could be bounded by a constant. For this purpose we define the local variant of the edge-length ratio as follows:

The *local edge-length ratio* of a planar straight-line drawing of a graph G is the maximum ratio between the lengths of two adjacent edges (sharing a common vertex) of the drawing.

▶ **Definition 4.1.** The *local edge-length ratio* $\rho_l(G)$ of a planar graph G is the infimum local edge-length ratio taken over all planar straight-line drawings of G.

$$\rho_l(G) = \inf_{\text{drawing of } G} \max_{UV,VW \in E_G} \frac{|UV|}{|VW|}$$

Observe that the local edge-length ratio $\rho_l(G)$ is by definition bounded by the global edge-length ratio $\rho(G)$. In particular, every outerplanar graph G allows a drawing witnessing $\rho_l(G) \leq 2$ [11]. We extend this positive result to a class of all 2-trees with a slightly increased bound on the ratio.

▶ **Theorem 4.2.** *The local edge-length ratio of any n-vertex 2-tree G is $\rho_l(G) \leq 4$. Also, for any arbitrarily small positive constant ε, a planar straight-line drawing of G with local edge-length ratio at most $4+\varepsilon$ can be computed in $O(n)$ time assuming the real RAM model of computation.*

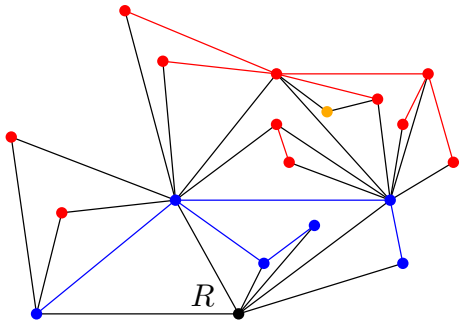


Figure 3 Decomposition rooted in vertex R of a 2-tree. Each connected component of colored vertices forms a part of the decomposition.

The proof of Theorem 4.2 is based on a construction that for a given 2-tree G and any $\varepsilon > 0$ provides a straight-line drawing of local edge-length ratio $4 + \varepsilon$. The general idea of the construction is as follows: We cover G by subgraphs, called parts, so that these parts could be arranged into a rooted tree (see Fig. 3). For any vertex u the parts containing u form a subtree of height one, where the root of this subtree has one edge in common with each of its children and such edge is always incident to u. For each part, we reserve

a suitable area where this part can be drawn and then describe how to draw it there with the local edge-length ratio at most $2 + \frac{\varepsilon}{2}$. For any two adjacent edges, either they belong to the same part or to two parts that have a parent-child or sibling relationship in the tree. By this reasoning we can prove that the local edge-length ratio is at most 4.

5 Open Problems

1. Corollary 3.10 of this paper gives a logarithmic lower bound while Corollary 1 of [4] gives a sub-linear upper bound on the edge-length ratio of planar straight-line drawings of partial 2-trees. We find it interesting to close the gap between upper and lower bound.
2. Theorem 4.2 gives an upper bound of 4 on the local edge-length ratio of partial 2-trees. It would be interesting to establish whether such an upper bound is tight.
3. The construction in Theorem 4.2 creates drawings where the majority of angles are very close to 0 or π radians. Hence, it would make sense to study the interplay between (local or global) edge-length ratio and angular resolution in planar straight-line drawings.

References

1 Sandeep N. Bhatt and Stavros S. Cosmadakis. The complexity of minimizing wire lengths in VLSI layouts. *Inf. Process. Lett.*, 25(4):263–267, 1987.

2 Václav Blažej, Jiří Fiala, and Giuseppe Liotta. On the edge-length ratio of 2-trees. *CoRR*, abs/1909.11152, 2019.

3 Manuel Borrazzo and Fabrizio Frati. On the edge-length ratio of planar graphs. *CoRR*, abs/1908.03586, 2019.

4 Manuel Borrazzo and Fabrizio Frati. On the edge-length ratio of planar graphs. In Daniel Archambault and Csaba D. Tóth, editors, *Graph Drawing and Network Visualization - 27th International Symposium, GD 2019, Prague, Czech Republic, September 17-20, 2019, Proceedings*, volume 11904 of *Lecture Notes in Computer Science*, pages 165–178. Springer, 2019.

5 Sergio Cabello, Erik D. Demaine, and Günter Rote. Planar embeddings of graphs with specified edge lengths. *J. Graph Algorithms Appl.*, 11(1):259–276, 2007.

6 Jianer Chen, Anxiao Jiang, Iyad A. Kanj, Ge Xia, and Fenghui Zhang. Separability and topology control of quasi unit disk graphs. *Wireless Networks*, 17(1):53–67, 2011.

7 Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice-Hall, 1999.

8 Peter Eades and Nicholas C. Wormald. Fixed edge-length graph drawing is NP-hard. *Discrete Applied Mathematics*, 28(2):111–134, 1990.

9 Emilio Di Giacomo, Giuseppe Liotta, and Roberto Tamassia. Graph drawing. In Jacob E. Goodman, Joseph O'Rourke, and Csaba D. Tóth, editors, *Handbook of Discrete and Computational Geometry*, pages 247–284. CRC Press LLC, 2017.

10 Michael Hoffmann, Marc J. van Kreveld, Vincent Kusters, and Günter Rote. Quality ratios of measures for graph drawing styles. In *Proceedings of the 26th Canadian Conference on Computational Geometry, CCCG 2014, Halifax, Nova Scotia, Canada, 2014*. Carleton University, Ottawa, Canada, 2014.

11 Sylvain Lazard, William J. Lenhart, and Giuseppe Liotta. On the edge-length ratio of outerplanar graphs. *Theor. Comput. Sci.*, 770:88–94, 2019.

12 Takao Nishizeki and Md. Saidur Rahman. *Planar Graph Drawing*, volume 12 of *Lecture Notes Series on Computing*. World Scientific, 2004.

13 Roberto Tamassia, editor. *Handbook on Graph Drawing and Visualization*. Chapman and Hall/CRC, 2013.

Simple Drawings of $K_{m,n}$ Contain Shooting Stars*

Oswin Aichholzer[1], Alfredo García[2], Irene Parada[1], Birgit Vogtenhuber[1], and Alexandra Weinberger[1]

1 **Graz University of Technology, Austria**
[oaich|iparada|bvogt|weinberger]@ist.tugraz.at
2 **Departamento de Métodos Estadísticos and IUMA, Universidad de Zaragoza.**
olaverri@unizar.es

Abstract

Simple drawings are drawings of graphs in which all edges have at most one common point (either a common endpoint, or a proper crossing). It has been an open question whether every simple drawing of a complete bipartite graph $K_{m,n}$ contains a plane spanning tree as a subdrawing. We answer this question to the positive by showing that for every simple drawing of $K_{m,n}$ and for every vertex v in that drawing, the drawing contains a *shooting star* rooted at v, that is, a plane spanning tree with all incident edges of v.

1 Introduction

A *simple drawing* is a drawing of a graph on the sphere S^2 or, equivalently, in the Euclidean plane where (1) the vertices are distinct points in the plane, (2) the edges are non-self-intersecting continuous curves connecting their incident points, (3) no edge passes through vertices other than its incident vertices, (4) and every pair of edges intersects at most once, either in a common endpoint, or in the relative interior of both edges, forming a proper crossing. Simple drawings are also called *good drawings* [3, 5] or *(simple) topological graphs* [7, 8]. In *semi-simple drawings*, the last requirement is softened such that edges without common endpoints are allowed to cross several times. Note that in any simple or semi-simple drawing, there are no tangencies between edges and incident edges do not cross. If a drawing does not contain any crossing at all, it is called *plane*.

The search for plane subdrawings of a given drawing has been a widely considered topic for simple drawings of the complete graph K_n which still holds tantalizing open problems. For example, Rafla [10] conjectured that every simple drawing of K_n contains a plane Hamiltonian cycle, a statement which is by now known to be true for $n \leq 9$ [1] and several classes of simple drawings (e.g., 2-page book drawings, monotone drawings, cylindrical drawings), but still remains open in general. A related question concerns the least number of pairwise disjoint edges in any simple drawing of K_n. The currently best lower bound of $\Omega(n^{1/2-\varepsilon})$, for any $\varepsilon > 0$, for this number has been obtained by Ruiz-Vargas [11], improving over several previous bounds [8, 9, 12], while the trivial upper bound of $n/2$ would be implied by a positive answer to Rafla's conjecture. A structural result of Fulek and Ruiz-Vargas [6] implies that every simple drawing of K_n contains a plane sub-drawing with at least $2n - 3$ edges.

* O.A., I.P., and A.W. partially supported by the Austrian Science Fund (FWF) grant W1230. A.G. supported by MINECO project MTM2015-63791-R and Gobierno de Aragón under Grant E41-17 (FEDER). I.P. and B.V. partially supported by the Austrian Science Fund within the collaborative DACH project *Arrangements and Drawings* as FWF project I 3340-N35. This work has been initiated at the 6th Austrian-Japanese-Mexican-Spanish Workshop on Discrete Geometry which took place in June 2019 near Strobl, Austria. We thank all the participants for the great atmosphere and fruitful discussions.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

In this work, we consider the search for plane spanning trees in drawings of complete bipartite graphs. For complete graphs, plane spanning trees trivially exist in both simple and semi-simple drawings, as incident edges don't cross, and as every vertex is incident to all other vertices. Hence the *star* of any vertex, which consists of all edges incident to that vertex and all vertices, is a plane spanning tree.

The task of finding plane spanning trees in drawings of complete bipartite graphs $K_{m,n}$ turns out to be more involved. In fact, not every semi-simple drawing of a complete bipartite graph contains a plane spanning tree; see Figure 1.

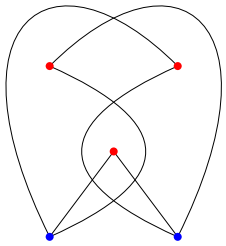


Figure 1 A semi-simple drawing of $K_{2,3}$ that does not contain a plane spanning tree.

For simple drawings of $K_{m,n}$, the existence of plane spanning trees has so far only been proven for specific types of drawings. It is not too hard to see that monotone drawings always contain plane spanning trees. Aichholzer et al. showed in [2] that simple drawings of $K_{2,n}$ and $K_{3,n}$, as well as so-called outer drawings of $K_{m,n}$ [4], always contain plane spanning trees of a special type, which they called shooting stars. A *shooting star rooted at* v is a plane spanning tree with root v that has height 2 and contains the star of vertex v.

In the present work, we show that every simple drawing of $K_{m,n}$ contains shooting stars rooted at an arbitrary vertex of $K_{m,n}$.

▶ **Theorem 1.1.** *Let D be a simple drawing of the complete bipartite graph $K_{m,n}$ and let r be an arbitrary vertex of $K_{m,n}$. Then D contains a shooting star rooted at r.*

2 Proof of Theorem 1.1

Proof. We can assume that D is drawn on a point set $S = R \cup B$, $R = \{r_1, \ldots, r_m\}$, $B = \{b_1, \ldots, b_n\}$, in which the points in the two vertex partitions R and B are colored red and blue, respectively. Without loss of generality let $r = r_1$.

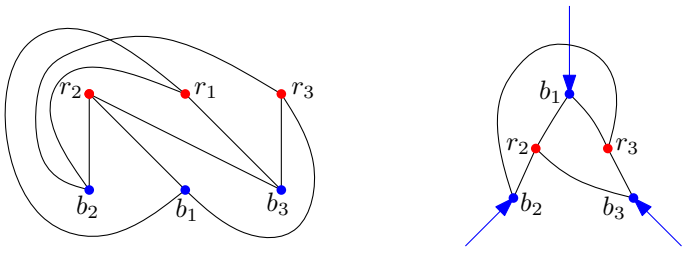


Figure 2 A simple drawing of $K_{3,3}$ (left) and its stereographic projection from r_1 (right).

To simplify the figures, we consider the drawing D on the sphere and apply a stereographic projection from r onto a plane. In that way, the edges in the star of r are represented as (not necessarily straight-line) infinite rays; see Figure 2. We will depict them in blue. In the

following, we consider all edges oriented from their red to their blue endpoint. In order to specify how two edges cross each other, we introduce some notation. Consider two crossing edges $e_1 = r_i b_k$ and $e_2 = r_j b_l$ and let $\times$ be their crossing point. Consider the arcs $\times r_i$ and $\times b_k$ on e_1 and $\times r_j$ and $\times b_l$ on e_2. We say that e_2 crosses e_1 in *clockwise direction* if the clockwise cyclic order of these arcs around the crossing $\times$ is $\times r_i$, $\times r_j$, $\times b_k$, and $\times b_l$. Otherwise, we say that e_2 crosses e_1 in *counterclockwise direction*; see Figure 3.

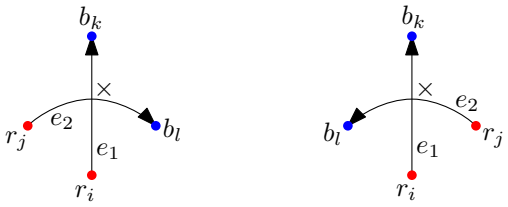


Figure 3 Edge e_2 crosses edge e_1 in clockwise direction (left) or counterclockwise direction (right).

We prove Theorem 1.1 by induction on n. For $n = 1$ and any $m \geq 1$, the whole drawing D is a shooting star rooted at any vertex, and in particular at r.

Assume that the existence of shooting stars rooted at any vertex has been proven for any simple drawing of $K_{m,n'}$ with $n' < n$. Let M be a subset of the edges of D connecting each vertex $r_i \neq r$ to some blue vertex in B (with exactly one edge for every red vertex), such that (i) $M \cup \{\bigcup_{j=2}^{n} rb_j\}$ does not contain any crossing and (ii) the number of crossings of M with edge rb_1 is the minimum possible. Observe that the set M is well defined, since, by the induction hypothesis, the subdrawing of D obtained by deleting the blue vertex b_1 and its incident edges contains a shooting star rooted at r. Thus there exists a set M_1 of edges from D connecting each $r_i \neq r$ to some blue vertex in $B \setminus \{b_1\}$ such that $M_1 \cup \{\bigcup_{j=2}^{n} rb_j\}$ does not contain any crossing. As arbitrarily many of the edges in M_1 might cross rb_1, M_1 might not fulfill condition (ii). We will show that M does not contain any crossing with rb_1. Since $M \cup \{\bigcup_{j=2}^{n} rb_j\}$ does not contain crossings by construction and as rb_1 does not cross any edges of $\{\bigcup_{j=2}^{n} rb_j\}$ in a simple drawing, it follows that the edges in $M \cup \{\bigcup_{j=1}^{n} rb_j\}$ form the desired shooting star.

Assume for a contradiction that rb_1 crosses at least one edge in M. When traversing rb_1 from b_1 to r, let x be the first crossing point of rb_1 with an edge $r_k b_t$ in M. Without loss of generality, when orienting rb_1 from r to b_1 and $r_k b_t$ from r_k to b_t, $r_k b_t$ crosses rb_1 in counterclockwise direction (otherwise we can mirror the drawing).

Suppose first that the arc $r_k x$ (on $r_k b_t$ and oriented from r_k to x) is crossed in counterclockwise direction by an edge incident to b_1 (and oriented from the red endpoint to b_1). Let $e = r_l b_1$ be such an edge whose crossing with $r_k x$ at a point y is the closest to x. Otherwise, let e be the edge $r_k b_1$ and y be the point r_k. In the remaining figures, we represent in blue the edges of the star of r, in red the edges in M, and in black the edge e.

We distinguish two cases depending on whether e crosses an edge of the star of r. The idea in both cases is to define a region Γ and, inside it, redefine the connections between red and blue points to reach a contradiction.

Case 1: e does not cross any edge of the star of r. Let Γ be the closed region of the plane bounded by the arcs yb_1 (on e), $b_1 x$ (on rb_1), and xy (on $r_k b_t$); see Figure 4. Observe that all the blue points b_j lie outside the region Γ and that for all the red points r_i inside region Γ, the edge $r_i b_1$ must be in Γ. Let M_Γ denote the set of edges $r_i b_1$ with $r_i \in \Gamma$ and note that $r_k b_1 \in M_\Gamma$. Consider the set M' of red edges obtained from M by replacing, for each red point $r_i \in \Gamma$, the (unique) edge incident to r_i in M by the edge $r_i b_1$ in M_Γ, and keeping the

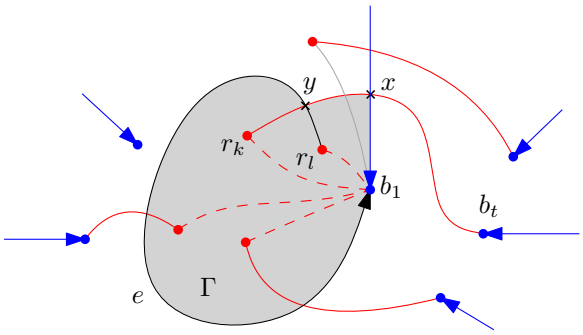


Figure 4 Illustration of Case 1.

other edges in M unchanged. In particular, the edge $r_k b_t$ has been replaced by the edge $r_k b_1$. The edges in M_Γ neither cross each other nor cross any of the blue edges rb_j. Moreover, we now show that the non-replaced edges in M must lie completely outside Γ. These edges can neither cross $r_k b_t$ (by definition of M) nor the arc $b_1 x$ (on rb_1). Thus, if they are incident to b_1 they cannot cross the boundary of Γ, and otherwise their endpoints lie outside Γ and they can only cross one arc of the boundary. Therefore, M' satisfies that $M' \cup \{\bigcup_{j=2}^{n} rb_j\}$ does not contain any crossing, and has fewer crossings with rb_1 than M. This contradicts the definition of M as the one with the minimum number of crossings with rb_1.

Case 2: e crosses the star of r. When traversing e from r_k or r_l (depending on the definition of e) to b_1, let $I = \{\alpha, \beta, \ldots, \rho\}$ be the indices of the edges of the star of r in the order as they are crossed by e and let $y_\alpha, \ldots, y_\rho$ be the corresponding crossing points on e. Note that, when orienting e from r_k or r_l to b_1, the edges $rb_\xi, \xi \in I$, oriented from r to b_ξ, cross e in counterclockwise direction, since they can neither cross $r_k b_t$ (by definition of M) nor rb_1.

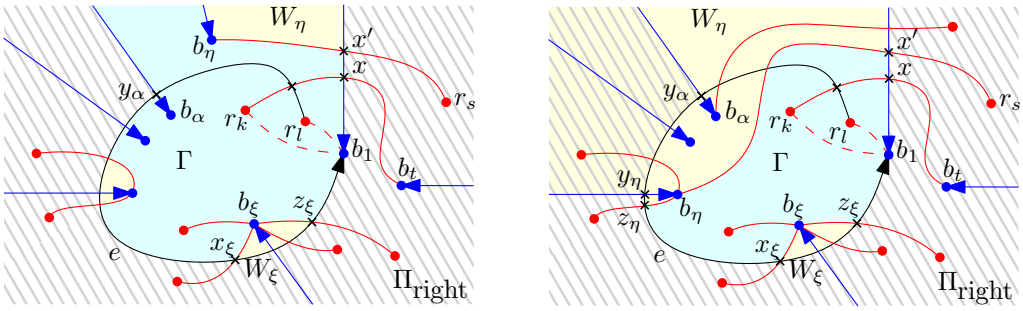


Figure 5 Illustration of Case 2. Region Π_{right} is striped in gray, region Γ is shaded in blue, and regions in $\bigcup_{\xi \in I} W_\xi \cup W_\eta$ are shaded in yellow. Left: $\eta \notin I$. Right: $\eta \in I$.

The three arcs ry_α (on rb_α), $y_\alpha b_1$ (on e), and $b_1 r$ divide the plane into two (closed) regions, Π_{left}, containing vertex r_k, and Π_{right}, containing vertex b_t. For each $\xi \in I$, let M_ξ be the set of red edges of M incident to some red point in Π_{right} and to b_ξ. Note that all the edges in M_ξ (if any) must cross the edge e. When traversing e from r_k or r_l to b_1, we denote by x_ξ, z_ξ the first and the last crossing points of e with the edges of $M_\xi \cup rb_\xi$, respectively; see Figure 5 for an illustration. We remark that both x_ξ and z_ξ might coincide with y_ξ and, in particular, if $M_\xi = \emptyset$ then $x_\xi = y_\xi = z_\xi$.

We now define some regions in the drawing D. Suppose first that there are edges in M (oriented from the red to the blue point) that cross rb_1 (oriented from r to b_1) in clockwise direction. Let $r_s b_\eta$ be the edge in M whose clockwise crossing with rb_1 at a point x' is the

closest one to x (recall that the arc b_1x on rb_1 is not crossed by edges in M). Then, if $\eta \notin I$, we denote by W_η the region bounded by the arcs rx' (on rb_1), $x'b_\eta$ (on r_sb_η), and rb_η and not containing b_1; see Figure 5 (left). If $\eta \in I$, we define W_η as the region bounded by the arcs rx' (on rb_1), $x'b_\eta$ (on r_sb_η), $b_\eta z_\eta$, $z_\eta y_\eta$ (on e), and $y_\eta r$ (on rb_η) and not containing b_1; see Figure 5 (right). If no edges in M cross rb_1 in clockwise direction, we set $W_\eta = \emptyset$. Moreover, for each $\xi \in I \setminus \{\eta\}$, we define W_ξ as the region bounded by the arcs $x_\xi b_\xi$, $b_\xi z_\xi$, and $z_\xi x_\xi$ (and not containing b_1); see again Figure 5.

We can finally define the region Γ for Case 2, which is the region obtained from Π_{left} by removing the interior of all the regions W_ξ, $\xi \in I$ plus region W_η if $\eta \notin I$ (otherwise it is already contained in $\bigcup_{\xi\in I} W_\xi$). Now consider the set of red and blue vertices contained in the region Γ. Let J denote the set of indices such that for all $j \in J$, the blue point b_j lies in Γ (note that $1 \in J$). Since b_t is not in Γ, by the induction hypothesis, we can find a set of edges M_Γ connecting each red point in Γ with a blue point b_j, $j \in J$ satisfying that $M_\Gamma \cup \{\bigcup_{j\in J} rb_j\}$ does not contain any crossing. Moreover, all the edges in M_Γ lie entirely in Γ: An edge in M_Γ cannot cross any of the edges rb_j, with $j \in J$. Thus, it cannot leave Π_{left}, as otherwise it would cross e twice. Further, if it entered one of the regions in $\bigcup_{\xi\in I} W_\xi \cup W_\eta$, it would have to leave it crossing e, and then it could not reenter Γ.

Consider the set M' of red edges obtained from M by replacing, for each red point $r_i \in \Gamma$, the edge r_ib_ξ in M by the edge r_ib_j, $j \in J$, in M_Γ, and keeping the other edges in M unchanged. In particular, the edge r_kb_t has been replaced by some edge r_kb_j, $j \in J$. The edges in M_Γ neither cross each other nor cross any of the blue edges rb_j, $j \in J$ nor any of the other ones, lying completely outside Γ. Moreover, the non-replaced edges in M cannot enter Γ since the only boundary part of Γ that they can cross are arcs on e. Therefore, M' satisfies that $M' \cup \{\bigcup_{j=2}^{n} rb_j\}$ does not contain any crossing, and has fewer crossings with rb_1 than M. This contradicts the definition of M as the one with the minimum number of crossings with rb_1. ◀

3 Some Observations on Tightness

There exist simple drawings of $K_{m,n}$ in which every plane subdrawing has at most as many edges as a shooting star. For example, consider a straight line drawing of $K_{m,n}$ where all vertices are in convex position such that all red points are next to each other in the convex hull; see Figure 6 (left). The convex hull is a $(m+n)$-gon which shares only two edges with the drawing of $K_{m,n}$; see Figure 6 (right). All other edges of the drawing of $K_{m,n}$ are diagonals of the polygon. As there can be at most $(m+n)-3$ pairwise non-crossing diagonals in a convex $(m+n)$-gon, any plane subdrawing of this drawing of $K_{m,n}$ contains at most $m+n-1$ edges.

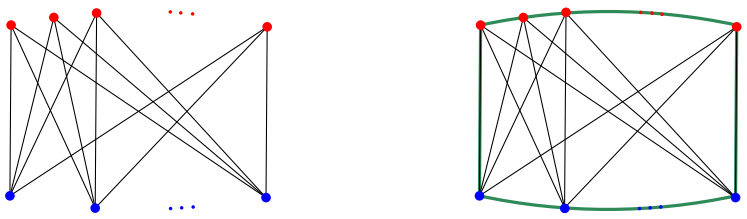


Figure 6 Left: A simple drawing of $K_{m,n}$ where no plane subdrawing can have more edges than a shooting star. Right: A convex $(n+m)$-gon (in green lines) around the simple drawing of $K_{m,n}$.

Furthermore, both requirements from Theorem 1.1 – simplicity of the drawing and having

a complete bipartite graph – are in fact necessary: As mentioned in the introduction, not all semi-simple drawings of $K_{m,n}$ contain a plane spanning tree. Further, if in the example in Figure 6, we delete one of the two edges of $K_{m,n}$ on the boundary of the convex hull, then any plane subdrawing has at most $m + n - 2$ edges. Hence the resulting drawing cannot contain any plane spanning tree.

4 Remarks on an Algorithm

The proof of Theorem 1.1 contains an algorithm with which we can find shooting stars in given simple drawings. We start with constructing the shooting star for a subdrawing that is a $K_{m,1}$ and then inductively add more vertices. Every time we are adding a new vertex, the shooting star of the step before is a set fulfilling all requirements of $M_1 \cup \{\bigcup_{j=2}^{n} rb_j\}$ in the proof. By replacing edges as described in the proof, we obtain a new set with the same properties and fewer crossings. We continue replacing edges until we obtain a set of edges (M in the proof) that form a shooting star for the extended vertex set. We remark that the runtime of this algorithm might be exponential, as finding the edges of M_Γ might require solving the problem for the subgraph induced by Γ. However, we believe that there exists a polynomial-time algorithm for this task.

▶ Open Problem 1. Given a simple drawing of the complete bipartite graph, is there a polynomial-time algorithm to find a plane spanning tree contained in the drawing?

References

1 B.M. Ábrego, O. Aichholzer, S. Fernández-Merchant, T. Hackl, J. Pammer, A. Pilz, P. Ramos, G. Salazar, and B. Vogtenhuber. All good drawings of small complete graphs. In *Proc. 31st European Workshop on Computational Geometry EuroCG '15*, pages 57–60, Ljubljana, Slovenia, 2015.

2 Oswin Aichholzer, Irene Parada, Manfred Scheucher, Birgit Vogtenhuber, and Alexandra Weinberger. Shooting stars in simple drawings of $K_{m,n}$. In *Proceedings of the 34th European Workshop on Computational Geometry (EuroCG'19)*, pages 59:1–59:6, 2019. URL: `http://www.eurocg2019.uu.nl/papers/59.pdf`.

3 Alan Arroyo, Dan McQuillan, R. Bruce Richter, and Gelasio Salazar. Levi's lemma, pseudolinear drawings of K_n, and empty triangles. *Journal of Graph Theory*, 87(4):443–459, 2018. `doi:10.1002/jgt.22167`.

4 Jean Cardinal and Stefan Felsner. Topological drawings of complete bipartite graphs. In *Proceedings of the 24th International Symposium on Graph Drawing and Network Visualization (GD'16)*, volume 9801 of *LNCS*, pages 441–453. Springer, 2016. `doi:10.1007/978-3-319-50106-2_34`.

5 Paul Erdős and Richard K. Guy. Crossing number problems. *The American Mathematical Monthly*, 80(1):52–58, 1973. `doi:10.2307/2319261`.

6 Radoslav Fulek and Andres J. Ruiz-Vargas. Topological graphs: empty triangles and disjoint matchings. In *Proceedings of the 29th Annual Symposium on Computational Geometry (SoCG'13)*, pages 259–266, New York, 2013. ACM. `doi:10.1145/2462356.2462394`.

7 Jan Kynčl. Enumeration of simple complete topological graphs. *European Journal of Combinatorics*, 30:1676–1685, 2009. `doi:10.1016/j.ejc.2009.03.005`.

8 János Pach, József Solymosi, and Géza Tóth. Unavoidable configurations in complete topological graphs. *Discrete & Computational Geometry*, 30(2):311–320, 2003. `doi:10.1007/s00454-003-0012-9`.

9 János Pach and Géza Tóth. Disjoint edges in topological graphs. In *Proceedings of the 2003 Indonesia-Japan Joint Conference on Combinatorial Geometry and Graph Theory (IJCCGGT'03)*, volume 3330 of *LNCS*, pages 133–140, Berlin, 2005. Springer. doi:10.1007/978-3-540-30540-8_15.

10 Nabil H. Rafla. *The good drawings D_n of the complete graph K_n*. PhD thesis, McGill University, Montreal, 1988. URL: http://digitool.library.mcgill.ca/thesisfile75756.pdf.

11 Andres J. Ruiz-Vargas. Many disjoint edges in topological graphs. In *Proceedings of the 8th Latin-American Algorithms, Graphs and Optimization Symposium (LAGOS'15)*, volume 50, pages 29–34, 2015. doi:10.1016/j.endm.2015.07.006.

12 Andrew Suk. Disjoint edges in complete topological graphs. *Discrete & Computational Geometry*, 49(2):280–286, 2013. doi:10.1007/s00454-012-9481-x.

On the Number of Delaunay Triangles occurring in all Contiguous Subsequences

Stefan Funke and Felix Weitbrecht

Universität Stuttgart
{funke,weitbrecht}@fmi.uni-stuttgart.de

Abstract

Given an ordered sequence of points $P = \{p_1, p_2, \dots p_n\}$, we are interested in the number of different Delaunay triangles occurring when considering the Delaunay triangulations of all contiguous subsequences within P. While clearly point sets and orderings with $\Theta(n^2)$ Delaunay triangles exist, we prove that for an arbitrary point set in random order, the expected number of Delaunay triangles is $\Theta(n \log n)$.

1 Introduction

Given an ordered sequence of points $P = \{p_1, p_2, \dots, p_n\}$, we consider for $1 \leq i < j \leq n$ the contiguous subsequences $P_{i,j} := \{p_i, p_{i+1}, \dots, p_j\}$ and the set of Delaunay triangles $T_{i,j}$ that appear in the Delaunay triangulation $DT(P_{i,j})$ of the respective subset. We are interested in the size of the set $T := \bigcup_{i<j} T_{i,j}$ of distinct Delaunay triangles occurring over all contiguous subsequences.

There are sequences of points where $|T| = \Theta(n^2)$, e.g., see Figure 1. Here, points are ordered as shown in the Figure (note that the collinearity can easily be perturbed away). For $j > \frac{n}{2}$, any point p_j will be connected to all points $\{p_1, ..., p_{\frac{n}{2}}\}$ in $T_{1,j}$, so any such $T_{1,j}$ contains $\Theta(n)$ Delaunay triangles which are not contained in any $T_{1,j'}$ with $j' < j$, hence $|T| = \Theta(n^2)$. Note, though, that for this argument we only used linearly many contiguous subsequences. It is conceivable that the quadratically many contiguous subsequences create even a superquadratic number of distinct Delaunay triangles. We will show, though, that for an arbitrary point set in *random order*, $E[|T|] = \Theta(n \log n)$, and $|T| = O(n^2)$ for any order.

Applications and Motivation

Subcomplexes of the Delaunay triangulation have proven to be very useful for representing the shape of objects from a discrete sample in many contexts, see for example α-shapes [3],

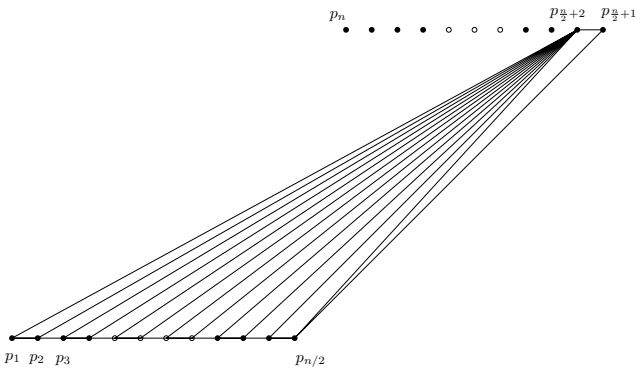


Figure 1 Example for a sequence of points with $|T| = \Theta(n^2)$, as in [4]

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

the β-skeleton [5], or the crust [1]. If the samples are acquired over time, a subcomplex of the Delaunay triangulation of the samples within a contiguous time interval might allow for interesting insights into the data, see for example [2], where the authors use α-shapes to visualize the regions of storm event data within the United States between 1991 and 2000.

While in real world application scenarios, samples do not occur in truly random order, it has also been observed in [2] that the potentially huge size of T seems more like a pathological setting. Our result provides some sort of explanation for this observation. It also suggests the possibility of precomputing all Delaunay triangles occurring in all contiguous subsequences and indexing them with respect to time and possibly some other parameter (e.g., the α value in case of α-shapes, or the β values in case of the β-skeleton) for faster retrieval.

2 Counting Delaunay Edges and Triangles

Our proof will proceed in two steps. We first show that the expected number of Delaunay edges created when considering all contiguous subsequences is $\Theta(n \log n)$ for a uniformly random ordering of an arbitrary point set. Then we show that, for an arbitrary order, there is a linear dependence between the number of created Delaunay triangles and Delaunay edges.

As usual, we assume non-degeneracy of P, i.e., absence of four co-circular or three co-linear points. Also let us define the set of edges used in triangles of T as:

$$E_T := \{e \mid \exists\, t \in T : e \text{ edge of } t\}$$

We consider an arbitrary point set ordered uniformly at random and bound the expected size of E_T, that is, the number of edges that appear in at least one of the Delaunay triangulations $DT(P_{i,j})$. The following Lemma is a simple observation that helps to focus on a smaller subsequence when considering a potential Delaunay edge $\{p_i, p_j\}$.

▶ **Lemma 2.1.** *Any edge $e = \{p_i, p_j\} \in E_T$ (w.l.o.g. $i < j$) appears in $DT(P_{i,j})$.*

Proof. There exists some triangle $t \in T$ which uses e, so for suitable $a \leq i, b \geq j$, e appears in $DT(P_{a,b})$, i.e., there exists a disk with p_i, p_j on its boundary and its interior free of points from $P_{a,b}$. As $P_{i,j} \subseteq P_{a,b}$ this disk is also free of points from $P_{i,j}$, hence $e \in DT(P_{i,j})$. ◀

Essentially, Lemma 2.1 states that we only need to consider the minimal contiguous subsequence containing p_i and p_j to argue about the probability of an edge $\{p_i, p_j\}$ being present in E_T. Let us now bound the probability that an edge $e = \{p_i, p_j\}$, with $j > i + 1$, appears as Delaunay edge in some $DT(P_{i,j})$.

▶ **Lemma 2.2.** *For a potential edge $e = \{p_i, p_j\}$, $j > i+1$, we have $Pr[e \in DT(P_{i,j})] < \frac{6}{j-i}$.*

Proof. Observe that when considering the point set $P_{i,j}$, clearly $DT(P_{i,j})$ will be the same regardless of how the points in $P_{i,j}$ are ordered. All points in $P_{i,j}$ are equally likely to be p_i, or p_j. $DT(P_{i,j})$ is a planar graph with $j - i + 1 > 2$ nodes, and hence per Euler's formula contains at most $3(j - i + 1) - 6$ edges. $Pr[e \in DT(P_{i,j})]$ is thus bounded by the probability of two randomly chosen nodes in a graph with $j - i + 1$ nodes and at most $3(j - i + 1) - 6$ edges to be connected with an edge. By randomly choosing two nodes, we randomly choose one edge amongst all possible $\binom{j-i+1}{2}$ edges. The probability of that edge to be one of the $\leq 3(j - i + 1) - 6$ edges of $DT(P_{i,j})$ is $< \frac{6}{j-i}$. ◀

As due to Lemma 2.1 we have that $Pr[e \in E_T] = Pr[e \in DT(P_{i,j})])$, we can continue to bound the expected size of E_T.

▶ **Lemma 2.3.** *The expected size of E_T is $\Theta(n \log n)$.*

Proof. For the lower bound of $\Omega(n \log n)$ consider first for point p_1 the nearest neighbor in $P_{2,i}$ as i grows from 2 to n. It is well known that for random order of P, the nearest neighbor changes $\Theta(\log n)$ times in expectation. It is also well-known that the nearest neighbor graph of a point set is a subgraph of its Delaunay triangulation. Hence p_1 in expectation is involved in the creation of $\Omega(\log n)$ distinct Delaunay edges. The same argument can be applied to all other points, hence we get a $\Omega(n \log n)$ lower bound.

By linearity of expectation we can simply sum over all potential $\binom{n}{2}$ edges to obtain the upper bound on the expected size of E_T. We split the set of potential edges between those with neighboring nodes in the ordering (like p_i and p_{i+1}) and the remaining ones. The former always exist, but there are only linearly many of them, for the latter we use Lemma 2.2 to bound the probability of existence.

$$\begin{aligned} E[|E_T|] &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} Pr[\{p_i, p_j\} \in E_T] \\ &= \sum_{i=1}^{n-1} \left[Pr[\{p_i, p_{i+1}\} \in E_T] + \sum_{j=i+2}^{n} Pr[\{p_i, p_j\} \in DT(P_{i,j})] \right] \\ &\leq \sum_{i=1}^{n-1} \left[1 + \sum_{j=i+2}^{n} \frac{6}{j-i} \right] = (n-1) + 6 \sum_{i=1}^{n-1} \sum_{j=2}^{n-i} \frac{1}{j} \\ &\leq (n-1) + 6 \sum_{i=1}^{n-1} H_n = O(n \log n) \end{aligned}$$

◀

Note that in general, many of these edges are used by several Delaunay triangles, some may even be used by $\Theta(n)$ different Delaunay triangles, so it is not immediately obvious that the overall number of Delaunay edges linearly bounds the overall number of Delaunay triangles. Yet, the following Lemma shows why this is the case.

▶ **Lemma 2.4.** $|T| \in \Theta(|E_T|)$.

Proof. Consider some Delaunay triangle $t = p_a p_b p_c \in T$ (w.l.o.g. $a < b < c$). Due to Lemma 2.1, we have $t \in T_{a,c}$. Apart from t, there can exist at most one other triangle $t' \in T_{a,c}$ which uses the edge $\{p_a, p_c\}$. This way, we can charge every Delaunay triangle of T to some Delaunay edge of E_T, charging at most 2 Delaunay triangles to any Delaunay edge. So the overall number of Delaunay triangles is at most twice the overall number of Delaunay edges, hence $|T| \in O(|E_T|)$. The lower bound is obvious. ◀

As a corollary, Lemma 2.4 implies that $O(n^2)$ is also an upper bound for $|T|$ for arbitrary orderings of n points as there are only $O(n^2)$ possible edges.

▶ **Corollary 2.5.** $|T| \in O(n^2)$ *for arbitrary point sets and orderings.*

Finally we can state our main theorem.

▶ **Theorem 2.6.** *The expected number of different Delaunay triangles occurring in all contiguous subsequences of a (uniformly) randomly ordered point set of size n is $\Theta(n \log n)$.*

Proof. Follows from Lemmas 2.3 and 2.4. ◀

Note that our analysis relies on two main properties, namely, uniqueness of the triangulation of a point set, and that edges cannot disappear when removing points (except their endpoints, of course). It might be interesting to investigate other triangulations fulfilling this property.

References

1 Nina Amenta, Marshall W. Bern, and David Eppstein. The crust and the beta-skeleton: Combinatorial curve reconstruction. *Graphical Models and Image Processing*, 60(2):125–135, 1998.

2 Annika Bonerath, Benjamin Niedermann, and Jan-Henrik Haunert. Retrieving α-shapes and schematic polygonal approximations for sets of points within queried temporal ranges. In *SIGSPATIAL/GIS*, pages 249–258. ACM, 2019.

3 Herbert Edelsbrunner, David G. Kirkpatrick, and Raimund Seidel. On the shape of a set of points in the plane. *IEEE Trans. Information Theory*, 29(4):551–558, 1983.

4 Leonidas J. Guibas, Donald E. Knuth, and Micha Sharir. Randomized incremental construction of delaunay and voronoi diagrams. *Algorithmica*, 7(1):381–413, Jun 1992.

5 David G. Kirkpatrick and John D. Radke. A framework for computational morphology. In Godfried T. Toussaint, editor, *Computational Geometry*, volume 2 of *Machine Intelligence and Pattern Recognition*, pages 217 – 248. North-Holland, 1985.

Empty Rainbow Triangles in k-colored Point Sets*

Ruy Fabila-Monroy[1], Daniel Perz[2], and Ana Laura Trujillo[1]

1 **Departamento de Matemáticas, CINVESTAV**
`ruyfabila@math.cinvestav.edu.mx, ltrujillo@math.cinvestav.mx`

2 **Institute for Software Technology, Graz University of Technology, Graz, Austria**
`daperz@ist.tugraz.at`

Abstract

Let S be a set of n points in general position in the plane. Suppose that each point of S has been assigned one of $k \geq 3$ possible colors and that there is the same number, m, of points of each color class, so $n = km$. A triangle with vertices on S is empty if it does not contain points of S in its interior and it is rainbow if all its vertices have different colors. Let $f(k, m)$ be the minimum number of empty rainbow triangles determined by S. In this paper we show that $f(k, m) = \Theta(k^3)$. Furthermore we give a construction which does not contain an empty rainbow quadrilateral.

1 Introduction

A set of points in the plane is in *general* position if no three of its vertices are collinear. In this paper all sets of points are in general position. The well known Erdős-Szekeres theorem [15] states that for every positive integer $r > 3$ there exists a positive integer $n(r)$ such that every set of $n(r)$ or more points in the plane contains the vertices of a convex polygon of r vertices.

Let S be a set n points in the plane. A polygon with vertices on S is said to be *empty* if it does not contain a point of S in its interior. An *r-hole* of S is an empty convex r-gon spanned by points of S. In 1978, Erdős [14] asked if for every r, every sufficiently large set of points in the plane contains an r-hole. Klein [15] had already noted that every set of 5 points contains a 4-hole. Harboth [19] showed that every set of 10 points contains a 5-hole. Horton [20] constructed an arbitrarily large set of points without a 7-hole. The case for 6-holes remained open until Nicolás [25] and Gerken [18], independently showed that every sufficiently large point set contains a 6-hole.

Once the existence of a given r-hole is established, it is natural to ask what is the minimum number of r-holes in every set of n points in the plane. Katchalski and Meir [22] first considered this question for triangles. They showed that every set of n points determines $\Omega(n^2)$ empty triangles and provided an example of a point set determining $O(n^2)$ empty triangles. The lower and upper bounds on this number have been improved throughout the years [6, 13, 27, 17, 11, 3, 1]. The problem of determining the minimum number of r-holes for $r = 4, 5$ or 6 in every set of n points in the plane has also been considered in these papers.

* R. F.-M. is partially supported by Conacyt of Mexico, Grant 253261. D. P. is partially supported by the Austrian Science Fund within the collaborative DACH project *Arrangements and Drawings* as FWF project I 3340-N35.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 734922.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Colored variants of these problems where first studied by Devillers, Hurtado, Károlyi and Seara [12]. In these variants each point of S is given a color from a prescribed set. A point set is *k-colored* if every one of its points is assigned one of k available colors. We say that an r-hole on S is *monochromatic* if all its vertices are of the same color and *rainbow* if all its vertices are of different colors. Many chromatic variants on problems regarding r-holes in points sets have been studied since; see [7, 26, 2, 4, 8, 24, 5, 21, 9, 16, 28, 23]. In particular, Aichholzer, Fabila-Monroy, Flores-Peñaloza, Hackl, Huemer, and Urrutia showed that every 2-colored set of n points determines $\Omega(n^{5/4})$ empty monochromatic triangles [2]. This was later improved to $\Omega(n^{4/3})$ by Pach and Toth [26]. The best upper bound on this number is $O(n^2)$ and this is conjectured to be the right asymptotic value. If we take three colors, then there exist 3-colored point sets without a monochromatic triangle [12].

In this paper we consider the problem of counting the number of empty rainbow triangles in k-colored point sets in which there are the same number, m, of points of each color class. Let $f(k, m)$ be the minimum number empty rainbow triangles in such a point set. We give the following asymptotic tight bound for $f(k, m)$.

▶ **Theorem 1.1.** *Let $m \geq k$ be positive integers then*

$$f(k, m) = \Theta(k^3).$$

The lower bound is shown in Section 2 and the upper bound in Section 3. We point out that in contrast to the number of empty monochromatic triangles, the number of rainbow empty triangles does not necessarily grow with the number of points. Further we show that for every $k \geq 4$ there are k-colored point sets without a rainbow 4-hole.

2 Lower Bound

▶ **Theorem 2.1.** *Let $m \geq k$ be positive integers then*

$$f(k, m) \geq \frac{1}{6}k^3 - \frac{1}{2}k^2 + \frac{1}{3}k.$$

Proof. Let S be a k-colored set of points such that there are the $m \geq k$ points of each color class. Without loss of generality assume that no two points of S have the same x-coordinate. Assume that the set of colors is $\{1, \ldots, k\}$. For each $1 \leq i \leq k$, let p_i be the leftmost point of color i. Without loss of generality assume that when sorted by x-coordinate these points are $p_k, p_{k-1} \ldots, p_1$.

We now show that for $i \geq 3$ there are at least $(i^2 - 3i + 2)/2$ empty rainbow triangles having a point of color i as its rightmost point. Let $q_1 := p_i, q_2, \ldots, q_{i-2}$ be the first $i - 2$ points of color i when sorted by x-coordinate. For each $1 \leq j \leq i - 2$ do the following. Sort the points of S to the left of q_j counterclockwise by angle around q_j. Note that any two consecutive points in this order define an empty triangle with q_j as its rightmost point. Since the points $p_1, \ldots, p_{i-1}$ are to the left of q_j, there at least $i-2$ of these empty triangles such that the first point is of a color l distinct from i, and the next point is of a color distinct from l and i. Furthermore, for at least $(i - 2) - (j - 1) = i - j - 1$ of these triangles the next point is not of color i; thus they are rainbow. We have least

$$\sum_{j=1}^{i-1} i - j - 1 = \frac{i^2 - 3i + 2}{2}$$

empty rainbow triangles with a point of color i as its rightmost point.

Thus, S determines at least

$$\sum_{i=1}^{k} \frac{i^2 - 3i + 2}{2} = \frac{1}{6}k^3 - \frac{1}{2}k^2 + \frac{1}{3}k$$

empty rainbow triangles. ◀

3 Upper Bound

In this section we construct a k-colored point set which gives us an upper bound for $f(k, m)$.

3.1 The Empty Triangles of the Horton Set

In this section we define the point set introduced by Horton [20] and characterize its empty triangles. Let H be a set of n points in the plane with no two points having the same x-coordinate; sort its points by their x-coordinate so that $H := \{p_0, p_1, \ldots, p_{n-1}\}$. Let H_0 be the subset of the even-indexed points, and H_1 be the subset of the odd-indexed points. That is, $H_0 = \{p_0, p_2, \ldots\}$ and $H_1 = \{p_1, p_3, \ldots\}$. Let X and Y be two sets of points in the plane. We say that X is *high above* Y if: every line determined by two points in X is above every point in Y, and every line determined by two points in Y is below every point in X.

▶ **Definition 3.1.** H is a **Horton set** if

1. $|H| = 1$; or
2. $|H| \geq 2$; H_0 and H_1 are Horton sets; and H_1 is high above H_0.

Assume that H is a Horton set. We say that an edge $e := (p_i, p_j)$ is a *visible edge* of H if one of the following two conditions are met.

- Both i and j are even and for every even $i < l < j$, the point p_l is below the line passing through e. In this case we say that e is *visible from above*.
- Both i and j are odd and for every odd $i < l < j$, the point p_l is above the line passing through e. In this case we say that e is *visible from below*.

▶ **Lemma 3.2.** *The number of visible edges of H is less than $2n$.*

▶ **Lemma 3.3.** *Let p_i, p_j and p_l be the vertices of a triangle τ of H such that either*

- *(p_i, p_j) is an edge visible from below and $p_l \in H_0$; or*
- *(p_i, p_j) is an edge visible from above and $p_l \in H_1$.*

Then τ is empty. Moreover, every empty triangle of H with at least one vertex in each of H_0 and H_1 is of one these forms.

▶ **Corollary 3.4.** *[10] The number of empty triangles of H is at most $2n^2$.*

3.2 Blockers

Our strategy is to start with a Horton set H of k points and replace each point p_i of H with a cluster C_i of $m \geq k$ points. All of the points of C_i are of the same color and are at a distance of at most ε_1 from p_i. We choose ε_1 to be sufficiently small. Let S be the resulting set. Note that every rainbow triangle of S must have all its vertices in different clusters. Moreover, since each C_i is arbitrarily close to p_i we have the following. If τ is an empty triangle of S with vertices in clusters C_i, C_j and C_l then p_i,p_j and p_l are the vertices of an empty triangle in H. In principle, this gives up to m^3 empty triangles in S per empty

triangle of H. However, we place the points within each cluster in such a way so that only very few of these triangles are actually empty.

We now define real numbers $\varepsilon_1 > \varepsilon_2 > \cdots > \varepsilon_k > 0$. Suppose that ε_r has been defined. We define ε_{r+1} small enough so that the following is satisfied for every triple of distinct indices i, j, l. Let q be a point at distance ε_r from p_l such that q is in the interior of every triangle with vertices p'_i, p'_j and p_l; where p'_i and p'_j are at a distance of at most ε_1 of p_i and p_j, respectively. Then q is in the interior of every triangle with vertices p'_i, p'_j and p'_l, where p'_l is any point at a distance of at most ε_{r+1} from p'_l. In this case we say that q blocks the triangle with vertices p'_i, p'_j and p'_l. For each point p_i of H, we place the points $q_1, \ldots, q_{k-1}$ of C_i at a distance of $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_{k-1}$ from p_i, respectively. We say that q_r is at *layer* r. The remaining points of C_i are placed at a distance of at most ε_k from p_i.

We now describe how these "blocker" points are placed for each point $p_i \in H$. Let $s_1, \ldots, s_r$ be strings of 0's and 1's such that: $s_1 = \emptyset$; $s_{j+1} = s_j 0$ or $s_{j+1} = s_j 1$. Further let all H_{s_j} be Horton sets and $H_{s_1} = H$. Define recursively the sets $H_{s_j 0}$ and $H_{s_j 1}$ as Horton sets, such that $H_{s_j} = H_{s_j 0} \cup H_{s_j 1}$ and $H_{s_j 1}$ is high above $H_{s_j 0}$. Let p_i be contained in every of the sets H_{s_j}. Note that $r \leq \lceil \log_2(k) \rceil$. For every $j = 1, \ldots, r$ we place points q_{2j-1} and q_{2j} in layers $2j-1$ and $2j$, respectively as follows. Sort the points of $H \setminus \{p_i\}$ counterclockwise by angle around p_i

- Suppose that p_i is in $H_{s_j 0}$. Place q_{2j-1}, just after the leftmost point of $H_{s_j 1}$ in counterclockwise order around p_i; place q_{2j}, just before the rightmost point of $H_{s_j 1}$ in counterclockwise order around p_i;
- suppose that p_i is in $H_{s_j 1}$. Place q_{2j-1}, just before the leftmost point of $H_{s_j 0}$ in counterclockwise order around p_i; place q_{2j}, just after the rightmost point of $H_{s_j 0}$ in counterclockwise order around p_i;

For any two consecutive points of H in counterclockwise order around p_i, such that between them there is not yet a blocker, place a blocker in a new layer. Place the remaining points of C_i in any way but at a distance of at most ε_k from p_i.

Let S be the set that results from replacing each $p_i \in H$ with the cluster C_i.

▶ **Theorem 3.5.** *S determines $O(k^3)$ empty rainbow triangles.*

Proof. We classify the empty triangles of H as follows. Let τ be an empty triangle of H. Let s be the string of 0's and 1's such that the vertices of τ are contained in H_s but not in H_{s0} and H_{s1}. We say that τ is in layer $|s|$. Let p_i, p_j, p_l be the vertices of τ, such that p_j and p_l are both contained in H_{s0} or are both contained in H_{s1}. Then τ contains two blocker points at layers at most $2|s|$ in clusters C_j and C_l, respectively. Note τ also contains a blocker point in cluster C_i of layer at most $k-1$. Therefore, τ produces at most

$$4|s|^2 k$$

empty rainbow triangles in S.

By Lemma 3.3, (p_j, p_l) is a visible edge of H_s. Since $|H_s| \leq \lceil k/2^{|s|} \rceil$, by Lemma 3.2 there are at most $2 \lceil k/2^{|s|} \rceil \lceil k/2^{|s|+1} \rceil \leq 8 \left(k^2/2^{2|s|}\right)$ empty triangles in H_s. Thus, for every $0 \leq r \leq \log_2(k)$ there at most $2^r 8 \left(k^2/2^{2r}\right) = 8k^2/2^r$ empty triangles in H of layer r. Therefore, S contains at most

$$\sum_{r=0}^{\lceil \log_2(k) \rceil} \left(4r^2 k\right) \left(8k^2/2^r\right) = 32k^3 \sum_{r=0}^{\lceil \log_2(k) \rceil} \frac{r^2}{2^r} = 192k^3$$

empty rainbow triangles. ◀

4 Empty rainbow 4-gons

A natural generalization is to consider empty rainbow r-gons for $r \geq 4$. Empty r-gons can also be non-convex, in contrast to r-holes, which are convex. If there does not exist an empty rainbow 4-gon, then also empty rainbow r-gons, $r \geq 5$, do not exist. We construct arbitrary large colored point sets which do not contain any empty rainbow 4-gon. Before constructing our point set we observe that the colored point set in Figure 1 does not contain any empty rainbow 4-gon. Note, that we can place arbitrary many further red points between the red points on the lines, such that the point set still does not contain an empty rainbow 4-gon. A more detailed explanation for this can be found in the full version.

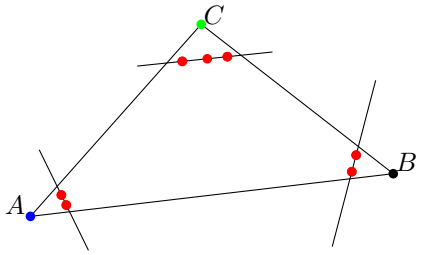


Figure 1 Colored point set without an empty rainbow 4-gon.

We use this observation to construct our point set. First we take a regular $(k-1)$-gon, P, with vertices $p_1, \ldots, p_{k-1}$; we replace every point p_i with a cluster C_i of m points with color i. Let P' be a copy of P with vertices $p'_1, \ldots, p'_{k-1}$, which is rotated by $\frac{360^\circ}{2(k-1)} = \frac{180^\circ}{k-1}$. So $p_1, p'_1, p_2, \ldots, p_{k-1}, p'_{k-1}$ form a regular $2(k-1)$-gon. Let ε be sufficiently small. For every $1 \leq i \leq k-1$, we place the points of C_i at a distance of at most ε from p_i. We place at least $2(k-3)$ points of color k on the line segment $p'_{i-1}p'_i$, for $1 \leq i \leq k-1$ with $p'_0 = p'_{k-1}$, so that the following holds. Let q_1, q_2 be any two consecutive points of P distinct from p_i. In the triangle with vertices p_i, q_1 and q_2 there are at least two points of $p'_{i-1}p'_i$ of color k. Further these points have at least distance ε to the lines $\overline{p_iq_1}$ and $\overline{p_iq_2}$. The construction for $k = 6$ is depicted in Figure 2. Note, that the clusters C_i are drawn enlarged.

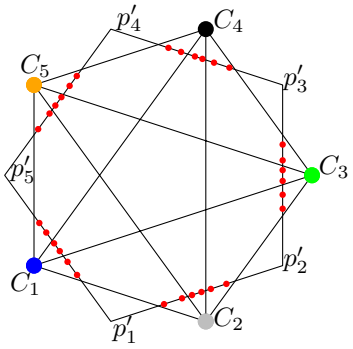


Figure 2 Construction for a 6-colored point set without empty rainbow 4-gons.

▶ **Theorem 4.1.** *All clusters C_i, $1 \leq i \leq k-1$, along with the points with color k describe a k-colored point set without an empty rainbow 4-gon.*

A detailed proof can be found in the full version of the paper. Note that in this construction we have that $m \geq 2k^2 - 8k + 6$. Further the points placed on the line segments $p'_{i-1}p'_i$

can be moved slightly such that the point set still does not contain an empty rainbow quadrilateral but that the points are in general position.

5 Open Problems

We finish the paper with two open problems.

For our results we relied heavily on the fact that $m \geq k$; it would be interesting to obtain sharp bounds of $f(m, k)$ when $m < k$.

▶ Problem 1. Compute $f(m, k)$ for $m < k$.

We constructed a k-colored point set with the same number of points in each color class and without (convex or non-convex) empty rainbow 4-gon. This point set contains many empty monochromatic 4-gons. This leads us to the following question.

▶ Problem 2. Does every sufficiently large k-colored ($k \geq 4$) point set with the same number of points in each color class contain an empty rainbow 4-gon or an empty monochromatic 4-gon?

References

1 O. Aichholzer, M. Balko, T. Hackl, J. Kyncl, I. Parada, M. Scheucher, P. Valtr, and B. Vogtenhuber. A superlinear lower bound on the number of 5-holes. In *33rd International Symposium on Computational Geometry (SoCG 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

2 O. Aichholzer, R. Fabila-Monroy, D. Flores-Peñaloza, T. Hackl, C. Huemer, and J. Urrutia. Empty monochromatic triangles. *Comput. Geom.*, 42(9):934–938, 2009.

3 O. Aichholzer, R. Fabila-Monroy, T. Hackl, C. Huemer, A. Pilz, and B. Vogtenhuber. Lower bounds for the number of small convex k-holes. *Comput. Geom.*, 47(5):605–613, 2014.

4 O. Aichholzer, T. Hackl, C. Huemer, F. Hurtado, and B. Vogtenhuber. Large bichromatic point sets admit empty monochromatic 4-gons. *SIAM J. Discrete Math.*, 23(4):2147–2155, 2010.

5 O. Aichholzer, J. Urrutia, and B. Vogtenhuber. Balanced 6-holes in linearly separable bichromatic point sets. *Electronic Notes in Discrete Mathematics*, 44:181–186, 2013.

6 I. Bárány and P. Valtr. Planar point sets with a small number of empty convex polygons. *Studia Sci. Math. Hungar.*, 41(2):243–266, 2004.

7 D. Basu, K. Basu, B. B. Bhattacharya, and S. Das. Almost empty monochromatic triangles in planar point sets. *Discrete Appl. Math.*, 210:207–213, 2016.

8 S. Bereg, J. M. Díaz-Báñez, R. Fabila-Monroy, P. Pérez-Lantero, A. Ramírez-Vigueras, T. Sakai, J. Urrutia, and I. Ventura. On balanced 4-holes in bichromatic point sets. *Comput. Geom.*, 48(3):169–179, 2015.

9 P. Brass. Empty monochromatic fourgons in two-colored point sets. *Geombinatorics*, 14(2):5–7, 2004.

10 I. Bárány and Z. Furedi. Empty simplices in euclidean space. *Canadian Mathematical Bulletin*, 30, 12 1987.

11 K. Dehnhardt. *Leere konvexe Vielecke in ebenen Punktmengen.* PhD thesis, TU, Braunschweig, 1987.

12 O. Devillers, F. Hurtado, G. Károlyi, and C. Seara. Chromatic variants of the Erdős-Szekeres theorem on points in convex position. *Comput. Geom.*, 26(3):193–208, 2003.

13 A. Dumitrescu. Planar sets with few empty convex polygons. *Studia Sci. Math. Hungar.*, 36(1-2):93–109, 2000.

14 P. Erdős. Some more problems on elementary geometry. *Austral. Math. Soc. Gaz.*, 5(2):52–54, 1978.

15 P. Erdős and G. Szekeres. A combinatorial problem in geometry. *Compositio Math.*, 2:463–470, 1935.

16 E. Friedman. 30 two-colored points with no empty monochromatic convex fourgons. *Geombinatorics*, 14(2):53–54, 2004.

17 A. García. *A Note on the Number of Empty Triangles*, pages 249–257. Springer Berlin Heidelberg, 2012.

18 T. Gerken. Empty convex hexagons in planar point sets. *Discrete Comput. Geom.*, 39(1-3):239–272, 2008.

19 H. Harborth. Konvexe Fünfecke in ebenen Punktmengen. *Elem. Math.*, 33(5):116–118, 1978.

20 J. D. Horton. Sets with no empty convex 7-gons. *Canad. Math. Bull.*, 26(4):482–484, 1983.

21 C. Huemer and C. Seara. 36 two-colored points with no empty monochromatic convex fourgons. *Geombinatorics*, 19(1):5–6, 2009.

22 M. Katchalski and A. Meir. On empty triangles determined by points in the plane. *Acta Math. Hungar.*, 51(3-4):323–328, 1988.

23 V. Koshelev. On Erdős–Szekeres problem and related problems. *arXiv preprint arXiv:0910.2700*, 2009.

24 L. Liu and Y. Zhang. Almost empty monochromatic quadrilaterals in planar point sets. *Math. Notes*, 103(3-4):415–429, 2018.

25 C. M. Nicolás. The empty hexagon theorem. *Discrete Comput. Geom.*, 38(2):389–397, 2007.

26 J. Pach and G. Tóth. Monochromatic empty triangles in two-colored point sets. *Discrete Appl. Math.*, 161(9):1259–1261, 2013.

27 P. Valtr. On the minimum number of empty polygons in planar point sets. *Studia Sci. Math. Hungar.*, 30(1-2):155–163, 1995.

28 R. van Gulik. 32 two-colored points with no empty monochromatic convex fourgons. *Geombinatorics*, 15(1):32–33, 2005.

Bitonicity of Euclidean TSP in Narrow Strips*

Henk Alkema[1], Mark de Berg[2], and Sándor Kisfaludi-Bak[3]

1 **Department of Mathematics and Computer Science, TU Eindhoven, the Netherlands**
h.y.alkema@tue.nl
2 **Department of Mathematics and Computer Science, TU Eindhoven, the Netherlands**
m.t.d.berg@tue.nl
3 **Max Planck Institute for Informatics, Germany**
sandor.kisfaludi-bak@mpi-inf.mpg.de

Abstract

We investigate how the complexity of Euclidean TSP for point sets $P \subset (-\infty, +\infty) \times [0, \delta]$ depends on the strip width δ. We prove that if the points have distinct integer x-coordinates, a shortest bitonic tour (which can be computed in $O(n \log^2 n)$ time using an existing algorithm) is guaranteed to be a shortest tour overall when $\delta \leqslant 2\sqrt{2}$, a bound which is best possible.

1 Introduction

In the Traveling Salesman Problem one is given an edge-weighted complete graph and the goal is to compute a tour—a simple cycle visiting all nodes—of minimum total weight. Due to its practical as well as theoretical importance, the Traveling Salesman Problem and its many variants are among the most famous problems in computer science and combinatorial optimization. In this paper we study the Euclidean version of the problem. In Euclidean TSP the input is a set P of n points in $\mathbb{R}^d$, and the goal is to compute a minimum-length tour visiting each point. Euclidean TSP in the plane was proven to be NP-hard in the 1970s [1, 2]. Unlike the general (metric) version, however, it can be solved in *subexponential* time, that is, in time $2^{o(n)}$. In particular, Kann [3] and Hwang *et al.* [4] presented algorithms with $n^{O(\sqrt{n})}$ running time. Smith and Wormald [5] gave a subexponential algorithm that works in any (fixed) dimension; its running time in $\mathbb{R}^d$ is $n^{O(n^{1-1/d})}$. Very recently De Berg *et al.* [6] improved this to $2^{O(n^{1-1/d})}$, which is tight up to constant factors in the exponent, under the Exponential-Time Hypothesis (ETH) [7].

There has also been considerable research on special cases of Euclidean TSP that are polynomial-time solvable. One example is Bitonic TSP, where the goal is to find a shortest *bitonic* tour. (A tour is bitonic if any vertical line crosses it at most twice; here the points from the input set P are assumed to have distinct x-coordinates.) It is a classic exercise [8] to prove that Bitonic TSP can be solved in $O(n^2)$ time by dynamic programming. De Berg *et al.* [9] showed how to speed up the algorithm to $O(n \log^2 n)$.

Our contribution. The computational complexity of Euclidean TSP in $\mathbb{R}^d$ is $2^{\Theta(n^{1-1/d})}$ (for $d \geqslant 2$), assuming ETH. Thus the complexity depends heavily on the dimension d. This is most pronounced when we compare the complexity for $d = 2$ with the trivial case $d = 1$: in the plane Euclidean TSP takes $2^{\Theta(\sqrt{n})}$ time in the worst case, while the 1-dimensional case is trivially solved in $O(n \log n)$ time by just sorting the points. We study the complexity of Euclidean TSP for planar point sets that are "almost 1-dimensional". In particular,

* The work in this paper is supported by the Netherlands Organisation for Scientific Research (NWO) through Gravitation-grant NETWORKS-024.002.003.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

we assume the point set P is contained in the strip $(-\infty, \infty) \times [0, \delta]$ for some relatively small δ, and that all points have distinct integer x-coordinates. BITONIC TSP can be solved in $O(n \log^2 n)$ time [9]. It is natural to conjecture that for sufficiently small δ, an optimal bitonic tour on P is a shortest tour overall. We give a (partially computer-assisted) proof that this is indeed the case: we prove that when $\delta \leqslant 2\sqrt{2}$ an optimal bitonic tour is optimal overall, and we show that the bound $2\sqrt{2}$ is best possible.

Notation and terminology. Let $P := \{p_1, \ldots, p_n\}$ be a set of points with distinct integer x-coordinates in a horizontal strip of width δ—we call such a strip a *δ-strip*—which we assume without loss of generality to be the strip $(-\infty, \infty) \times [0, \delta]$. We denote the x-coordinate of a point $p \in \mathbb{R}^2$ by $x(p)$, and its y-coordinate by $y(p)$. To simplify the notation, we also write x_i for $x(p_i)$, and y_i for $y(p_i)$. We sort the points in P such that $x_i < x_{i+1}$ for all $1 \leqslant i < n$.

For two points $p, q \in \mathbb{R}^2$, we write pq to denote the *directed* edge from p to q. The length of an edge pq is denoted by $|pq|$, and the total length of a set E of edges is denoted by $\|E\|$.

A *separator* is a vertical line not containing any of the points in P that separates P into two non-empty subsets. For our purposes, two separators s, s' that induce the same partitioning of P are equivalent. Therefore, we can define $\mathcal{S} := \{s_1, \ldots, s_{n-1}\}$ as the set of all combinatorially distinct separators, obtained by taking one separator between any two points p_i, p_{i+1}. Let E, F be sets of edges with endpoints in P. The *tonicity of E at a separator s*, written as $\mathrm{ton}(E, s)$, is the number of edges in E crossing s. We say that E has *lower tonicity* than F, denoted by $E \preccurlyeq F$, if $\mathrm{ton}(E, s_i) \leqslant \mathrm{ton}(F, s_i)$ for all $s_i \in \mathcal{S}$. E has *strictly lower tonicity* than F, denoted by $E \prec F$, if there also exists at least one i for which $\mathrm{ton}(E, s_i) < \mathrm{ton}(F, s_i)$. Finally, we call E *bitonic* if $\mathrm{ton}(E, s_i) = 2$ for all $s_i \in \mathcal{S}$.

2 Bitonicity for points with integer x-coordinates

The goal of this section is to prove the following theorem.

▶ **Theorem 1.** *Let P be a set of points with distinct and integer x-coordinates in a δ-strip. When $\delta \leqslant 2\sqrt{2}$, a shortest bitonic tour on P is a shortest tour overall. Moreover, for any $\delta > 2\sqrt{2}$ there is a point set P in a δ-strip such that a shortest bitonic tour on P is not a shortest tour overall.*

The construction for the case $\delta > 2\sqrt{2}$ is shown in Fig. 1. It is easily verified that, up to symmetrical solutions, the tours T_1 and T_2 are the only candidates for the shortest tour. Observe that $\|T_2\| - \|T_1\| = |p_1p_4| - |p_4p_5| = 3 - \sqrt{1 + \delta^2}$. Hence, for $\delta > 2\sqrt{2}$ we have $\|T_2\| < \|T_1\|$, which proves the second statement of Theorem 1. The remainder of the section is devoted to proving the first statement.

Let P be a point set in a δ-strip for $\delta = 2\sqrt{2}$, where all points in P have distinct integer x-coordinates. Among all shortest tours on P, let T_{opt} be one that is minimal with respect to the $\preccurlyeq$-relation; T_{opt} exists since the number of different tours on P is finite. We claim that T_{opt} is bitonic, proving the upper bound of Theorem 1.

Suppose for a contradiction that T_{opt} is not bitonic. Let $s^* \in \mathcal{S}$ be the rightmost separator for which $\mathrm{ton}(T_{\mathrm{opt}}, s^*) > 2$. We must have $\mathrm{ton}(T_{\mathrm{opt}}, s^*) = 4$ because otherwise $\mathrm{ton}(T_{\mathrm{opt}}, s) > 2$ for the separator $s \in \mathcal{S}$ immediately to the right of s^*, since there is only one point between s^* and s. Let F be the four edges of T_{opt} crossing s^*, and let E be the remaining edges of T_{opt}. Let Q be the set of endpoints of the edges in F. We will argue that there exists a set F' of edges with endpoints in Q such that $E \cup F'$ is a tour and (i) $\|F'\| < \|F\|$, or (ii) $\|F'\| = \|F\|$ and $F' \prec F$. We will call such an F' *superior to* F. Option (i) contradicts that T_{opt} is a shortest tour, and (ii) contradicts that T_{opt} is a shortest

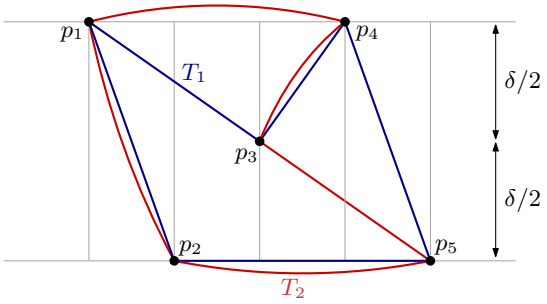


Figure 1 Construction for $\delta > 2\sqrt{2}$ for Theorem 1. The grey vertical segments are at distance 1 from each other. If $\delta > 2\sqrt{2}$ then T_1, the shortest bitonic tour (in blue), is longer than T_2, the shortest non-bitonic tour (in red).

tour, minimal with respect to $\preccurlyeq$ (since $E \cup F' \prec E \cup F$ if and only if $F' \prec F$). Hence, proving a superior set F' exists finishes the proof.

The remainder of the proof proceeds in two steps. In the first step we argue that we can assume without loss of generality that Q uses consecutive integer x-coordinates. In the second step we then give a computer-assisted proof that a superior set F' exists.

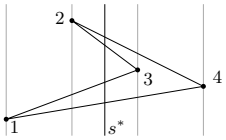
Step 1: Reduction to an instance where Q has consecutive x-coordinates. The goal of Step 1 is to move the points in Q to obtain a set $\overline{Q}$ of points with consecutive x-coordinates in such a way that finding a superior set $\overline{F}'$ for $\overline{Q}$ also gives us a superior set F' for Q. Let $\overline{F}$ be the same set as F, but now on the moved point set $\overline{Q}$, and define $\widetilde{E}$, the *connectivity pattern* of E, to be the set of edges obtained by contracting each path in E to a single edge.

▶ **Lemma 2.** *Let $T_{\text{opt}}, s^*, E, \widetilde{E}, F, \overline{F}$ and Q be defined as above. Then there exists a $\overline{Q}$ s.t.:*
1. *$\widetilde{E}, \overline{F}$ and $\overline{Q}$ adhere to one of the six cases in Figure 2.*
2. *If there exists an $\overline{F}'$ superior to $\overline{F}$, there exists an F' superior to F.*

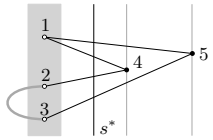
Sketch of proof. We start by taking $\overline{Q} = Q$. Now, property 2 trivially holds. We will now transform $\overline{Q}$, making sure that property 2 keeps holding, until property 1 also holds.
Let p be a point of $\overline{Q}$ which only has one incident edge pq in F. Suppose we move p some distance d along pq towards q. The total length of any candidate $\overline{F}'$ decreases by at most d by doing so, while $\|\overline{F}\|$ decreases by exactly d. Similar reasoning can be made about the tonicity. Therefore, this move does not affect the desired properties. If a point r has two incident edges pr, rq in $\overline{F}$, we can split it into two points r_1, r_2, and add an edge r_1r_2 between them. Then, we can move them towards p and q, respectively. Doing so also does not affect the desired properties. Since all edges of $\overline{F}$ cross the separator s^*, the points can be moved towards each other such that they have consecutive integer x-coordinates. ◀
The complete proof can be found in the full version of this paper.

Step 2: Finding the set F'. The goal of Step 2 of the proof is the following: given a tour $\widetilde{E} \cup \overline{F}$ on a point set $\overline{Q}$ adhering to one of the six cases in Figure 2, show that there exists a set $\overline{F}'$ of edges superior to $\overline{F}$. Lemma 2 then implies that a superior set of edges exists for any Q, E, F, finishing the proof of Theorem 1.

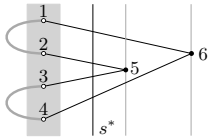
Each of the six cases has several subcases, depending on the left-to-right order of the vertices inside the gray rectangles in the figure. Once we fixed the ordering, we can still vary the y-coordinates in the range $[0, \delta]$, which may lead to scenarios where different sets $\overline{F}'$ are required. We handle this potentially huge amount of cases in a computer-assisted



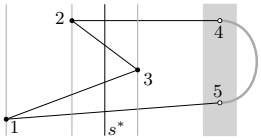
- $(n_{\text{left}}, n_{\text{right}}) = (2, 2)$
- $\overline{F} = \{(1,3), (1,4), (2,3), (2,4)\}$
- $\widetilde{E} = \emptyset$
- $X = [\{-2\}, \{-1\}, \{0\}, \{1\}]$



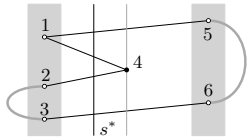
- $(n_{\text{left}}, n_{\text{right}}) = (3, 2)$
- $\overline{F} = \{(1,4), (1,5), (2,4), (3,5)\}$
- $\widetilde{E} = \{(2,3)\}$
- $X = [\{-3,-2,-1\}, \{-3,-2,-1\}, \{-3,-2,-1\}, \{0\}, \{1\}]$



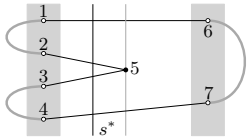
- $(n_{\text{left}}, n_{\text{right}}) = (4, 2)$
- $\overline{F} = \{(1,6), (2,5), (3.5), (4,6)\}$
- $\widetilde{E} = \{(1,2), (3,4)\}$
- $X = [\{-3,-2,-1\}, \{-3,-2,-1\}, \{-3,-2,-1\}, \{-3,-2,-1\}, \{0\}, \{2\}]$



- $(n_{\text{left}}, n_{\text{right}}) = (2, 3)$
- $\overline{F} = \{(1,3), (1,5), (2,3), (2,4)\}$
- $\widetilde{E} = \{(4,5)\}$
- $X = [\{-2\}, \{-1\}, \{0\}, \{1,2\}, \{1,2\}]$



- $(n_{\text{left}}, n_{\text{right}}) = (3, 3)$
- $\overline{F} = \{(1,4), (1,5), (2,4), (3,6)\}$
- $\widetilde{E} = \{(2,3), (5,6)\}$
- $X = [\{-3,-2,-1\}, \{-3,-2,-1\}, \{-3,-2,-1\}, \{0\}, \{1,2\}, \{1,2\}]$



- $(n_{\text{left}}, n_{\text{right}}) = (4, 3)$
- $\overline{F} = \{(1,6), (2,5), (3.5), (4,7)\}$
- $\widetilde{E} = \{(1,2), (3,4), (6,7)\}$
- $X = [\{-3,-2,-1\}, \{-3,-2,-1\}, \{-3,-2,-1\}, \{-3,-2,-1\}, \{0\}, \{1,2\}, \{1,2\}]$

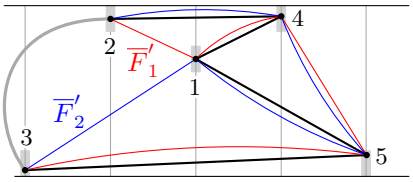
Figure 2 The six different cases that result after applying Step 1 of the proof. Points indicated by filled disks have a fixed x-coordinate. The left-to-right order of points drawn inside a grey rectangle, on the other hand, is not known yet. The vertical order of the edges is also not fixed, as the points can have any y-coordinate in the range $[0, 2\sqrt{2}]$.

manner, using an automated prover *FindShorterTour*$(n_{\text{left}}, n_{\text{right}}, \overline{F}, \widetilde{E}, X, \delta, \varepsilon)$. The input parameter X is an array where $X[i]$ specifies the set from which the x-coordinate of the i-th point in the given scenario may be chosen, where we assume w.l.o.g. that $x(s^*) = -1/2$; see Fig. 2. The role of the parameter ε will be explained below.

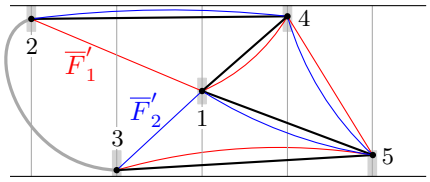
The output of *FindShorterTour* is a list of *scenarios* and an *outcome* for each scenario. A scenario contains for each point q an x-coordinate $x(q)$ from the set of allowed x-coordinates for q, and a range y-range$(q) \subseteq [0, 2\sqrt{2}]$ for its y-coordinate, where the y-range is an interval of length at most ε. The outcome is either Success or Fail. Success means that a set $\overline{F}'$ has been found with the desired properties: $\widetilde{E} \cup \overline{F}'$ is a tour, and for all possible instantiations of the scenario—that is, all choices of y-coordinates from the y-ranges in the scenario—we have $\|\overline{F}'\| < \|\overline{F}\|$. Fail means that such an $\overline{F'}$ has not been found, but it does not guarantee that such an $\overline{F'}$ does not exist for this scenario. The list of scenarios is complete in the sense that for any instantiation of the input case there is a scenario that covers it.

FindShorterTour works brute-force, by checking all possible combinations of x-coordinates and subdividing the y-coordinate ranges until a suitable $\overline{F}'$ can be found or until the y-ranges have length at most ε. The implementation details of the procedure can be found in the full version of this paper.

Note that case $(n_{\text{left}}, n_{\text{right}}) = (2, 3)$ in Fig. 2 is a subcase of case $(n_{\text{left}}, n_{\text{right}}) = (3, 2)$, if we exchange the roles of the points lying to the left and to the right of s^*. Hence, we ignore this subcase and run our automated prover on the remaining five cases, where we set $\varepsilon := 0.001$. It successfully proves the existence of a suitable set $\overline{F}'$ in four cases; the



- $X = [\{-1\}, \{-2\}, \{-3, \}, \{0\}, \{1\}]$
- $Y = [[1.61, 1.62], [2.82, 2\sqrt{2}],$ $[0, 0.01], [2.82, 2\sqrt{2}], [0, 0.01]]$



- $X = [\{-1\}, \{-3\}, \{-2, \}, \{0\}, \{1\}]$
- $Y = [[1.41, 1.42], [2.82, 2\sqrt{2}],$ $[0, 0.01], [2.82, 2\sqrt{2}], [0, 0.01]]$

Figure 3 Two scenarios covering all subscenarios where the automated prover fails. Each point has a fixed x-coordinate and a y-range specified by the array Y; the resulting possible locations are shown as small grey rectangles (drawn larger than they actually are for visibility). For all subscenarios, at least one of $\overline{F}'_1$ (in red) and $\overline{F}'_2$ (in blue) is at most as long as $\overline{F}$ (in black).

case where the prover fails is the case $(n_{\text{left}}, n_{\text{right}}) = (3, 2)$. For this case it fails for the two scenarios depicted in Fig. 3; all other scenarios for these cases are handled successfully (up to symmetries). For both scenarios we consider two alternatives for the set $\overline{F}'$: the set $\overline{F}'_1$ shown in red in Fig. 3, and the set $\overline{F}'_2$ shown in blue in Fig. 3. We will show that in any instantiation of both scenarios, either $\overline{F}'_1$ or $\overline{F}'_2$ is at least as short as $\overline{F}$; since both alternatives are bitonic this finishes the proof.

For $1 \leqslant i \leqslant 5$, let q_i be the point labeled i in Fig. 3. We first argue that (for both scenarios) we can assume without loss of generality that $y(q_2) = y(q_4) = 2\sqrt{2}$ and $y(q_3) = y(q_5) = 0$. To this end, consider arbitrary instantiations of these scenarios, and imagine moving q_2 and q_4 up to the line $y = 2\sqrt{2}$, and moving q_3 and q_5 down to the line $y = 0$. It suffices to show, for $i \in \{1, 2\}$, that if we have $\|\overline{F}'_i\| \leqslant \|\overline{F}\|$ after the move, then we also have $\|\overline{F}_i\| \leqslant \|\overline{F}\|$ before the move. This can easily be proven by repeatedly applying the following observation.

▶ **Observation 3.** *Let a, b, c be three points. Let ℓ be the vertical line through c, and let us move c downwards along ℓ. Let α be the smaller angle between ac and ℓ if $y(c) < y(a)$, and the larger angle otherwise, and let β be the smaller angle between bc and ℓ if $y(c) < y(b)$, and the larger angle otherwise, and suppose $\alpha < \beta$ throughout the move. Then the move increases $|ac|$ more than it increases $|bc|$.*

So now assume $y(q_2) = y(q_4) = 2\sqrt{2}$ and $y(q_3) = y(q_5) = 0$. Consider the left scenario in Fig. 3, and let $y := y(q_3)$. If $y \geqslant (8\sqrt{2})/7$ then

$$|q_2q_1| + |q_4q_5| = \sqrt{1 + (2\sqrt{2} - y)^2} + 3 \leqslant 2 + \sqrt{4 + y^2} = |q_2q_4| + |q_1q_5|,$$

so $\|\overline{F}'_1\| \leqslant \|\overline{F}\|$. On the other hand, If $y \leqslant (8\sqrt{2})/7$ then

$$|q_3q_1| + |q_4q_5| = \sqrt{4 + y^2} + 3 \leqslant \sqrt{1 + (2\sqrt{2} - y)^2} + 4 = |q_1q_4| + |q_3q_5|,$$

so $\|\overline{F}'_2\| \leqslant \|\overline{F}\|$. So either $\overline{F}'_1$ or $\overline{F}'_2$ is at least as short as $\overline{F}$, finishing the proof for the left scenario in Fig. 3. The proof for the right scenario in Fig. 3 is analogous, with cases $y \geqslant \sqrt{2}$ and $y \leqslant \sqrt{2}$. This finishes the proof for the right scenario and, hence, for Theorem 1.

3 Concluding remarks

In our paper, we proved that for points with integer x-coordinates in a strip of width δ, an optimal bitonic tour is optimal overall when $\delta \leqslant 2\sqrt{2}$. The proof of this bound, which is

tight in the worst case, is partially automated to reduce the potentially very large number of cases to two worst-case scenarios. It would be interesting to see if a direct proof can be given for this fundamental result. Finally, we note that the proof of Theorem 1 can easily be adapted to point sets of which the x-coordinates of the points need not be integer, as long as the difference between x-coordinates of any two consecutive points is at least 1.

In the full version of this paper, we also investigate the case $\delta > 2\sqrt{2}$. We present a fixed-parameter tractable algorithm with respect to δ. More precisely, our algorithm has running time $2^{O(\sqrt{\delta})}n^2$ for point sets where each $1 \times \delta$ rectangle inside the strip contains $O(1)$ points. For point sets where the points are chosen uniformly at random from the rectangle $[0, n] \times [0, \delta]$, it has an expected running time $2^{O(\sqrt{\delta})}n^2 + O(n^3)$.

References

1 M.R. Garey, R.L. Graham, and D.S. Johnson. Some NP-complete geometric problems. In *Proc. 8th ACM Symp. Theory Comp. (STOC)*, pages 10–22, 1976.

2 C.H. Papadimitriou. The Euclidean traveling salesman problem is NP-complete. *Theoret. Comput. Sci.* 4(3): 237–244 (1977).

3 V. Kann. *On the approximability of NP-complete optimization problems.* Ph.D. Dissertation, Royal Institute of Technology, Stockholm, 1992.

4 R.Z. Hwang, R.C. Chang, and R.C.T. Lee. The searching over separators strategy to solve some NP-hard problems in subexponential time. *Algorithmica* 9(4): 398–423 (1993).

5 W.D. Smith and N.C. Wormald. Geometric separator theorems and applications. In *Proc. 39th IEEE Symp. Found. Comput. Sci. (FOCS)*, pages 232–243, 1998.

6 M. de Berg, H.L. Bodlaender, S. Kisfaludi-Bak, and S. Kolay. An ETH-tight exact algorithm for Euclidean TSP. In *Proc. 59th IEEE Symp. Found. Comput. Sci. (FOCS)*, pages 450–461, 2018.

7 R. Impagliazzo and R. Paturi. On the complexity of k-SAT. *J. Comput. Syst. Sci.* 62(2): 367–375 (2001).

8 T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms (3rd edition).* MIT Press, 2009.

9 M. de Berg, K. Buchin, B.M.P. Jansen, and G. Woeginger. Fine-grained complexity analysis of two classic TSP variants. In *Proc. 43rd Int. Conf. Automata Lang. Prog. (ICALP)*, pages 5:1–5:14, 2016.

Experimental Evaluation of Straight Skeleton Implementations Based on Exact Arithmetic*

Günther Eder[1], Martin Held[1], and Peter Palfrader[1]

1 **Universität Salzburg, FB Computerwissenschaften, Salzburg, Austria,**
{geder,held,palfrader}@cs.sbg.ac.at

Abstract

We present C++ implementations of two algorithms for computing straight skeletons in the plane, based on exact arithmetic. One code, named Surfer2, can handle multiplicatively weighted planar straight-line graphs (PSLGs) while our second code, Monos, is specifically targeted at monotone polygons. Both codes are available on GitHub. We sketch implementational and engineering details and discuss the results of an extensive performance evaluation in which we compared Surfer2 and Monos to the straight-skeleton package included in Cgal. Our tests provide ample evidence that both implementations can be expected to be faster and to consume significantly less memory than the Cgal code.

1 Introduction

Straight skeletons were introduced to computational geometry by Aichholzer et al. [2]. Suppose that the edges of a simple polygon P move inwards with unit speed in a self-parallel manner, thus generating mitered offsets inside of P. Then the *(unweighted) straight skeleton* of P is the geometric graph whose edges are given by the traces of the vertices of the shrinking mitered offset curves of P. The process of simulating the shrinking offsets is called *wavefront propagation*. In the presence of multiplicative weights, wavefront edges no longer move at unit speed. Rather, every edge moves at its own constant speed; see Figure 1. Straight skeletons are known to have applications in diverse fields, with the modeling of roof-like structures being one of the more prominent ones [14, 10, 11]. We refer to Huber [12] for a detailed discussion of typical applications.

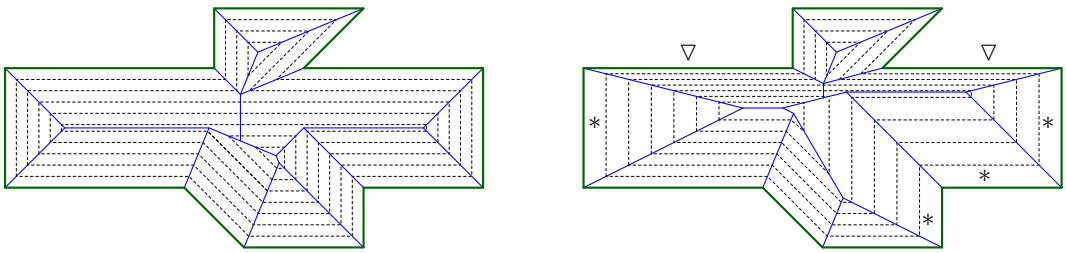


Figure 1 Left: The (unweighted) straight skeleton (in blue) plus a family of wavefronts (dashed) for the green polygon. Right: The weighted straight skeleton for the case that edges marked with $*$ have twice the weight and edges marked with $\triangledown$ have half the weight of the unmarked edges.

The straight-skeleton algorithms with the best worst-case bounds are due to Eppstein and Erickson [9] and Vigneron et al. [6, 17]. These algorithms seem difficult to implement. Indeed, progress on implementations has been rather limited so far. The first comprehensive code for computing straight skeletons was implemented by Cacciola [5] and is shipped with

* Work supported by Austrian Science Fund (FWF): Grants ORD 53-VO and P31013-N31.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

CGAL [16]. It handles polygons with holes as input. The straight-skeleton code Stalgo by Huber and Held [13] handles PSLGs as input and runs in $O(n \log n)$ time and $O(n)$ space in practice. (A PSLG is an embedding of a planar graph such that all edges are straight-line segments which do not intersect pairwise except at common end-points.) However, it would be difficult to extend to weighted skeletons [7].

2 Implementations

Monotone Polygons. Biedl et al. [4] describe an $\mathcal{O}(n \log n)$ time algorithm to compute the straight skeleton of a simple n-vertex x-monotone polygon $\mathcal{P}$. Their algorithm consists of two steps: (1) The polygon $\mathcal{P}$ is split into an upper and lower monotone chain, and the straight skeleton of each chain is computed individually by means of a classical wavefront propagation. (2) The final straight skeleton $\mathcal{S}(\mathcal{P})$ is obtained by merging these two straight skeletons.

Weighted PSLGs. Aichholzer and Aurenhammer [1] describe an algorithm for computing the straight skeleton of general PSLGs in the plane. It carries over to multiplicatively weighted input in a natural way, provided that all weights are positive. Their approach constructs the straight skeleton by simulating a wavefront propagation. As the wavefront sweeps the plane, they maintain a kinetic triangulation of that part of the plane which has not yet been swept. This triangulation is obtained by triangulating the area inside the convex hull of all wavefronts. Furthermore, all edges of the convex hull are linked with a dummy vertex at infinity.

The area of each triangle of this kinetic triangulation changes over time as its vertices, which are vertices of the wavefront, move along angular (straight-line) bisectors of the input edges. Every change in the topology of the wavefront is witnessed by a triangle collapse. (But not all triangle collapses correspond to changes of the wavefront topology.) We refer to Palfrader et al. [15] for more details of this algorithm.

Codes. Our implementations, Monos and Surfer2, of these two algorithms use Cgal's `Exact_predicates_exact_constructions_kernel_with_sqrt` algebraic kernel, which is backed by Core's `Core::Expr` exact number type. (Surfer2 can also be run with standard IEEE 754 arithmetic.) Both source codes are provided on GitHub and can be used freely under the GPL(v3) license: `https://github.com/cgalab/monos` and `https://github.com/cgalab/surfer2`.

3 Engineering Aspects

Careful algorithm engineering was applied to both Monos and Surfer2. Due to lack of space we only sketch a few of our engineering considerations. For instance, a major computational task to be carried out by Monos during the merge step are intersection tests and intersection computations between bisectors and skeleton arcs. Initially, we applied Cgal's `do_intersect` and `intersection` rather naïvely to an arc segment (seen as a straight-line segment) and a bisector. However, explicitly deciding whether the end-points of an arc segment lie on different sides of the supporting line of a merge bisector is sufficient to decide whether an intersection occurs. Once we know that an intersection occurs then we apply Cgal's `intersection` routine to the supporting lines of the arc and the bisector. Tests showed average runtime savings of about 9 % when using the latter method. Similarly,

switching from C++'s `std::set` to a self-developed binary min-heap resulted in an average performance gain of 5 %.

While the actual collapse time of a triangle of the kinetic triangulation is one of the roots of a quadratic polynomial, solving quadratics is not always necessary. Avoiding root finding for quadratic polynomials will increase accuracy when working with limited-precision data types, and it will result in less complex expression trees when working with exact numbers as provided by CORE's `CORE::Expr`. In particular, it will avoid the computation of another square root. Hence, SURFER2 tries to employ geometric knowledge derived from local combinatoric properties as much as this is possible. For instance, consider a triangle with exactly one incident wavefront edge. Its collapse can correspond to an edge event, split event or flip event but we can determine each such event without computing the roots of a determinant: The times of split and flip events can be found by considering the distance between the vertex opposite the wavefront edge to the supporting line of the wavefront edge. This distance is linear in time and when it passes through zero, we either have a flip or split event as the vertex comes to lie on the supporting line of the wavefront edge.

The use of CORE's `CORE::Expr` makes it easy for SURFER2 to know which events happen simultaneously. The significant price to be paid is that comparisons are no longer unit-cost. Hence, SURFER2 attempts to reduce the number of actual comparisons of event times where possible. E.g., if a closed loop partitions the plane into two connected components then the straight skeleton within one component is entirely independent of the straight skeleton within the other component, thus allowing SURFER2 to avoid comparing event times if the events happen within different components.

4 Experimental Results

Setup. All runtime tests were carried out on a 2015 Intel Core i7-6700 CPU. For most of our tests, memory consumption was limited to 10 GiB. The codes were compiled with `clang++`, version 7.0.1, against CGAL 5.0 except where stated otherwise.

The default setting for CGAL's straight-skeleton package [5] is to use the exact predicates but inexact constructions kernel that ships with CGAL. The use of this kernel ensures that the straight skeleton computed is combinatorially correct, even if the locations of the nodes need not be correct. CGAL's straight skeleton package can also be run with the exact predicates and exact constructions kernel. However this causes the runtimes to increase by a factor of roughly 100. Our codes, MONOS and SURFER2, use the CGAL exact predicates and exact constructions with square-root kernel by default, as we construct wavefront vertices with velocities, and these computations involve square roots.

We tested both CGAL and SURFER2 on many different classes of polygons. Our test data consists of real-world (multiply-connected) polygons as well as of synthetic data generated by RPG [3] and similar tools provided by the Salzburg Database [8]. For a polygon that has holes we used CGAL's straight-skeleton code that supports holes. Otherwise we used the implementation which only supports simple polygons.

Additionally, we tested both of our codes, MONOS and SURFER2, with large monotone polygons, for up to 10^6 vertices. (We did not run CGAL on these inputs due to memory constraints.) Given the lack of a sufficiently large number of monotone real-world inputs, we used RPG [3] to automatically generate thousands of monotone input polygons. We also ran SURFER2 on hundreds of real-world PSLGs. Most of our data came from GIS sources and represents road and river networks, contour lines and the like. The runtimes on those inputs are quite comparable to the runtimes for polygons. That is, the test results presented

here are also representative for Surfer2's performance on real-world data.

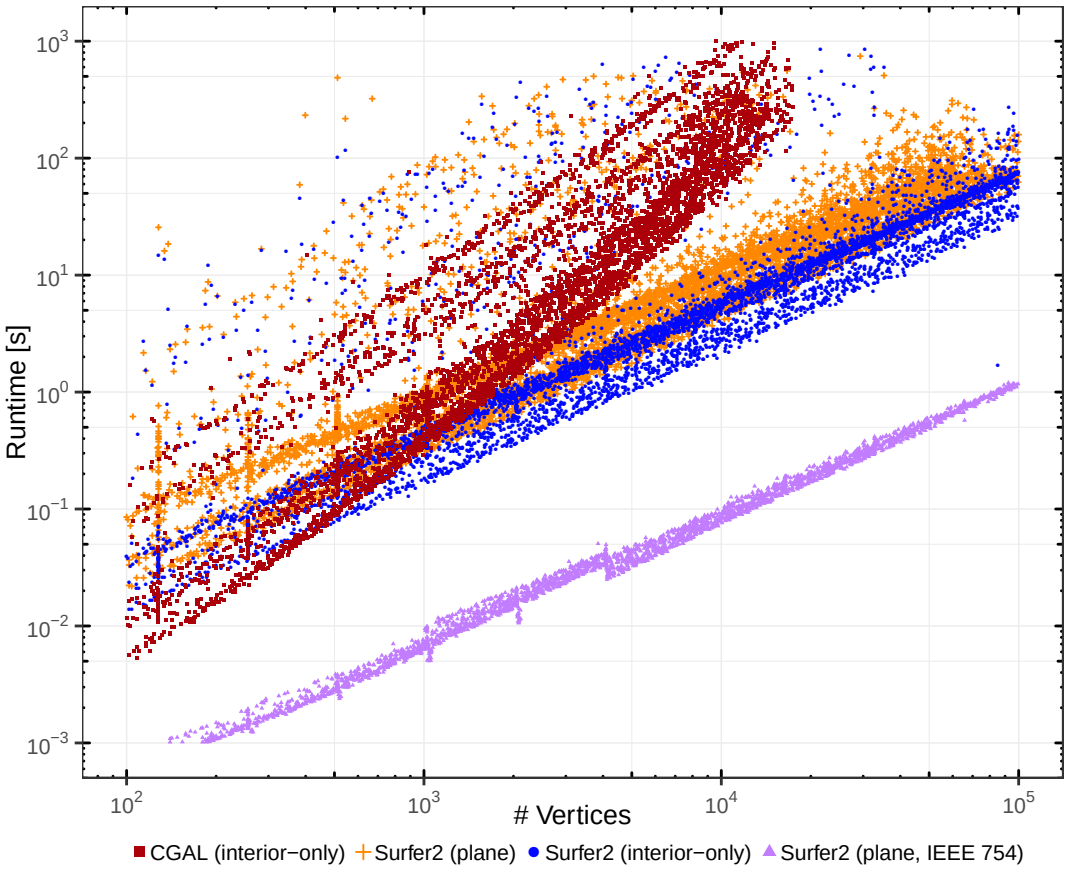


Figure 2 Runtimes of Cgal's code and different variants of Surfer2.

Runtimes and memory consumption. While Cgal's code computes the straight skeleton either in the interior or the exterior of a polygon, Surfer2 can do both in one run because it treats a polygon as a PSLG. But Surfer2 can also be restricted to just the interior or the exterior of a polygon. Hence, for the plot in Figure 2 we ran Surfer2 twice, once applied to the entire plane and once for just the interiors of the polygons that were also handled by Cgal. Our tests make it evident that Surfer2 is significantly faster than Cgal for a large fraction of the inputs. In particular, its runtime seems to exhibit an $n \log n$ growth, compared to the clearly quadratic increase of the runtime of Cgal's code. Figure 3 shows that Surfer2's memory consumption grows (mostly) linearly while Cgal's code requires a clearly quadratic amount of memory.

The results for Cgal's straight skeleton package were to be expected because (at least back in 2010) it computed potential split events for each pair of reflex vertex and wavefront edge [12, Section 2.5.4]. Indeed, tests carried out in 2010 indicated that it requires $\mathcal{O}(n^2 \log n)$ time and $\Theta(n^2)$ space for n-vertex polygons, as discussed in [13]. Our test results suggest that the same algorithm and implementation are applied in the current Cgal 5.0, which we used for our tests. The theoretical upper bound on the runtime complexity of Surfer2 is given by $O(n^3 \log n)$. Thus, its very decent practical performance is noteworthy.

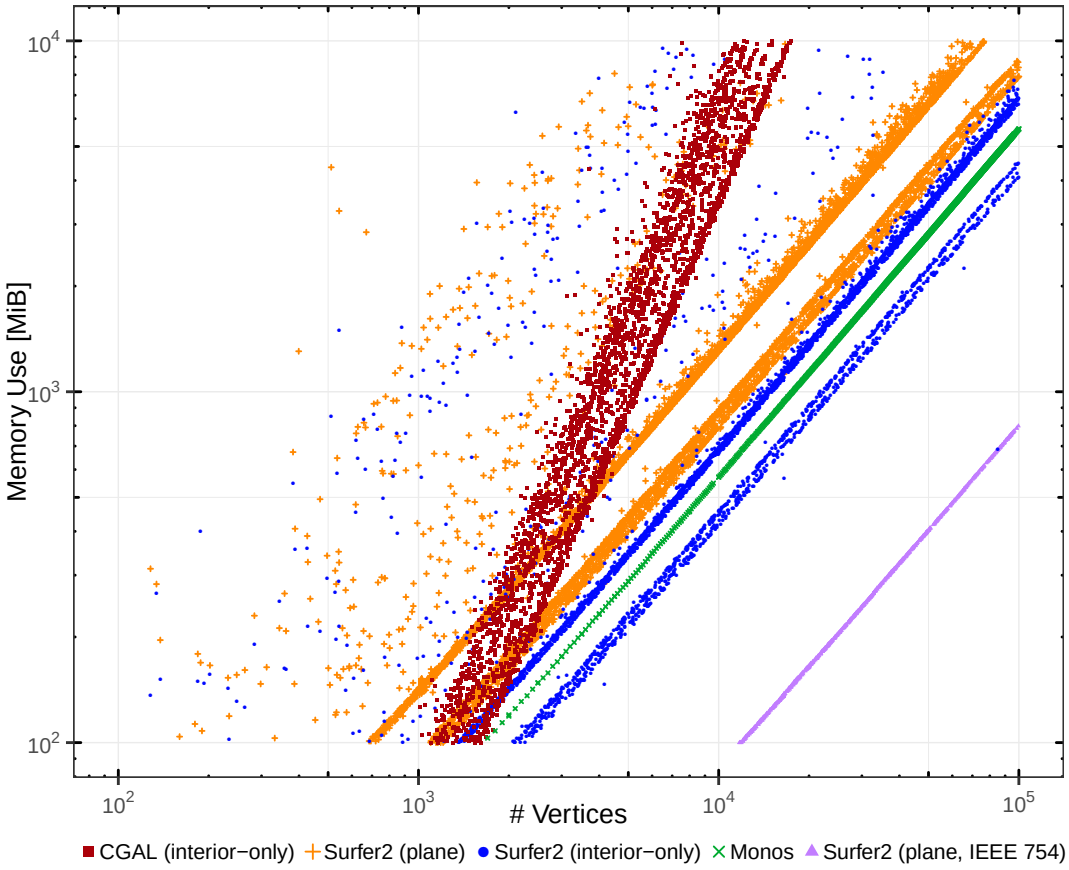


Figure 3 Memory use of CGAL, SURFER2 variants, and MONOS.

To see the runtime and memory characteristics of the practical costs of using `CORE::Expr`, we also ran SURFER2 with IEEE 754 double as a number type; cf. Figures 2 and 3.

The $\mathcal{O}(n \log n)$ bound for the complexity of MONOS is apparent in Figure 4, left. Given that MONOS is a special-purpose code designed specifically for handling monotone polygons, it had to be expected that it outperforms SURFER2 consistently.

Dependence on input characteristics. There are outliers both in the runtime as well as in the memory consumption of SURFER2 which are clearly visible in Figures 2 and 3. (To a lesser extent this noise is visible for CGAL, too.) Given the fact that such outliers do not show up for the IEEE 754-based version of SURFER2, there is reason to assume that this behavior is not intrinsic to SURFER2's algorithm but that it has its roots in the use of exact arithmetic. To probe this issue further we investigated which input classes trigger these outliers. Figure 5 shows the runtimes of both codes for three different input classes.

The class of input that was most time-consuming to handle for all implementations were our random octagonal polygons. All polygons out of this group have interior angles which are multiples of 45°. We were surprised to see that orthogonal polygons need not be troublesome per se. The key difference between both groups of polygons is given by the fact that all octagonal polygons have their vertices on an integer grid while the vertices of the orthogonal polygons are (random) real numbers. Hence, for the octagonal polygons many events tend to happen at exactly the same time when opposite edges become incident in different parts

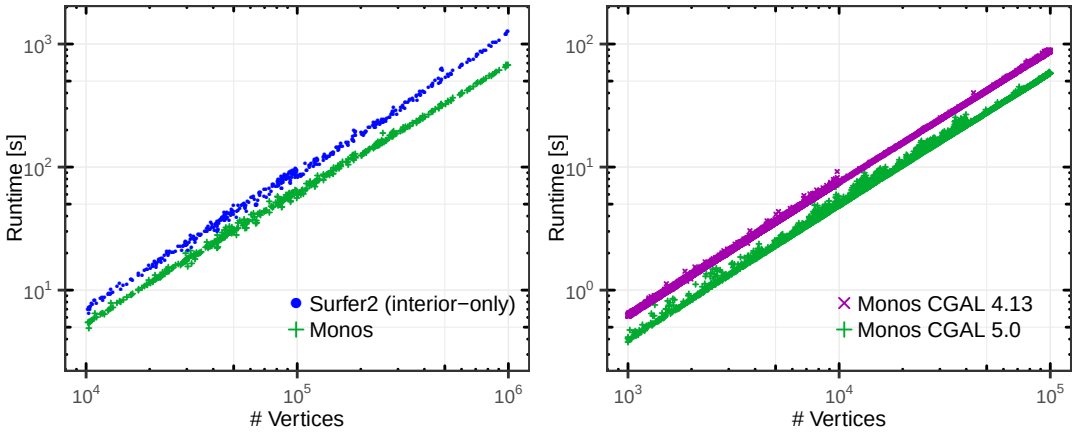


Figure 4 Left: Runtime of MONOS v. SURFER2 on large monotone inputs.
Right: Runtime of MONOS with CGAL 4 v. CGAL 5.

of the polygon. It is apparent that the proper time-wise ordering of these events incurs a significant cost if `CORE::Expr` is used. The presence of parallel edges does not automatically increase the runtime of SURFER2, though. The likely explanation is that many events are caused by opposite wavefront edges that become incident. In such a scenario, SURFER2 constructs a so-called infinitely-fast vertex. These vertices are easy to sort because they are handled immediately. For comparison purposes we ran the code on random simple polygons as a third class of input, with random vertex coordinates. Having any kind of co-temporal event is highly unlikely for those inputs, as are infinitely-fast vertices.

The excessive amounts of time consumed by ordering simultaneous events became even more apparent when we studied the impact of multiplicative weights on SURFER2's runtime. As expected, a difference in the timings for weighted and unweighted random polygons was hardly noticeable. For our randomly weighted octagonal polygons we expected reduced runtimes and fewer outliers even when using exact arithmetic, due to very few truly simultaneous events. And, indeed, this was confirmed impressively by our tests; see Figure 5.

Experiences with CGAL. While running our tests, we also compared CGAL versions 4.13 and 5.0, as the latter was released only recently. We witnessed an improvement in the performance of our codes for the newer version; see Figure 4, right.

References

1 O. Aichholzer and F. Aurenhammer. Straight Skeletons for General Polygonal Figures in the Plane. In *Voronoi's Impact on Modern Sciences II*, volume 21, pages 7–21. Institute of Mathematics of the National Academy of Sciences of Ukraine, 1998.

2 O. Aichholzer, F. Aurenhammer, D. Alberts, and B. Gärtner. A Novel Type of Skeleton for Polygons. *Journal of Universal Computer Science*, 1(12):752–761, 1995.

3 T. Auer and M. Held. Heuristics for the Generation of Random Polygons. In *Proceedings of the 8th Canadian Conference on Computational Geometry (CCCG)*, pages 38–44, 1996.

4 T. Biedl, M. Held, S. Huber, D. Kaaser, and P. Palfrader. A Simple Algorithm for Computing Positively Weighted Straight Skeletons of Monotone Polygons. *Information Processing Letters*, 115(2):243–247, 2015.

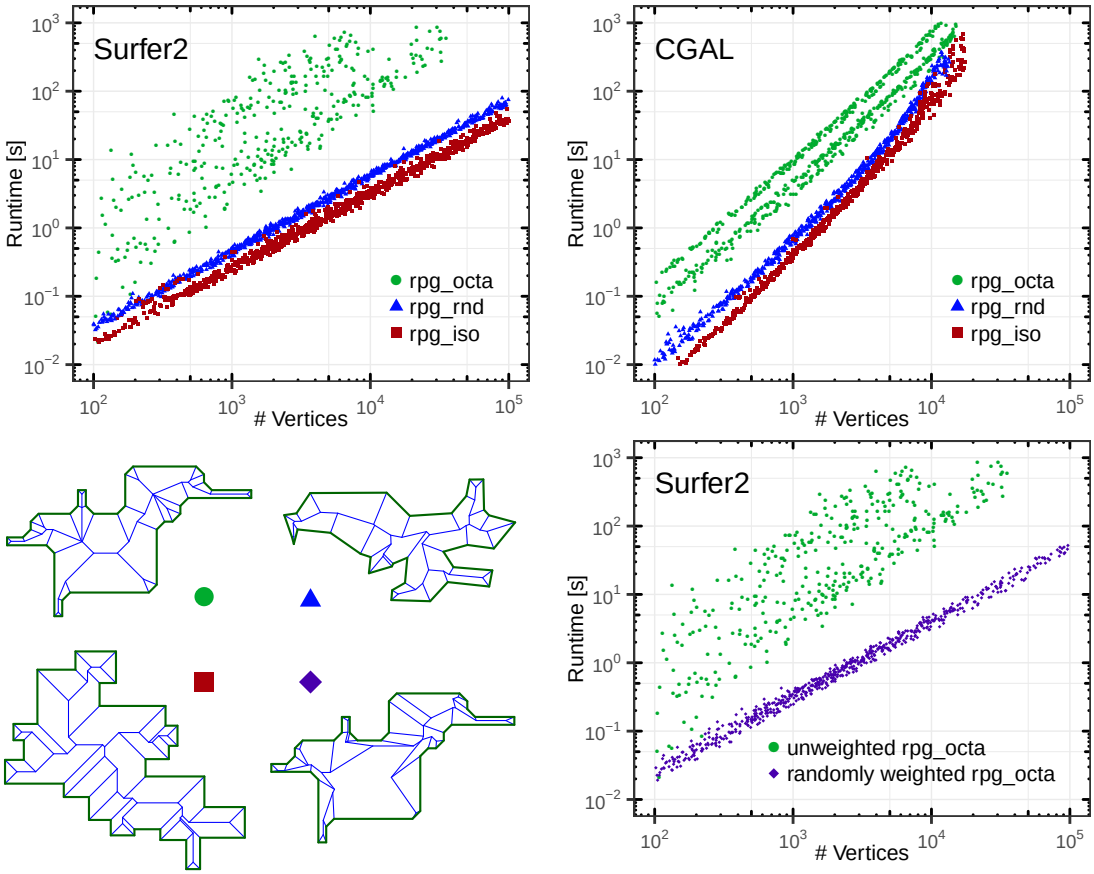


Figure 5 Top: The effect of different input classes on the runtime of SURFER2 and on CGAL. Bottom-left: Samples for different input classes (left to right, top to bottom): Octagonal input (on integer grid), random polygon, orthogonal polygon not on integer grid, and weighted octagonal input. Bottom-right: SURFER2 runtimes for unweighted and randomly weighted octagonal input.

5 F. Cacciola. 2D straight skeleton and polygon offsetting. In *CGAL User and Reference Manual*. CGAL Editorial Board, 5.0 edition, 2019.

6 S.-W. Cheng, L. Mencel, and A. Vigneron. A Faster Algorithm for Computing Straight Skeletons. *ACM Transactions on Algorithms*, 12(3):44:1–44:21, 2016.

7 G. Eder and M. Held. Computing Positively Weighted Straight Skeletons of Simple Polygons Based on Bisector Arrangement. *Information Processing Letters*, 132:28–32, 2018.

8 G. Eder, M. Held, S. Jasonarson, P. Mayer, and P. Palfrader. On Generating Polygons: Introducing the Salzburg Database. In *Proceedings of the 36th European Workshop on Computational Geometry*, pages 75:1–7, 2020.

9 D. Eppstein and J. Erickson. Raising Roofs, Crashing Cycles, and Playing Pool: Applications of a Data Structure for Finding Pairwise Interactions. *Discrete & Computational Geometry*, 22(4):569–592, 1999.

10 M. Held and P. Palfrader. Straight Skeletons with Additive and Multiplicative Weights and Their Application to the Algorithmic Generation of Roofs and Terrains. *Computer-Aided Design*, 92(1):33–41, 2017.

11 M. Held and P. Palfrader. Skeletal Structures for Modeling Generalized Chamfers and Fillets in the Presence of Complex Miters. *Computer-Aided Design and Applications*, 16(4):620–627, 2019.

12 S. Huber. *Computing Straight Skeletons and Motorcycle Graphs: Theory and Practice*. Shaker Verlag, 2012. ISBN 978-3-8440-0938-5.

13 S. Huber and M. Held. Theoretical and Practical Results on Straight Skeletons of Planar Straight-line Graphs. In *Proceedings of the 27th Symposium on Computational Geometry (SoCG)*, 2011.

14 T. Kelly and P. Wonka. Interactive Architectural Modeling with Procedural Extrusions. *ACM Transactions on Graphics*, 30(2):14:1–14:15, Apr. 2011.

15 P. Palfrader, M. Held, and S. Huber. On Computing Straight Skeletons by Means of Kinetic Triangulations. In *Proceedings of the 20th Annual European Symposium on Algorithms (ESA)*, pages 766–777, 2012.

16 The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 5.0 edition, 2019.

17 A. Vigneron and L. Yan. A Faster Algorithm for Computing Motorcycle Graphs. *Discrete & Computational Geometry*, 52(3):492–514, 2014.

Finding an Induced Subtree in an Intersection Graph is often hard*

Hidefumi Hiraishi[1], Dejun Mao[2], and Patrick Schnider[3]

1 **Graduate School of Information Science and Technology, The University of Tokyo**
`hiraishi1729@is.s.u-tokyo.ac.jp`
2 **Graduate School of Information Science and Technology, The University of Tokyo**
`maodejun001@is.s.u-tokyo.ac.jp`
3 **Department of Computer Science, ETH Zürich**
`patrick.schnider@inf.ethz.ch`

Abstract

We prove that the induced subtree isomorphism problem is NP-complete for penny graphs and chordal graphs as text graphs. As a step in the proofs, we reprove that the problem is NP-complete if the text graph is planar. For many other graph classes NP-completeness follows, as they contain one of the above three classes as a subclass, e.g., segment intersection graphs and coin graphs (contain planar graphs) or unit disk graphs (contain penny graphs).

1 Introduction

The Subtree isomorphism problem (STI) takes as input a graph G, called the *text graph*, and a tree T, called the *pattern graph*. The task is to determine whether G contains a copy of T, that is, a subgraph that is isomorphic to T. Clearly, this problem is in NP. It follows from the NP-completeness of Hamiltonian path that STI is NP-complete for all graph classes for which Hamiltonian path is hard. This includes many graph classes that we will talk about in this work, such as planar graphs, unit disk graphs and chordal graphs [2]. In some of the literature, such as [7, 9], STI is actually restricted to the case where G is a tree. In this case, STI allows a polynomial algorithm.

In contrast to STI, the *Induced Subtree isomorphism* (ISTI) asks whether a text graph G contains an *induced* copy of T. While this modification might seem rather minor at first glance, it can actually change the problem quite significantly. For example, on interval graphs, STI is NP-complete, whereas ISTI is tractable [4]. On the other hand, NP-completeness can still follow from the NP-completeness of Hamiltonian path, but only after subdividing each edge with an additional vertex (see e.g. [6]). The question then reduces to finding a path of length $2n - 1$ in the new graph, which can in turn be reduced to Hamiltonian path. However, this reduction only works for graph classes that are closed under subdivision of edges. This includes planar graphs, but not penny graphs or chordal graphs.

In this abstract, we will show that ISTI is still hard for the two latter classes. For chordal graphs, this answers a question by Heggernes, van't Hof and Milanič [4]. We will also give an alternative proof that ISTI is hard for planar graphs. While our proof is arguably more complicated than the argument above, it is the basis for the other reductions. For many other classes of intersection graphs, NP-completeness of ISTI follows from our results.

* This work was initiated when the third author was visiting the University of Tokyo.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

In general, an intersection graph is a graph whose vertices are subsets of some ground set and two vertices share an edge if and only if the corresponding sets intersect. The sets can be geometric objects such as straight-line segments in the plane, giving rise to *segment intersection graphs*, or disks in the plane, giving rise to *disk graphs*. For disk graphs, we can also enforce that any two disks have disjoint interiors (*coin graphs*), that all disks have the same radius (*unit disk graphs*), or both of these conditions (*penny graphs*). On the other hand, the sets can also be of a combinatorial nature. For example, the vertices could be subtrees of a given tree and two vertices share an edge if the corresponding subtrees share at least one vertex. These subtree intersection graphs are called *chordal graphs*. Another definition for chordal graphs is that every cycle of length at least 4 needs to have a chord, i.e., an edge that is not part of the cycle but connects two vertices of the cycle. These two definitions were shown to be equivalent by Gavril [3].

2 Planar graphs

We will use a reduction from a variant of 3-SAT, called Clause-Linked Planar 3-SAT. Given a CNF formula ϕ with clause set C and variable set V, the *incidence graph* $G_\phi = (C \cup V, E)$ is the graph that contains an edge between a variable and a clause if and only if the variable or its negation appear as a literal in the clause. We say that ϕ is *planar* if G_ϕ is a planar graph. The problem Planar 3-SAT is 3-SAT restricted to planar formulas. Planar 3-SAT is NP-complete [5]. We can enforce even more conditions without making the problem tractable: we say that a planar 3-CNF formula ϕ is *clause-linked* if there exists a path P (without additional vertices) connecting the clauses in $G(\phi)$ such that $G(\phi) \cup P$ is still a planar graph. Clause-Linked Planar 3-SAT, which is 3-SAT restricted to clause-linked planar formulas, is still NP-complete, see for example [8].

▶ **Theorem 2.1.** Induced Subtree Isomorphism *is NP-complete even when restricted to planar graphs.*

We will only describe the construction of the gadgets, the correctness is rather straightforward.

Proof. It is straight-forward to see that ISTI is in NP, so the rest of the proof will be dedicated to prove NP-hardness. We will do this by reduction from Clause-Linked Planar 3-SAT. For any clause-linked planar 3-SAT formula ϕ we construct a planar graph H with the property that there is an assignment satisfying ϕ if and only if H contains an induced copy of a certain tree.

Let ϕ be a clause-linked planar 3-SAT formula and let $G(\phi)$ be its associated graph and P the path through its clauses. Consider a plane drawing of $G(\phi) \cup P$. We will mimic the formula ϕ by constructing subgraphs, called *gadgets*, that serve as variables, negations and clauses, and concatenating them according to the drawing of the graph $G(\phi)$. We start with the construction of the variable and negation gadgets.

Let v be a variable with k occurrences $(v, C_1), \ldots, (v, C_k)$ in ϕ, where the order of the occurrences is according to the rotational order of the respective edges in the drawing of $G(\phi) \cup P$. For an illustration of the construction, see Figure 1. We first construct a path $W = (w_1, w_2, \ldots, w_{2k})$ of length $2k$. We then add k vertices $x_1, \ldots, x_k$ and connect each x_i to w_{2i-1} and w_{2i}. Consider the star on 5 vertices (4 leafs). We call this star the *variable signature*. For each x_i, construct a copy of the variable signature and connect one of its leafs to x_i. Finally, extend a path $p_i = (w_i, p_{i,1}, \ldots, p_{i,l_1}, \ldots, p_{i,l})$ of length $l = l_1 + l_2$ from each w_i, where l_1 and l_2 are some large enough numbers (that are still polynomial in the number

of variables), which we will define later. We call the vertices $p_{2i-1,l}$ and $p_{2i,l}$ the *endpoints* of (v, C_i). For every pair of such paths p_{2i-1}, p_{2i}, add two vertices a_i, b_i. If (v, C_i) is a positive occurrence, connect both a_i and b_i to p_{2i-1,l_1}, otherwise connect them to p_{2i,l_1}. We call such two vertices an *edge signature*.

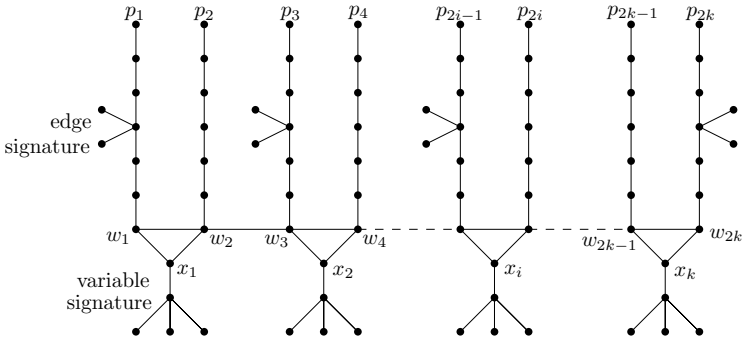


Figure 1 A variable gadget. The variable is negated in the clause C_k.

The idea behind this construction is that we will choose for each variable all the variable signatures and either the paths $p_1, p_3, \ldots, p_{2k-1}$ or the paths $p_2, p_4, \ldots, p_{2k}$ in the induced subtree. Also choosing the edge signature for a path will then mean that the variable is set to true. Note that every variable gadget admits a plane drawing in a disk of some radius such that only the paths p_i leave the disk and they do so according to the rotational order of the respective edges in the drawing of $G(\phi) \cup P$.

We will now construct the clause gadgets. For an illustration see Figure 2. Let $C_1, \ldots, C_m$ be the clauses in the order in which they appear along P. For every clause C_i, draw a vertex c_i and connect them with a path $c_1, \ldots, c_m$. For some clause C_i, let v_1, v_2 and v_3 be the variables occurring in C_i. Connect c_i to the endpoints of (v_1, C_i), (v_2, C_i) and (v_3, C_i). Again, this gadget admits a plane drawing that agrees with the rotational order around each C_i in the drawing of $G(\phi) \cup P$.

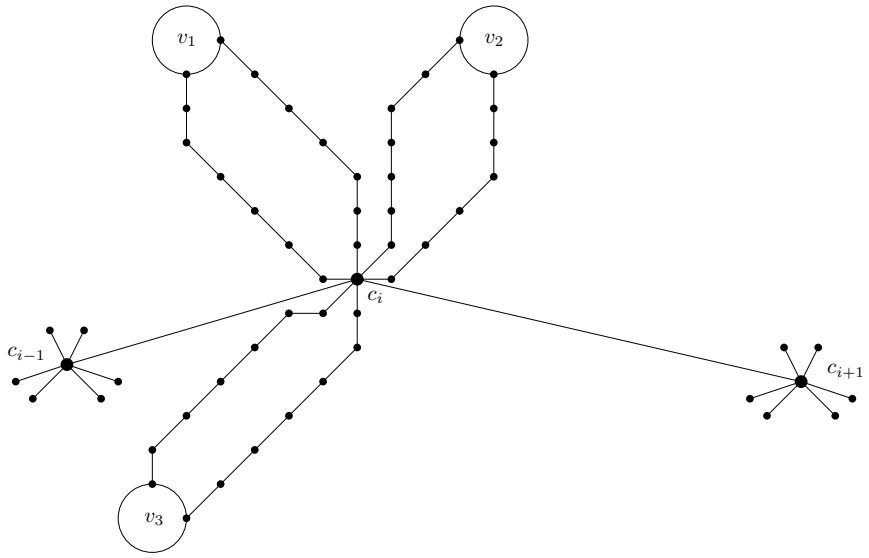


Figure 2 A clause gadget.

The union of all the gadgets will be the text graph H. As every gadget admits a planar drawing that agrees with the original rotational order, H admits a plane drawing. See Figure 3 for an example of the whole construction. We will now define our pattern graph T. Start with a path M of length m (the number of clauses). From each vertex of M, extend three paths of length l and connect each of them to a variable signature and exactly one of them to an edge signature.

It now follows from the construction that H contains an induced copy of T if and only if ϕ is satisfiable.

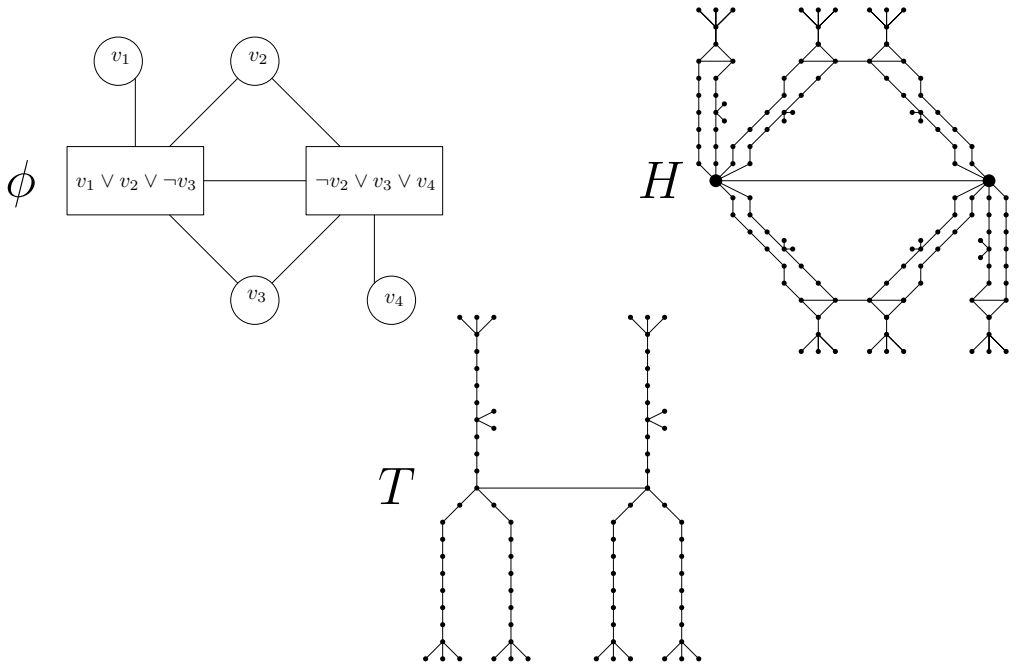


Figure 3 A drawing of a planar formula ϕ with the corresponding text graph H and pattern graph T.

◀

▶ **Corollary 2.2.** Induced Subtree Isomorphism *is NP-complete even when restricted to coin graphs or segment intersection graphs.*

Proof. The Koebe-Andreev-Thurston circle packing theorem states that every planar graph is a coin graph (and vice versa). Further, every planar graph is a segment intersection graph [1]. ◀

3 Penny graphs

In this section we will modify our reduction to fit penny graphs. It is again straightforward to check that ISTI is in NP, so we will only show NP-hardness. Again, we will only describe the construction of the gadgets, as all other arguments are analogous to the case of planar graphs.

▶ **Theorem 3.1.** Induced Subtree Isomorphism *is NP-complete even when restricted to penny graphs.*

Proof. We first show that we can draw unit disks with disjoint interiors such that their induced embedded penny graph coincides with the drawing of a variable gadget as constructed above. For an illustration of the construction see Figure 4. Place the centers of the disks the path W on a circle with very large radius, i.e., the disks will only cover a very small part of the circle. The disks corresponding to the x_i's and the variable signatures can then be placed inside the circle without any intersections. Further, the paths p_i can be placed outside the circle, and choosing l large enough, we can make enough space for the edge signatures. As for the clause gadgets, note that in our original construction, each vertex c_i is the center of an induced star on 9 vertices (8 leafs). However, the largest induced star induced star in a penny graph can have 6 vertices (5 leafs). So, instead of connecting the endpoints of p_{2i-1} and p_{2i} directly to c_i, we will connect them to a new vertex d and then connect this vertex to c_i by a path of some length l_3. Similarly, we will also connect c_i to c_{i+1} by a path of length l_4. Here, l_3 and l_4 are chosen large enough to have enough room the place the corresponding unit disks without unwanted intersections. Defining the pattern graph H accordingly, it is straight-forward to check that all the above arguments still go through.

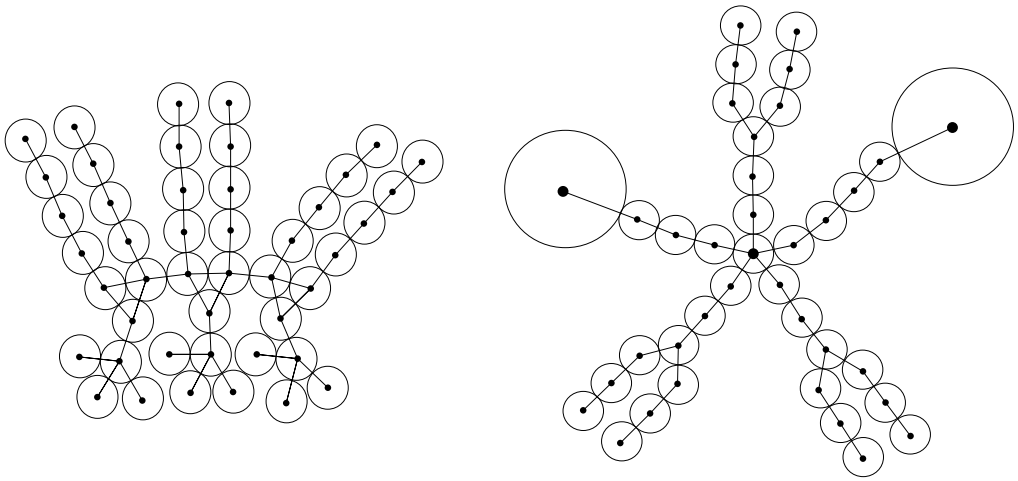


Figure 4 A variable gadget (left) and a clause gadget (right) for penny graphs.

◀

Clearly, every penny graph is a unit disk graph, so we immediately get the following corollary:

▶ **Corollary 3.2.** Induced Subtree Isomorphism *is NP-complete even when restricted to unit disk graphs.*

4 Chordal graphs

In this section we will modify our reduction to fit chordal graphs. It is again straightforward to check that ISTI is in NP, so we will only show NP-hardness. We will again only change the gadgets slightly, but we will include a large number of additional edges to make the graph chordal. As above, we will only describe the modifications in the construction.

▶ **Theorem 4.1.** Induced Subtree Isomorphism *is NP-complete even when restricted to chordal graphs.*

Proof. We make the two following modifications to the variable gadgets for planar graphs: first, we place an additional vertex y_i on the edge between w_i and w_{i+1}. So, the path W is now $w_1, y_1, w_2, y_2, \ldots, w_{2k-1}, y_{2k-1}, w_{2k}$. Secondly, instead of placing an edge signature at one place on a path p_i, we place one at every vertex. We further add an additional vertex for every vertex on some p_i, connecting it to only this vertex. We call these vertices *edge leafs*. The clause gadgets we extend by constructing a set of k paths of length 3 for each clause and connecting each of them to the respective c_i. We call these k 3-paths *clause signatures*.

Now, we will add chords to all cycles in our current graph. There are two types of cycles: (i) cycles defined by p_{2i-1} and p_{2i} for some variable and (ii) cycles inherited from cycles in $G(\phi) \cup P$. For (i), we add a complete bipartite graph between the vertices of p_{2i-1} and p_{2i}. Further, we connect all vertices of p_{2i-1} and p_{2i} to y_{2i-1}. As every such cycle must contain vertices from both p_{2i-1} and p_{2i} and possibly y_i, any such cycle will now have a chord. Similarly we add a complete bipartite graph between the vertices of p_{2i} and p_{2i+1} and connect all vertices of p_{2i} and p_{2i+1} to y_{2i}. This does not destroy any existing cycles, but it also does not introduce any chordless cycles. These edges will be helpful for the correctness proof, as they ensure that not both p_{2i} and p_{2i+1} can be chosen.

For (ii), recall that every edge in $G(\phi)$ corresponds to some pair of paths p_{2i-1} and p_{2i} for some variable. Further recall that all edges of P correspond to single edges in our constructed graph. Thus, every cycle in $G(\phi) \cup P$ that uses t edges of $G(\phi)$ induces 2^t cycles in our constructed graph. Further, each of these cycles in the constructed graphs will go through some vertices y_j and paths p_i. For every p_i and every y_j, we connect all vertices on p_i to y_j. Again, this puts a chord in every cycle.

Denote by H the chordal graph that we get from the above construction. We will now define our pattern graph T, which will be very similar to the pattern graph in the reduction for planar graphs. Start with a path M of length m (the number of clauses). From each vertex of M, connect it to a clause signature and three paths of length l. Add an edge leaf to each vertex on these paths. Finally, connect each of the paths to a variable signature and for exactly one path per clause, connect each vertex on the path to an edge signature.

Similar to above it can now be shown that H contains an induced copy of T if and only if ϕ is satisfiable.

We claim that H contains an induced copy of T if and only if ϕ is satisfiable. Finding a copy of T given a satisfying assignment of ϕ is analogous to the proof for planar graphs. For the other direction, we will again show that every induced copy of T corresponds to a satisfying assignment of ϕ. For this, we first note that T contains $3m$ disjoint paths of length l with edge leafs, each connected to a variable signature. On the other hand, H contains $6m$ paths of length l with edge leafs, always two of which (p_{2i-1} and p_{2i} for every variable) are connected to the same variable signature. In particular, every induced copy of T must contain either p_{2i-1} or p_{2i} and it cannot contain both. Further, due to the complete bipartite graph between p_{2i} and p_{2i+1}, T cannot contain both of them. Thus, T either contains $p_1, p_3, \ldots, p_{2k-1}$ or $p_2, p_4, \ldots, p_{2k}$. Again, we set the corresponding variable to TRUE in the first case, and to FALSE in the second case. Note that each the clause signatures ensure that c_i is the endpoint of three such paths and necessarily in T. By construction, each of these paths is connected to an edge signature if and only if the corresponding literal on the connected clause is positive. In particular, each c_i being incident to a path connected to an edge signature implies that under this assignment, each clause contains at least one positive literal, i.e., ϕ is satisfiable, which concludes the proof.

◀

References

1 Jérémie Chalopin and Daniel Gonçalves. Every planar graph is the intersection graph of segments in the plane. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 631–638. ACM, 2009.

2 H. N. de Ridder et al. Information system on graph classes and their inclusions. `www.graphclasses.org`, 2016.

3 Fănică Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B*, 16(1):47 – 56, 1974. URL: `http://www.sciencedirect.com/science/article/pii/009589567490094X`, `doi:https://doi.org/10.1016/0095-8956(74)90094-X`.

4 Pinar Heggernes, Pim van't Hof, and Martin Milanič. Induced subtrees in interval graphs. In *International Workshop on Combinatorial Algorithms*, pages 230–243. Springer, 2013.

5 D. Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982. `doi:10.1137/0211025`.

6 Jiří Matoušek and Robin Thomas. On the complexity of finding iso-and other morphisms for partial k-trees. *Discrete Mathematics*, 108(1-3):343–364, 1992.

7 David W. Matula. Subtree isomorphism in o(n5/2). In B. Alspach, P. Hell, and D.J. Miller, editors, *Algorithmic Aspects of Combinatorics*, volume 2 of *Annals of Discrete Mathematics*, pages 91 – 106. Elsevier, 1978. URL: `http://www.sciencedirect.com/science/article/pii/S0167506008703248`, `doi:https://doi.org/10.1016/S0167-5060(08)70324-8`.

8 A. Pilz. Planar 3-SAT with a clause/variable cycle. *CoRR*, abs/1710.07476, 2017. URL: `http://arxiv.org/abs/1710.07476`, `arXiv:1710.07476`.

9 Ron Shamir and Dekel Tsur. Faster subtree isomorphism. *Journal of Algorithms*, 33(2):267 – 280, 1999. URL: `http://www.sciencedirect.com/science/article/pii/S0196677499910441`, `doi:https://doi.org/10.1006/jagm.1999.1044`.

Scaling and compressing melodies using geometric similarity measures

L. E. Caraballo[1], J.M. Díaz-Báñez[1], F. Rodríguez[1], V. Sánchez-Canales[1], and I. Ventura[1]

1 **University of Seville**
lcaraballo@us.es,dbanez@us.es,fabrodsan@us.es,vscanales@us.es,iventura@us.es

Abstract

Melodic similarity measurement is of key importance in Music Information Retrieval. In this paper, we use geometric matching techniques to measure the similarity between two monophonic melodies. We propose efficient algorithms for optimization problems inspired in two operations on melodies: scaling and compressing. In the scaling problem, an incoming query melody is scaled forward until the similarity measure between the query and the reference melody is minimized. The compressing problem asks for a subset of notes of a given melody so that the matching cost between the selected notes and the reference melody is minimized.

1 Introduction

Musicological and computational studies on rhythmic and melodic similarity have given rise to a number of geometric problems [6]. A melody can be codified as a consecutive sequence of musical notes and each note can be represented by a point (point-representation) [3] or a horizontal segment (segment-representation) in a time pitch value [5]. In this paper we study two problems that arise in Musical Information Retrieval (MIR): scaling and melody compressing. Scaling is used for tempo variation [4] and compression for clustering [2]. Given a reference melody, a query melody and a similarity measure, in the *scaling problem* the incoming query is scaled forward in the horizontal direction to find the minimum similarity measure between it and the reference melody. The *compressing problem* of a given melody asks to select k notes of such melody so that the similarity measure between the reference and the simplified melody is minimized. The study of measures for melodic similarity is of key importance for a music retrieval system. Techniques based on geometric similarity measures have been used in the literature. In [1], the following geometric matching technique for music similarity measurement was proposed: Each note is represented as a horizontal line segment so that a sequence of notes can be described as a rectangular contour in a 2D coordinate system, in which the horizontal and vertical coordinates correspond to time and pitch value, respectively. Then, the used similarity measure between two melodies is the minimum area between them. They solve the problem of searching a query melodic segment of length m into a reference melody of length n in $O(nm \log n)$ time. The optimal matching is obtained by moving from left to right and bottom to up the query segment until the matched area is minimized.

This research has been supported by the project GALGO (Spanish Ministry of Economy and Competitiveness, MTM2016-76272-R AEI/FEDER,UE) and the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 734922.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

1.1 Problems statement

Let $R = (R_1, R_2, ..., R_n)$ and $Q = (Q_1, Q_2, ..., Q_m)$ be sequences representing two melodies with $m < n$. R is the reference melody from the data set and Q is the query melody to be matched. In the point-representation, we assume that the points representing the notes are just the middle points of the horizontal segments in the corresponding segment-representation. We also call melodic contour to a segment-representation.

In the following, we introduce two geometric measures to compute similarity and two operations on a melody that are the main ingredients of the problems.

Area: The region between two melodies with the same duration in the segment-representation can be partitioned into rectangles with vertical edges supported by vertical straight lines passing through the ending points of the segments. The area between two melodies is defined as the sum of the areas of the rectangular regions of the partition. See Figure 1a).

t-Monotone Matching: Let $R = \{R_i = (x_i, p_i), i = 1, \cdots, n\}$ and $Q = \{Q_j = (t_j, q_j), j = 1, \cdots, m\}$ be two melodies within the point-representation. In a t-monotone matching between R and Q, we map each point of R to its nearest t-coordinate point in Q, that is, the left- or the right-point in time. The unmatched points of Q are associated to their t-coordinate nearest in R. See Figure 1b). Note that this matching is designed for music matching and does not use the Euclidean distance. The cost of the note Q_j, $\phi(Q_j)$, is given by the sum of the l_1-distances between Q_j and its matched points in R and the total cost of the matching is

$$\phi(R, Q) = \sum_{Q_j \in Q} \phi(Q_j).$$

ε-scaling: Consider a segment-representation of two melodies R and Q. Let $X = (x_0 = 0, x_1, x_2, ..., x_n)$ and $T = (t_0 = 0, t_1, t_2, ..., t_m)$ be the partitions on time given by the R and Q, respectively. Given $\varepsilon > 0$, we define the *ε-scaling* on the query Q, $S_\varepsilon(Q)$, as the operation of increasing by ε the length of each segment of Q but keeping static the starting point of Q. Thus, after an ε-scaling, X does not change and T is transformed to $T + \varepsilon = (t_0, t_1 + \varepsilon, t_2 + 2\varepsilon, t_3 + 3\varepsilon, \cdots, t_m + m\varepsilon)$. The query can be scaled forward until the two melodies have the same time duration. Thus, $0 \leq \varepsilon \leq \frac{x_n - t_m}{m}$. Note that the pitch of the notes are unchanged in the scaling operation. Now, for a point-representation of R and Q, $R = \{(x_i, p_i), i = 1, \cdots, n\}$ and $Q = \{(t_j, q_j), j = 1, \cdots, m\}$, the *$\varepsilon$-scaling* of Q is given by $S_\varepsilon(Q) = \{(t_1 + \frac{\varepsilon}{2}, q_1), (t_2 + \varepsilon, q_2), \cdots, (t_m + \frac{m}{2}\varepsilon, q_m))\}$.

k-compressed melody: Given a melody R with n notes in the segment-representation, a k-compressed melody, Q_k, is a melody composed by k segments such that Q_k and R have the same duration and each segment of Q_k contains at least a segment of R. Figure 2b). For the point-representation, a k-compressed melody Q_k is a subset of k notes of R. Figure 2a).

In this paper we study the following optimization problems:

▶ Problem 1.1. Given two melodies R and Q in the segment-representation, compute the value of $\varepsilon > 0$ such that the area between R and $S_\varepsilon(Q)$ is minimized.

▶ Problem 1.2. Given two melodies R and Q in the point-representation, compute the value of $\varepsilon > 0$ such that the cost of the t-monotone matching between R and $S_\varepsilon(Q)$ is minimized.

▶ Problem 1.3. Given a melody R in the point-representation, compute a k-compressed melody Q_k so that the cost of the t-monotone matching between R and Q_k is minimized.

▶ Problem 1.4. Given a melody R with n notes in the segment-representation, compute a k-compressed melody Q_k so that the area between R and Q_k is minimized.

Note that in the scaling problems, the reference melody is fixed whereas the query melody is dynamic. However, the compressing problems ask for an optimal selection of the notes in the reference melody. Figures 1 and 2 illustrate the problems.

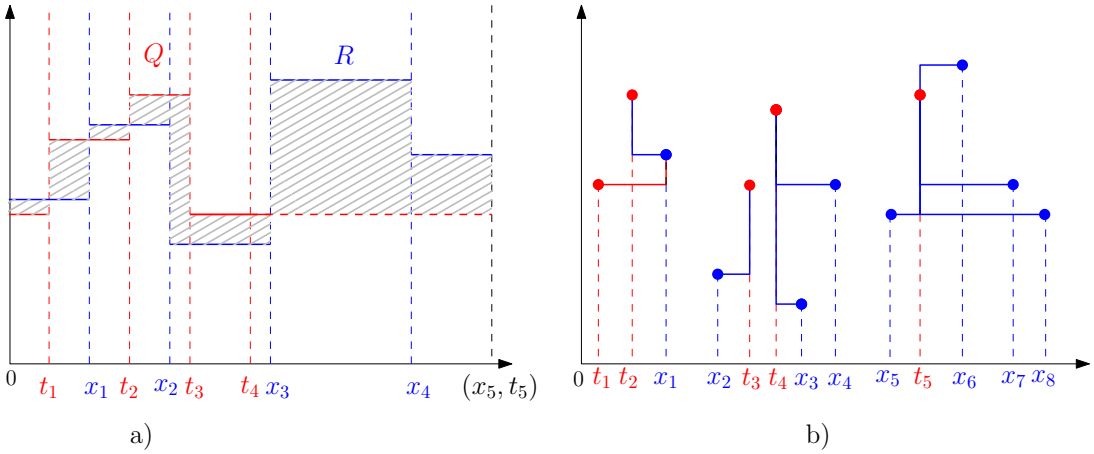


Figure 1 a) Area between the reference R and the query Q, Problem 1.1. b) t-Monotone matching between the reference R (blue) and the query Q (red), Problem 1.2.

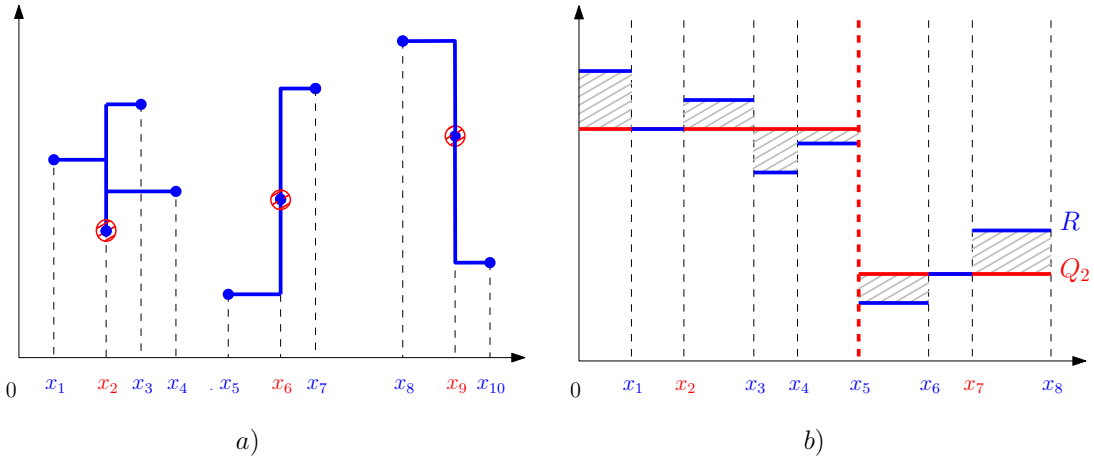


Figure 2 a) A 3-compressed melody from an input of 10 notes, Problem 1.3. b) Area between a melody R and a 2-compressed melody Q_2, Problem 1.4.

2 Scaling

2.1 Area as similarity measure

We assume that the last segment of the scaled query $S_\varepsilon(Q)$ is extended so that the duration of R and $S_\varepsilon(Q)$ is the same. Thus, the area between the melody R and $S_\varepsilon(Q)$ is the sum of $O(m+n)$ rectangles as illustrated in Fig 1a). Denote by $A_{RQ}(\varepsilon)$ the area between R and $S_\varepsilon(Q)$ as a function of ε. Observe that after an ε-scaling of the query for a big enough ε, at least one pair of vertical edges coincide, that is, $x_i = t_j$ for some i, j. At this instant, a rectangle disappears and after that, a new rectangle appears. We call this value of ε an *event*.

Also note that between two events, the area of some rectangles increase, others decrease and others are unaffected. In fact, the type of a rectangle can be determined by its vertical edges. Rectangles can be classified in four types: Type C_0, with vertical edges $[x_i, x_{i+1}]$; type C_1, with vertical edges $[t_j, t_{j+1}]$; type C_2, with vertical edges $[x_i, t_j]$ and type C_3, with vertical edges $[t_j, x_i]$. Figure 3 illustrates an event in which a rectangle of type C_3 disappears.

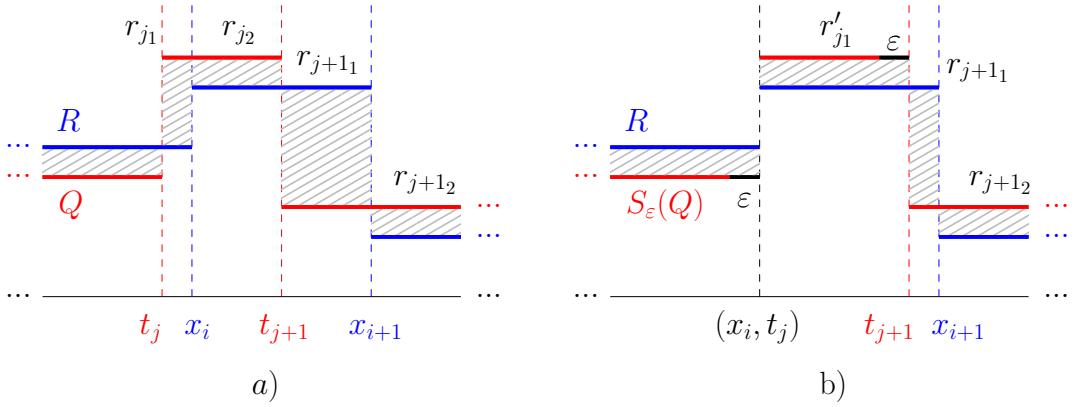


Figure 3 a) Rectangles between R and Q. b) After the scaling $S_\varepsilon(Q)$, an event is found when t_j touches x_i. Rectangle r_{j_1} disappears and r_{j_2} is renamed as r'_{j_1}.

The following result allows us to discretize the problem:

▶ **Lemma 2.1.** *$A_{RQ}(\varepsilon)$ is a piecewise linear function.*

As a consequence of Lemma 2.1, the minimum area is reached at the events and our approach is to efficiently reevaluate the area in each event and take the minimum value.

▶ **Lemma 2.2.** *$A_{RQ}(\varepsilon)$ can be updated between two consecutive events in $O(1)$ time.*

▶ **Theorem 2.3.** *Problem 1.1 can be solved in $O(nm \log m)$ time.*

2.2 t-Monotone matching as similarity measure

Let $R = \{(x_i, p_i), i = 1, \cdots, n\}$ and $Q = \{(t_j, q_j), j = 1, \cdots, m\}$ be two melodies in the point-representation. The t-Monotone matching between R and Q generates two sets of pairs of points. Let P_1(resp. P_2) be the set of pairs such that $x_i \leq t_j$ (resp. $x_i > t_j$). Observe that for a small $\varepsilon > 0$, after the scaling $S_\varepsilon(Q)$ of Q, the costs related to pairs in P_1 (resp. P_2) increase (decrease). The sets P_1 and P_2 are dynamics in the scaling operation, that is, t_j is moving from left to right but x_i is static. For the sake of simplicity, for any ε in the scaling process, we use (x_i, t_j) or (i, j) to refer the pair with abscissas x_i and t_j. We call events to the values of $\varepsilon > 0$ where the pair (i, j) can change. We have three type of events:

1. The instant at which $x_i = t_j$: the pair (i, j) changes from P_2 to P_1.
2. The instant at which the bisector of t_j and t_{j+1} passes through x_i: the pair $(i, j+1)$ in P_2 becomes the pair (i, j) in P_1.
3. The instant at which t_j passes trough the bisector of x_i and x_{i+1}: the pair (i, j) in P_1 becomes the pair $(i+1, j)$ in P_2.

▶ **Lemma 2.4.** *The cost $\phi(R, S_\varepsilon(Q))$ is a linear function between two consecutive events.*

▶ **Lemma 2.5.** *The cost $\phi(R, S_\varepsilon(Q))$ can be updated between two consecutive events in $O(1)$ time.*

Using Lemma 2.4 and Lemma 2.5, the problem can be solved by efficiently updating the local optimum values reached at the events and we can prove:

▶ **Theorem 2.6.** *Problem 1.2 can be solved in* $O(nm\log(n+m))$ *time.*

3 Compressing

3.1 t-Monotone matching as similarity measure

Let $R = \{R_i = (x_i, p_i), i = 1, \cdots, n\}$ be a sequence of notes according the point-representation and $k \in \{1, \cdots, n\}$. Let $C_k = \{c_1, \ldots, c_k\} \subseteq R$ be a k-set of R. Consider a t-monotone matching between R and C_k and set $\phi(R, C_k) = \sum_{c_i \in C_k} \phi(c_i)$.

▶ **Definition 3.1.** Let $C_k = \{c_1, \ldots, c_k\}$ be a k-set of R.

1. Set $\overleftarrow{R} = \{R_i \in R : x(R_i) \leq x(c_k)\}$. The *left partial evaluation* of C_k is defined as:
$$\overleftarrow{\phi}(R, C_k) = \phi(\overleftarrow{R}, C_k).$$
2. We say that C_k is *left optimal* if $\overleftarrow{\phi}(C_k) \leq \overleftarrow{\phi}(C'_k)$ for all $C'_k = \{c'_1, \ldots, c'_k\}$ where $c'_k = c_k$.
3. The j-prefix of C_k, is the j-set formed by the first j points of C_k.

The following result can be easily proven by contradiction:

▶ **Lemma 3.2.** *Let* $C^*_k = \{c^*_1, \ldots, c^*_k\}$ *be an optimal* k*-set of* S*. Then, for all* $1 \leq j \leq k$ *the* j*-prefix of* C^*_k *is left optimal.*

▶ **Corollary 3.3.** *Let* $C_j = \{c_1, \ldots, c_j\}$ *be a* j*-set which is left optimal. Then every prefix of* C_j *is also left optimal.*

Above properties allow us to solve the problem with dynamic programming. A sketch of the idea is as follows. Let p_i and $p_{i'}$ be two consecutive points in a k-set. Let $W_{ii'}$ be the cost of the t-monotone matching for points between p_i and $p_{i'}$. Then, for a k-set $C_k = \{p_{i_1}, \ldots, p_{i_k}\}$, where $1 \leq i_j \leq k$ and $i_j < i_{j+1}$, the total cost $\phi(C_k)$ can be rewritten as $W_{0,i_1} + W_{i_1,i_2} + \cdots + W_{i_k,(n+1)}$, where W_{0,i_1} (resp. $W_{i_k,(n+1)}$) denotes the assignment cost of the points to the left (resp. right) of p_{i_1} (resp. p_{i_k}).

Now, assume that we have preprocessed the values $W_{ii'}$, for all $0 \leq i < i' \leq n+1$. Consider the tables $C[i,j]$ and $P[i,j]$ whose keys i and j are integers in the intervals $[1,n]$ and $[1,k]$. The cell $C[i,j]$ stores the cost of the left optimal j-set C_j that ends using p_i as the j-th point of the subset. The cell $P[i,j]$ stores the index $i' < i$ of the $(j-1)$-th point of C_j.

▶ **Theorem 3.4.** *Assuming that the values* $W_{i',i}$ *are already known, the optimum* k*-set can be computed in* $O(kn^2)$ *time.*

▶ **Lemma 3.5.** *We can compute all the values* $W_{i',i}$ *in* $O(n^2)$ *time.*

▶ **Theorem 3.6.** *Problem 1.3 can be solved in* $O(kn^2)$ *time.*

3.2 Area as similarity measure

Given a melodic contour R with n notes, we want to find a k-compression Q_k of R that minimizes the area between R and Q_k. Let $X = (x_0 = 0, \ldots, x_n)$ and $T = (t_0 = 0, \ldots, t_k)$ be the partitions on the time interval $[0, x_n = t_k]$ of R and Q_k, respectively. Let $P = (p_1, \ldots, p_\ell)$, $\ell \leq n$ be the set of different pitch values among the notes of R.

▶ **Lemma 3.7.** *Let R be a melodic contour with time partition $X = (x_0 = 0, \ldots, x_n)$. Let $0 < k \leq n$ be a natural number. There exists a melody contour Q_k^* of k notes and time partition $T = \{t_0 = 0, \ldots, t_k = x_n\}$ that minimizes the area difference with R overall all the melodic contours of k notes starting at time x_0 and ending at time x_n and fulfills the following two properties:*

1. *$T \subseteq X$ and,*
2. *each note of Q_k^* contains at least a note of R.*

Recall that the notes of a k-compression of R contain notes of R. The previous lemma says that there exists an optimal melodic contour with k notes that is a k-compression.

Let R be a melodic contour with time partition $X = \{x_0, \ldots, x_n\}$. For every $x_0 \leq t \leq x_n$, we denote by $R_{\overleftarrow{t}}$ the *prefix melody* of R with time partition $X_{\overleftarrow{t}} = \{x_0, \ldots, x_i, t\}$, where $x_i < t \leq x_{i+1}$. The following result establishes an optimal substructure of a solution.

▶ **Lemma 3.8.** *Given three values $1 \leq j \leq k$, $j \leq i \leq n$ and $p \in P$, denote by $S_p(i, j)$ the set of all the melodies formed by j notes starting at time x_0 and ending at time x_i whose last note μ has pitch $p \in P$ and starts at some time $x \in X$, $x < x_i$. Then, there exists a melody $C \in S_p(i, j)$ that minimizes the area difference with $R_{\overleftarrow{x_i}}$ over all the melodies in $S_p(i, j)$ such that $C_{\overleftarrow{x_{i'}}}$ is an optimum $(j-1)$-compression melody of $R_{\overleftarrow{x_{i'}}}$, where $x_{i'}$ is the starting time of the last note of C.*

Based on the previous result, dynamic programming can be used and we can prove:

▶ **Theorem 3.9.** *Problem 1.4 can be solved in $O(k\ell n)$ time, where ℓ is the number of different pitch values of R.*

References

1 G. Aloupis, T. Fevens, S. Langerman, T. Matsui, A. Mesa, Y. Nuñez, D. Rappaport, and G. Toussaint. Algorithms for computing geometric measures of melodic similarity. *Computer Music Journal*, 30(3):67–76, 2006.

2 R. Cilibrasi, P. Vitányi, and R. de Wolf. Algorithmic clustering of music based on string compression. *Computer Music Journal*, 28(4):49–67, 2004.

3 R. Clifford, M. Christodoulakis, T. Crawford, D. Meredith, and G. A. Wiggins. A fast, randomised, maximal subset matching algorithm for document-level music retrieval. In *ISMIR*, pages 150–155, 2006.

4 J.-S. Roger Jang, H.-R. Lee, and Ming-Yang K. Content-based music retrieval using linear scaling and branch-and-bound tree search. In *IEEE International Conference on Multimedia and Expo, 2001. ICME 2001.*, pages 74–74, 2001.

5 D. O. Maidín. A geometrical algorithm for melodic difference. *Computing in musicology: a directory of research*, (11):65–72, 1998.

6 G. Toussaint. Computational geometric aspects of rhythm, melody, and voice-leading. *Computational Geometry*, 43(1):2–22, 2010.

Rotational symmetric flexible placements of graphs*

Sean Dewar[1], Georg Grasegger[1], and Jan Legerský[2,3]

1 **Johann Radon Institute for Computational and Applied Mathematics (RICAM), Austrian Academy of Sciences**
sean.dewar@ricam.oeaw.ac.at, georg.grasegger@ricam.oeaw.ac.at

2 **Johannes Kepler University Linz, Research Institute for Symbolic Computation (RISC)**
jan.legersky@risc.jku.at

3 **Department of Applied Mathematics, Faculty of Information Technology, Czech Technical University in Prague**

Abstract

We study the existence of an n-fold rotational symmetric placement of a symmetric graph in the plane allowing a continuous deformation that preserves the symmetry and the distances between adjacent vertices. We show that such a flexible placement exists if and only if the graph has a NAC-colouring satisfying an additional property on the symmetry; a NAC-colouring is a surjective edge colouring by two colours such that every cycle is either monochromatic, or there are at least two edges of each colour.

Rigid graphs are those which have only finitely many non-congruent placements in the plane with the same edge lengths as a generic placement. These graphs can, however, have non-generic special choices of a placement such that there are infinitely many non-congruent placements in the plane with the same edge lengths. We call such a placement *flexible*.

The study of flexible placements of generically rigid graphs has a long history. Dixon found two types of flexible placements of the bipartite graph $K_{3,3}$ [3, 17, 14]. Walter and Husty [15] proved that these are indeed all (assuming that vertices do not overlap). Figure 1 shows some special symmetric cases of these two constructions applied to $K_{4,4}$. Further examples of graphs with flexible placements are Burmester's focal point mechanism [1], a 12-vertex graph studied by Kempe [11], and two constructions by Wunderlich [16, 18].

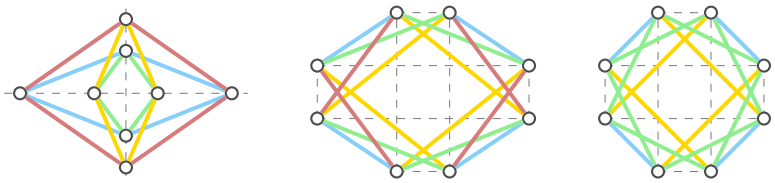


Figure 1 The vertices of $K_{4,4}$ can be placed symmetrically on orthogonal lines to make the graph flexible with 2-fold rotational symmetry (left). A 2-fold rotationally symmetric flexible instance of $K_{4,4}$ is obtained by placing the vertices of each part to a rectangle so that the two rectangles have the same intersection of diagonals and parallel/orthogonal edges (middle). Although there is a 4-fold rotationally symmetric choice of rectangles (right), the deformed placements preserving the edge lengths are only 2-fold symmetric. The colours indicate equality of edge lengths in a placement.

* This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 675789. The project was partially supported by the Austrian Science Fund (FWF): P31061, P31888 and W1214-N15.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

In recent works [7, 8, 4] a deeper analysis of existence of flexible placements is done via graph colourings. There is a special type of edge colourings, called NAC-colourings ("No Almost Cycles", see [7]), which classify the existence of a flexible placement in the plane and give a construction of the motion. Furthermore, determining the NAC-colourings of a given graph and the possible constructions that come from it can be done easily by using the SageMath package FlexRiLoG [5]. In [6] we used these methods for constructing flexible placements for symmetric graphs as in Figure 2. However, we did not take advantage of the symmetry for the construction, and instead had to construct the framework manually.

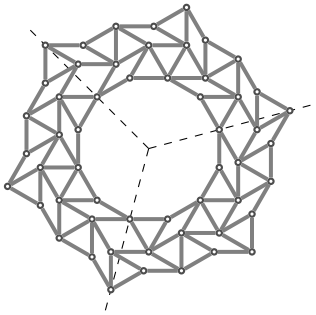


Figure 2 A symmetric graph which has a 3-fold rotationally symmetric flexible placement.

Symmetry plays an important role in art and design, and often appears in nature also. Due to this, there is a large body of work focused on symmetric frameworks and their properties in the context of rigidity theory; see [10, 12]. In particular, we shall be focusing on graphs and frameworks that display n-fold rotational symmetry, such as in Figure 2.

In this extended abstract, we formalise the NAC-colouring method for rotationally symmetric flexible placements, i. e. in such a way that the motion preserves the symmetry. By combining Theorem 3.1 and Theorem 3.2, we obtain the following result:

Let G be a $\mathcal{C}_n$-symmetric connected graph. Then G has a $\mathcal{C}_n$-symmetric NAC-colouring if and only if there exists a $\mathcal{C}_n$-symmetric flexible framework (G,p) in $\mathbb{R}^2$.

1 Preliminaries

We briefly recall some basic notions from rigidity theory and define NAC-colourings in this section. All graphs $G = (V(G), E(G))$ in the paper are connected and $|E(G)| \geq 1$.

▶ **Definition 1.1.** A *framework in* $\mathbb{R}^2$ is a pair (G,p) where G is a (finite simple) graph and $p : V(G) \to \mathbb{R}^2$ is a *placement* of G, a possibly non-injective map such $p(u) \neq p(v)$ if $uv \in E(G)$. We define frameworks (G,p) and (G,q) to be *equivalent* if for all $uv \in E(G)$,

$$\|p(u) - p(v)\| = \|q(u) - q(v)\| . \tag{1}$$

We define two placements p, q of G to be *congruent* if (1) holds for all $u, v \in V(G)$.

An equivalent definition for congruence is as follows; p and q are congruent if there exists an Euclidean isometry M of $\mathbb{R}^2$ such that $Mq(v) = p(v)$ for all $v \in V(G)$.

▶ **Definition 1.2.** Let (G,p) be a framework. A *flex (in $\mathbb{R}^2$)* of (G,p) is a continuous path $t \mapsto p_t$, $t \in [0,1]$, in the space of placements of G such that $p_0 = p$ and each (G,p_t) is equivalent to (G,p). If p_t is congruent to p for all $t \in [0,1]$ then p_t is *trivial*. We define (G,p) to be *flexible* if there is a non-trivial flex of (G,p) in $\mathbb{R}^2$, and *rigid* otherwise.

It was shown in [13] that a framework (G,p) with a generic placement of vertices (see [9]) is rigid if and only if G contains a Laman graph as a spanning subgraph. This does not inform us whether a graph will have a flexible placement; for example, any generic placement of $K_{4,4}$ is rigid, however as shown by Figure 1 we can construct flexible placements for it. To determine whether a graph has flexible placements we introduce the following.

▶ **Definition 1.3.** An edge colouring $\delta : E(G) \to \{\text{red}, \text{blue}\}$ of a graph G is a *NAC-colouring* if $\delta(E(G)) = \{\text{red}, \text{blue}\}$ and for each cycle in G, either all edges have the same colour, or there are at least two red and two blue edges. NAC-colourings $\delta, \overline{\delta}$ of G are *conjugated* if $\delta(e) \neq \overline{\delta}(e)$ for all $e \in E(G)$.

▶ Remark. The colourings considered within this paper are not required to have incident edges coloured differently, contrary to the common graph-theoretical terminology.

Having these definitions, we can recall the result [7, Theorem 3.1].

▶ **Theorem 1.4.** *A graph has a flexible placement in* $\mathbb{R}^2$ *if and only if it has a NAC-colouring.*

2 Rotational symmetry

The following definitions specify the rotational symmetric setting where Definition 2.1 gives a combinatorial description of symmetry of a graph and Definition 2.2 describes geometric symmetry of a framework. Note, that in figures we describe vertices in graphs by filled disks and in frameworks with circles.

▶ **Definition 2.1.** Let G be a graph and $n \geq 2$. Let the *n-fold rotation group*, $\mathcal{C}_n := \langle \omega : \omega^n = 1 \rangle$ act on G, i.e., there exists an injective group homomorphism $\theta : \mathcal{C}_n \to \mathrm{Aut}(G)$, where $\mathrm{Aut}(G)$ is the automorphism group of G. We define $\gamma v := \theta(\gamma)(v)$ for $\gamma \in \mathcal{C}_n$; similarly, for any edge $e = uv \in E(G)$, we define $\gamma e := \gamma u \gamma v$. We shall define $v \in V(G)$ to be an *invariant vertex* if $\gamma v = v$ for all $\gamma \in \mathcal{C}_n$, and *partially invariant* if $\gamma v = v$ for some $\gamma \in \mathcal{C}_n, \gamma \neq 1$. The graph G is called *$\mathcal{C}_n$-symmetric* if:

(a) a vertex is invariant if and only if it is partially invariant, and
(b) the set of invariant vertices of G forms an independent set.

▶ **Definition 2.2.** Let (G,p) be a framework in $\mathbb{R}^2$, G be $\mathcal{C}_n$-symmetric and $\tau : \mathcal{C}_n \to O(2,\mathbb{R})$ be a *symmetry map*, i.e., an injective group homomorphism, given by

$$\tau(\omega) = \begin{bmatrix} \cos(2\pi/n) & \sin(2\pi/n) \\ -\sin(2\pi/n) & \cos(2\pi/n) \end{bmatrix}.$$

If $p(\gamma v) = \tau(\gamma)p(v)$ for each $v \in V(G)$ and $\gamma \in \mathcal{C}_n$, then (G,p) is a *$\mathcal{C}_n$-symmetric framework*; likewise, we define p to be a *$\mathcal{C}_n$-symmetric placement* of G.

We note that if (G,p) is $\mathcal{C}_n$-symmetric then it is $\mathcal{C}_m$-symmetric for all $m|n$.

▶ **Definition 2.3.** Let (G,p) be a $\mathcal{C}_n$-symmetric framework in $\mathbb{R}^2$. If there is a non-trivial flex p_t of (G,p) such that each (G,p_t) is $\mathcal{C}_n$-symmetric, then (G,p) is *$\mathcal{C}_n$-symmetric flexible* (or *n-fold rotation symmetric flexible*), and *$\mathcal{C}_n$-symmetric rigid* otherwise.

We define the following for edge colourings of $\mathcal{C}_n$-symmetric graphs.

▶ **Definition 2.4.** Let G be a $\mathcal{C}_n$-symmetric graph with colouring δ. A *red, resp. blue, component* is a connected component of $G^\delta_{\text{red}} := (V(G), \{e \in E(G) : \delta(e) = \text{red}\})$, resp. G^δ_{blue}. A blue or red component $H \subset G$ is *partially invariant* if there exists $\gamma \in \mathcal{C}_n \setminus \{1\}$ such that $\gamma H = H$, and *invariant* if $\gamma H = H$ for all $\gamma \in \mathcal{C}_n$ (see Figure 3 for an example).

Figure 3 A partially invariant (but not invariant) red component on the left and an invariant red component (and therefore also partially invariant) on the right for $\mathcal{C}_6$-symmetry. The symmetry is indicated by the graph layout.

We focus on the following class of NAC-colourings suitable for dealing with symmetries.

▶ **Definition 2.5.** Let G be a $\mathcal{C}_n$-symmetric graph with NAC-colouring δ. We define δ to be a *$\mathcal{C}_n$-symmetric NAC-colouring* if $\delta(\gamma e) = \delta(e)$ for all $e \in E(G)$ and $\gamma \in \mathcal{C}_n$ and no two distinct blue, resp. red, partially invariant components are connected by an edge (see Figure 4 for examples).

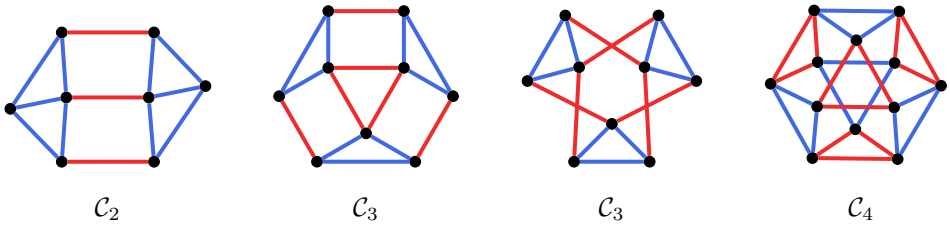


Figure 4 Valid $\mathcal{C}_n$-symmetric NAC-colouring for some graphs. For the first three graphs the symmetry is indicated by the layout. There is a $\mathcal{C}_4$-symmetry for the graph on the right which is not geometrically visible in the figure; see Figure 8 for a $\mathcal{C}_4$-symmetric placement.

▶ **Example 2.6.** The cartesian product of K_4 and K_2 has a single NAC-colouring (up to conjugation) δ, see Figure 5. The graph is $\mathcal{C}_2$-symmetric under the symmetry θ, where $\theta(\omega)$ is the permutation $(1,6)(2,5)(3,8)(4,7)$; further, the NAC-colouring δ is a $\mathcal{C}_2$-symmetric NAC colouring with respect to θ. The graph is also $\mathcal{C}_4$-symmetric under the symmetry θ', where $\theta'(\omega)$ is the permutation $(1,6,3,8)(5,2,7,4)$; however, the NAC-colouring δ is not a $\mathcal{C}_4$-symmetric NAC colouring with respect to θ', since the blue partially invariant components are connected by edges.

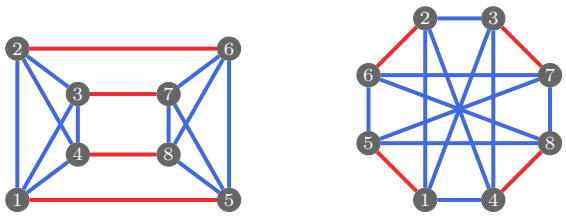


Figure 5 The cartesian product of K_4 and K_2 has only one NAC-colouring. It is $\mathcal{C}_2$-symmetric (left) but not $\mathcal{C}_4$-symmetric (right). The symmetries are indicated by the graph layout.

3 Necessary and sufficient conditions for rotationally symmetric flexibility

In this section we show the construction of $\mathcal{C}_n$-symmetric motions from $\mathcal{C}_n$-symmetric NAC-colourings. The inverse direction is also true.

▶ **Theorem 3.1.** *If (G,p) is a $\mathcal{C}_n$-symmetric flexible framework in $\mathbb{R}^2$, G being a connected graph, then G has a $\mathcal{C}_n$-symmetric NAC-colouring.*

The proof of Theorem 3.1 follows in a similar vein to the proof of [7, Theorem 3.1] by using methods from valuation theory. There are more complexities involved however, since we also require that the colouring obtained respects the symmetry of the graph and partially invariant components are not connected by edges.

▶ **Theorem 3.2.** *Let G be a $\mathcal{C}_n$-symmetric connected graph. If G has a $\mathcal{C}_n$-symmetric NAC-colouring δ, then there exists a $\mathcal{C}_n$-symmetric flexible framework (G,p) in $\mathbb{R}^2$.*

Proof. The proof is based on the "zigzag" grid construction from [7] with a specific choice of the grid. Let $R_1^0,\dots,R_1^{n-1},\dots,R_m^0,\dots,R_m^{n-1}$ be the red components of G^δ_{red} that are not partially invariant. We can assume that $R_j^i=\omega^i R_j^0$ for $0\le i<n$ and $1\le j\le m$. Similarly, let $B_1^0,\dots,B_1^{n-1},\dots,B_k^0,\dots,B_k^{n-1}$ be the blue components of G^δ_{blue} that are not partially invariant and $B_j^i=\omega^i B_j^0$ for $0\le i<n$ and $1\le j\le k$.

Let $a_1,\dots,a_m$ and $b_1,\dots,b_k$ be points in $\mathbb{R}^2\setminus\{(0,0)\}$ such that $a_j\neq\tau(\omega)^i a_{j'}$ and $b_j\neq\tau(\omega)^i b_{j'}$ for $j\neq j'$ and $1\le i<n$ arbitrary. We define functions $\overline{a},\overline{b}\colon V(G)\to\mathbb{R}^2$ by

$$\overline{a}(v)=\begin{cases}\tau(\omega)^i a_j & \text{if } v\in R_j^i\\ (0,0) & \text{otherwise,}\end{cases}\quad\text{and}\quad \overline{b}(v)=\begin{cases}\tau(\omega)^i b_j & \text{if } v\in B_j^i\\ (0,0) & \text{otherwise.}\end{cases}$$

We note that a vertex is mapped to the origin by $\overline{a}$ (respectively, $\overline{b}$) if and only if it lies in a red (respectively, blue) partially invariant component. We now obtain for each $t\in[0,2\pi]$ a placement p_t of G, where

$$p_t(v):=\begin{pmatrix}\cos t & -\sin t\\ \sin t & \cos t\end{pmatrix}\overline{a}(v)+\overline{b}(v)\,. \tag{2}$$

Let $uv\in E(G)$. If $\delta(uv)$ is red (resp. blue) then $\overline{a}(u)=\overline{a}(v)$ (resp. $\overline{b}(u)=\overline{b}(v)$). Hence, the edge length $\|p_t(u)-p_t(v)\|$ is independent of t.

Next, we have to show that no two adjacent vertices are mapped to the same point by the placement p_0. Assume that $\big(\overline{a}(u),\overline{b}(u)\big)=\big(\overline{a}(v),\overline{b}(v)\big)$ for some vertices u,v. Suppose this is due to the fact that u and v belong to the same red and same blue (possibly partially invariant) component. Hence, $uv\notin E(G)$, otherwise δ is not a NAC-colouring (uv would yield a cycle with a single edge in one color). On the other hand, if u and v are in two different red (resp. blue) components, then $\overline{a}(u)=\overline{a}(v)=(0,0)$ (resp. $\overline{b}(u)=\overline{b}(v)=(0,0)$). By our construction of $\overline{a}$ (resp. $\overline{b}$), it follows that u,v both lie in partially invariant red (resp. blue) components. Since these components are partially invariant, $uv\notin E(G)$ by the assumption that δ is $\mathcal{C}_n$-symmetric.

As edge lengths are preserved and no two vertices connected by an edge are mapped to the same point, $(G,p):=(G,p_0)$ is a framework with a flex p_t. Further, the flex is not trivial by surjectivity of δ, thus (G,p) is a flexible framework.

Finally, we show that p_t is $\mathcal{C}_n$-symmetric. If $v\in R_j^i\cap B_\ell^k$, then

$$\omega v\in\tau(\omega)R_j^i\cap\tau(\omega)B_\ell^k=R_j^{(i+1 \bmod n)}\cap B_\ell^{(k+1 \bmod n)}.$$

Hence, $\overline{a}(\omega v) = \tau(\omega)\overline{a}(v)$ and $\overline{b}(\omega v) = \tau(\omega)\overline{b}(v)$. The same equalities hold also if v belongs to a partially invariant component, since then ωv is also in a partially invariant component. Using commutativity of rotation matrices, we conclude the proof by (2):

$$p_t(\omega v) = \tau(\omega)\begin{pmatrix}\cos t & -\sin t\\ \sin t & \cos t\end{pmatrix}\overline{a}(v) + \tau(\omega)\overline{b}(v) = \tau(\omega)p_t(v). \qquad \blacktriangleleft$$

▶ **Example 3.3.** By using the construction described in Theorem 3.2 we can construct the $\mathcal{C}_n$-symmetric flexible frameworks given in Figures 6, 7 and 8.

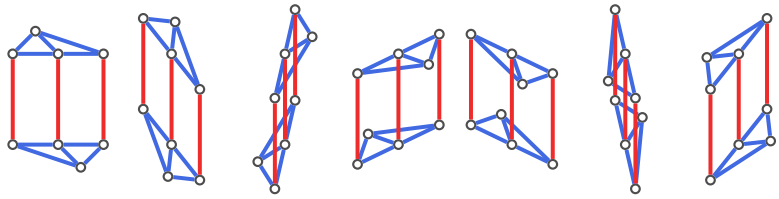


Figure 6 A flexible $\mathcal{C}_2$-symmetric placement for a given $\mathcal{C}_2$-symmetric NAC-colouring.

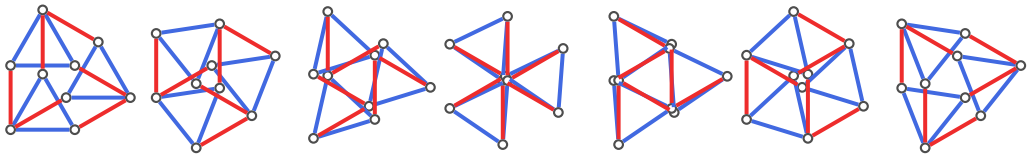


Figure 7 A flexible $\mathcal{C}_3$-symmetric placement for a given $\mathcal{C}_3$-symmetric NAC-colouring.

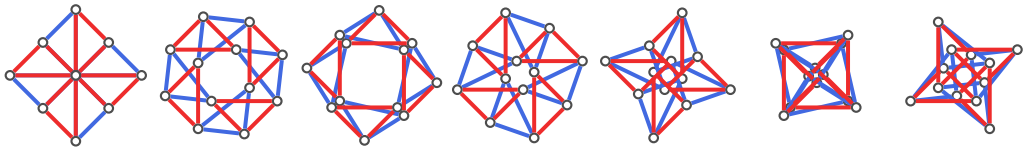


Figure 8 A flexible $\mathcal{C}_4$-symmetric placement for a given $\mathcal{C}_4$-symmetric NAC-colouring.

▶ **Example 3.4.** We consider the graph in Figure 9 with the given $\mathcal{C}_3$-symmetric NAC-colouring. Then the red 3-cycle forms a red partially invariant component. Therefore its position is fixed during the motion.

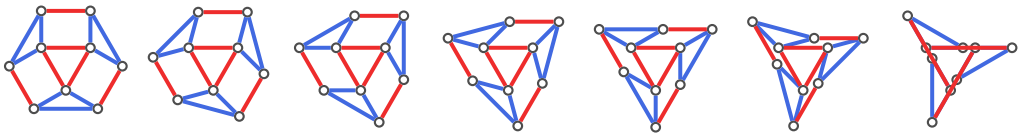


Figure 9 A flexible $\mathcal{C}_3$-symmetric placement for a given $\mathcal{C}_3$-symmetric NAC-colouring.

The constructed framework given by the proof of Theorem 3.2 may not be *proper flexible*, i.e., have no overlapping vertices, see Figure 10. As outlined in [8], there are necessary and sufficient conditions for determining when a graph will or will not have a proper flexible placements, although they have as of yet not been adapted fully to the symmetric case.

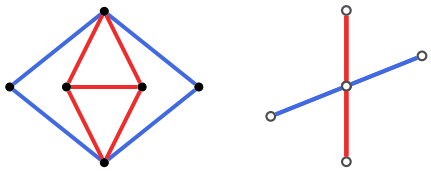


Figure 10 A flexible $\mathcal{C}_2$-symmetric placement for a given $\mathcal{C}_2$-symmetric NAC-colouring for which two vertices overlap.

▶ Remark. While we have only dealt with frameworks with rotational symmetry, there are other types of symmetry for the plane, namely reflectional and translational symmetry. Although flexible placements that preserve translational symmetry have very recently been investigated [2], not much is known for flexible placements that preserve reflectional symmetry or preserve both reflectional and rotational symmetry.

References

1. L. Burmester. Die Brennpunktmechanismen. *Zeitschrift für Mathematik und Physik*, 38:193–223, 1893.
2. S. Dewar. Flexible placements of periodic graphs in the plane, 2019. arXiv:1911.05634.
3. A. C. Dixon. On certain deformable frameworks. *Messenger*, 29(2):1–21, 1899.
4. M. Gallet, G. Grasegger, J. Legerský, and J. Schicho. On the existence of paradoxical motions of generically rigid graphs on the sphere, 2019. arXiv:arXiv:1908.00467.
5. G. Grasegger and J. Legerský. FlexRiLoG — SageMath package for Flexible and Rigid Labelings of Graphs. Zenodo, May 2019. doi:10.5281/zenodo.3078758.
6. G. Grasegger, J. Legerský, and J. Schicho. Animated Motions of Exceptional Flexible Instances of Generically Rigid Graphs. In *Bridges Linz 2019 Conference Proceedings*, pages 255–262, Phoenix, Arizona, 2019. Tessellations Publishing. doi:10.5281/zenodo.3518805.
7. G. Grasegger, J. Legerský, and J. Schicho. Graphs with Flexible Labelings. *Discrete & Computational Geometry*, 62(2):461–480, 2019. doi:10.1007/s00454-018-0026-9.
8. G. Grasegger, J. Legerský, and J. Schicho. Graphs with Flexible Labelings allowing Injective Realizations. *Discrete Mathematics*, in press, 2019. doi:10.1016/j.disc.2019.111713.
9. Jack Graver, Brigitte Servatius, and Herman Servatius. *Combinatorial rigidity*. Graduate Studies in Mathematics. American Mathematical Society, 1993. doi:10.1090/gsm/002.
10. T. Jordán, V. E. Kaszanitzky, and S. Tanigawa. Gain-sparsity and symmetry-forced rigidity in the plane. *Discrete Comput. Geom.*, 55(2):314–372, March 2016.
11. A. B. Kempe. On Conjugate Four-piece Linkages. *Proceedings of the London Mathematical Society*, s1-9(1):133–149, 11 1877. doi:10.1112/plms/s1-9.1.133.
12. J. Owen and S. Power. Frameworks symmetry and rigidity. *International Journal of Computational Geometry & Applications*, 20, 04 2012. doi:10.1142/S0218195910003505.
13. H. Pollaczek-Geiringer. Über die Gliederung ebener Fachwerke. *Zeitschrift für Angewandte Mathematik und Mechanik (ZAMM)*, 7:58–72, 1927. doi:10.1002/zamm.19270070107.
14. H. Stachel. On the flexibility and symmetry of overconstrained mechanisms. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 372, 2013. doi:10.1098/rsta.2012.0040.
15. D. Walter and M. L. Husty. On a nine-bar linkage, its possible configurations and conditions for paradoxical mobility. In *12th World Congress on Mechanism and Machine Science, IFToMM 2007*, 2007.
16. W. Wunderlich. Ein merkwürdiges Zwölfstabgetriebe. *Österreichisches Ingenieur-Archiv*, 8:224–228, 1954.

17 W. Wunderlich. On deformable nine-bar linkages with six triple joints. *Indagationes Mathematicae (Proceedings)*, 79(3):257–262, 1976. doi:10.1016/1385-7258(76)90052-4.

18 W. Wunderlich. Mechanisms related to Poncelet's closure theorem. *Mechanisms and Machine Theory*, 16:611–620, 1981. doi:10.1016/0094-114X(81)90067-7.

Augmenting Polygons with Matchings*

Alexander Pilz[1], Jonathan Rollin[2], Lena Schlipf[3], and André Schulz[2]

1 **Graz University of Technology**
apilz@ist.tugraz.at
2 **FernUniversität in Hagen**
{jonathan.rollin | andre.schulz}@fernuni-hagen.de
3 **Universität Tübingen**
schlipf@informatik.uni-tuebingen.de

Abstract

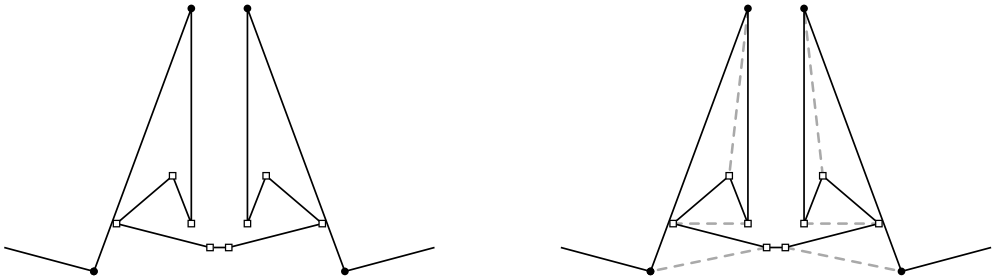
We study disjoint compatible noncrossing geometric matchings of simple polygons. That is, given a simple polygon P we want to draw a set of pairwise disjoint straight line edges with endpoints on the vertices of P such that these new edges neither cross nor contain any edge of the polygon. We prove NP-completeness of deciding whether there is such a perfect matching. For any n-vertex polygon we show that such a matching with $\leq (n-4)/8$ edges is not maximal, that is, it can be extended by another compatible matching edge. Complementing this we construct polygons with maximal matchings with $n/6$ edges. Finally we consider a related problem. We prove that it is NP-complete to decide whether a noncrossing geometric graph G admits a set of compatible noncrossing edges such that G together with these edges has minimum degree five.

1 Introduction

A geometric graph is a graph drawn in the plane with straight-line edges. Throughout this paper we additionally assume that all geometric graphs are noncrossing. Let G be a given (noncrossing) geometric graph G. We want to augment G with a geometric matching on the vertices of G such that no edges cross in the augmentation. We call such a (geometric) matching *compatible* with G. Note that our definition of a compatible matching implies that the matching is noncrossing and avoids the edges of G. Questions regarding compatible matchings were first studied by Rappaport et al. [13, 14]. Rappaport [13] proved that it is NP-hard to decide whether for a given geometric graph G there is a compatible matching M such that $G + M$ is a (spanning) cycle. Recently Akitaya et al. [3] confirmed a conjecture of Rappaport and proved that this holds even if G is a perfect matching. Note that in this case also M is necessarily a perfect matching. However, for some compatible perfect matchings M the union $G + M$ might be a collection of several disjoint cycles. There are graphs G that do not admit any compatible perfect matching, even when G is a matching. Such matchings were studied by Aichholzer et al. [1] who proved that each m-edge perfect matching G admits a compatible matching of size at least $\frac{4}{5}m$. Ishaque et al. [9] confirmed a conjecture of Aichholzer et al. [1] that any perfect matching G with an even number of edges admits a compatible perfect matching. For a geometric graph G let $d(G)$ denote the size of a largest compatible matching of G and for a family $\mathcal{F}$ of geometric graphs let $d(\mathcal{F}) = \min\{d(G) \mid G \in \mathcal{F}\}$. Aichholzer et al. [2] proved that for the families T_n and P_n of all n-vertex geometric trees, respectively n-vertex polygons, $\frac{1}{10}n \leq d(T_n) \leq \frac{1}{4}n$ and $\frac{n-3}{4} \leq d(P_n) \leq \frac{1}{3}n$ holds.

* This work was initiated during the 15th European Research Week on Geometric Graphs in Pritzhagen, Germany.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.



(a) (b)

Figure 1 (a) This gadget allows for simulating a "bend" in the polygon without a vertex that needs to be matched. The construction is scaled such that the eight points marked with squares do not see any other point outside of the gadget (in particular, narrowing it horizontally). (b) A possible matching is shown.

We continue this line of research and consider the following problems. Given a geometric cycle, i.e., a polygon, we first show that it is NP-complete to decide whether the polygon admits a compatible perfect matching. Then we ask for the "worst" compatible matchings for a given polygon. That is, we search for small maximal compatible matchings.

The first studied problem can also be phrased as follows: Given a geometric cycle, can we add edges to obtain a cubic geometric graph? In the last section, we consider a related augmentation problem. Given a geometric graph, we show that it is NP-complete to decide whether the graph can be augmented to a graph of minimum degree five. The corresponding problem for the maximal vertex degree asks to add a *maximal* set of edges to the graph such that the maximal vertex degree is bounded by constant. This problem is also known to be NP-complete for maximum degree at most seven [10].

A survey of Hurtado and Tóth [8] discusses several other augmentation problems for geometric graphs. Besides the problems mentioned in that survey decreasing the diameter [5] and the continuous setting (where every point along the edges of an embedded graph is considered as a vertex) received considerable attention [4, 7].

2 Compatible perfect matchings in polygons

▶ **Theorem 2.1.** *Given a simple polygon, it is* NP*-complete to decide whether it admits a compatible perfect matching.*

Proof. The problem is obviously in NP, as a certificate one can merely provide the added edges. NP-hardness is shown by a reduction from POSITIVE PLANAR 1-IN-3-SAT. In this problem, shown to be NP-hard by Mulzer and Rote [11], we are given an instance of 3-SAT with a planar variable-clause incidence graph and no negative literals; the instance is considered satisfiable if and only if there is exactly one true variable per clause.

For a given 1-in-3-SAT formula, we take an embedding of its incidence graph and replace its elements by gadgets. We first show that finding compatible matchings for a set of disjoint simple polygons is hard and then show how to connect the individual polygons to obtain a single polygon.

Our construction relies on a gadget that restricts the possible matching edges of vertices. In particular, we introduce a polygonal chain, whose vertices need to be matched to each other. This is achieved by the *twin-peaks gadget* as shown in Fig. 1. The gadget is scaled such that the eight vertices in its interior (which are marked with squares in Fig. 1) do not

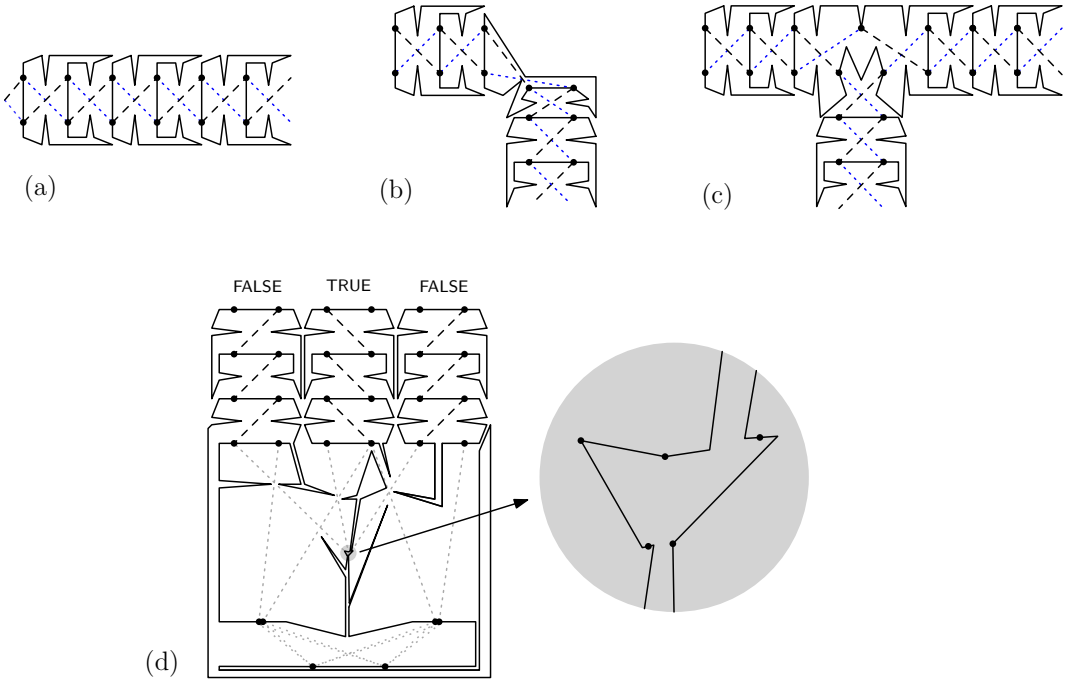


Figure 2 (a) A wire gadget and its two truth states (one in dashed, the other in dotted). (b) A bend in a wire gadget. (c) A split gadget that transports the truth setting of one wire to two other ones. This is used for representing the variables. (d) A clause gadget. The visibility among the vertices of degree two is indicated by the lighter lines. Exactly one vertex of degree two of the part in the circle must be connected to a wire above that carries the true state.

see any edges outside of the gadget. The two topmost vertices must have an edge to the vertices directly below as the vertices below do not see any other (non-adjacent) vertex. The remaining six "square" vertices do not have a geometric perfect matching on their own, so any perfect geometric matching containing them must connect them to the two bottommost vertices. Clearly, there is such a matching.

We now present the remaining gadgets (*wire*, *split*, and *clause*) for our reduction. The ideas are inspired by the reduction of Pilz [12]. In the following illustrations, vertices of degree two are drawn as a dot. Vertices in the figures without a dot represent a sufficiently small twin-peaks gadget.

The *wires* propagate the truth assignment of a variable. A wire consists of a sequence of polygons, each containing four vertices of degree two. There are only two possible global matchings for the vertices of degree two; see Fig. 2(a). A *bend* in a wire can be drawn as shown in Fig. 2(b). The truth assignment of a wire can be duplicated by a *split gadget*; see Fig. 2(c). A variable is represented by a cyclic wire with split gadgets. The *clause gadget* is illustrated in Fig. 2(d), where the wires enter from the top. The vertices there can be matched if and only if one of the vertices is connected to a wire that is in the true state. The vertices at the bottom of the gadget make sure that if there are exactly two wires in the false state, then we can add an edge to them. Hence, this set of polygons has a compatible perfect matching if and only if the initial formula was satisfiable.

It remains to "merge" the polygons of the construction to one simple polygon. The elements of the clause gadget are connected to the last polygons of the wires entering it. Observe that two neighboring polygons of a wiring gadget can be connected by adding four

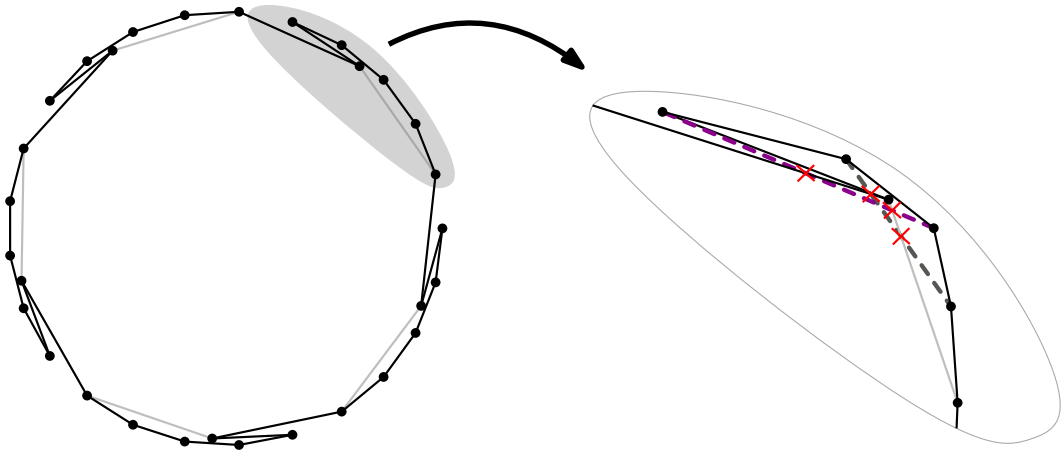


Figure 3 A polygon (black) with a maximal matching (gray) with only $\frac{n}{3}$ matched vertices. Notice that there is exactly one compatible edge between the six vertices in the gray area.

bends (using four twin-peaks gadgets). We can consider the incidence graph to be connected (otherwise the reduction splits into disjoint problems). Hence, we can always connect two disjoint polygons to one, until there is only a single polygon left. ◀

3 Compatible maximal matching in polygons

For a geometric graph G let $\mathrm{mm}(G)$ denote the size of a minimal maximal compatible matching of G and for a family $\mathcal{F}$ of geometric graphs let $\mathrm{mm}(\mathcal{F}) = \min\{\mathrm{mm}(G) \mid G \in \mathcal{F}\}$.

▶ **Theorem 3.1.** *Let P_n denote the family of all n-vertex polygons. Then $\mathrm{mm}(P_n) \geq \frac{n-4}{8}$ for all n and $\mathrm{mm}(P_n) \leq \frac{1}{6}n$ for infinitely many values of n.*

Proof. The construction in Fig. 3 shows that for infinitely many values of n there is an n-vertex polygon with a compatible maximal matching of size $\frac{n}{6}$. This shows $\mathrm{mm}(P_n) \leq \frac{n}{6}$ for infinitely many values of n.

It remains to prove the lower bound. Let P be an n-vertex polygon with a maximal compatible matching M. As the claim clearly holds for any triangle P we assume that $n \geq 4$. We shall subdivide the plane into cells with further edges as follows. First draw a rectangle enclosing P in the outer face. Then, for each reflex angle in $P + M$ (one after the other) draw a straightline edge starting at the incident vertex such that the edge cuts the reflex angle into two convex angles and stops when it hits some already drawn edge (but not a vertex). See Figure 4. The final drawing D contains at most $2 + |E(M)| + n$ bounded cells. Indeed, each edge on top of P subdivides some cell into two, where $|E(M)|$ such edges are in M and each vertex of P gives rise to at most one further such edge through a reflex angle. Moreover, all bounded cells in D are convex regions. Hence, any two unmatched vertices of P incident to a common cell F in D are connected by a side of P (within the boundary of F) as otherwise M is not maximal. This shows that each cell is incident to at most two unmatched vertices of P, since P is not a triangle. Each unmatched vertex of P is incident to exactly three bounded cells of D. Therefore, $3(n - |V(M)|) \leq 2\,(2 + |E(M)| + n)$ and hence $|E(M)| \geq (n-4)/8$. ◀

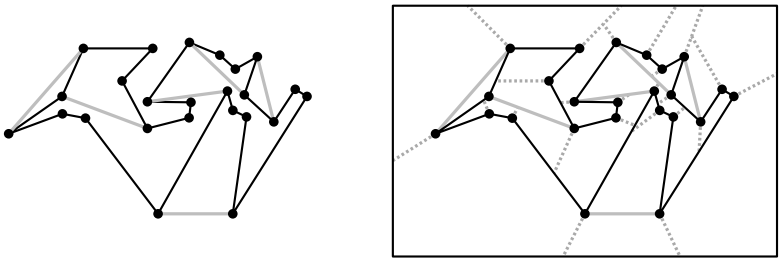


Figure 4 A polygon (black) with a maximal matching (gray) where each reflex angle is cut by a dotted edge.

4 Augmenting to Minimum Degree Five

In this section, we show that augmenting to a graph with minimum degree five is NP-complete. A related result states that it is NP-hard to decide whether a geometric graph can be augmented to a cubic graph [12].

▶ **Theorem 4.1.** *Given a geometric crossing-free graph G, it is* NP*-complete to decide whether there is a set of compatible edges E such that $G + E$ has minimum degree five.*

Proof. The problem is obviously in NP, a certificate provides the added edges. NP-hardness is shown by a reduction from MONTONE PLANAR RECTILINEAR 3-SAT. In this problem, shown to be NP-hard by de Berg and Khosravi [6], we are given an instance of monotone (meaning that each clause has only negative or only positive variables) 3-SAT with a planar incidence graph. In this graph, the variable and clause gadgets are represented by rectangles. All variable rectangles lie on a horizontal line. The clauses with positive variables lie above the variables and the ones with negative variables below. The edges connecting the clause gadgets to the variable gadgets are vertical line segments and no edges cross. See Fig. 5 (a).

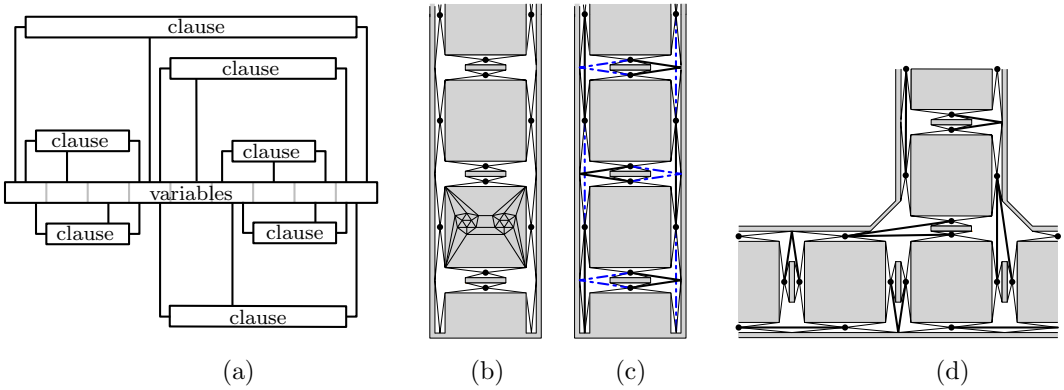
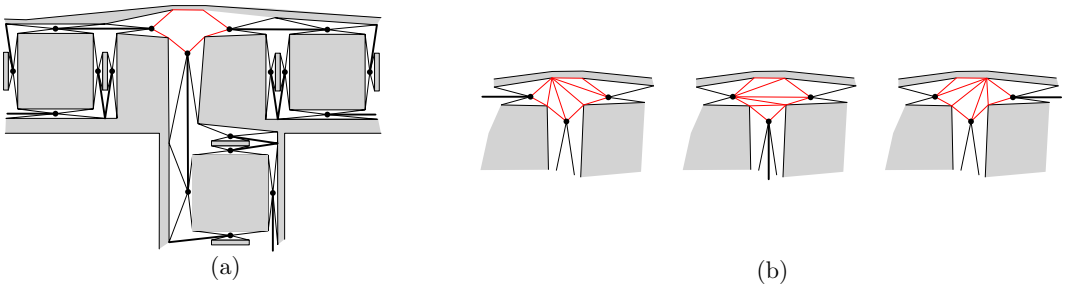


Figure 5 (a) A montone rectilinear representation of a planar 3SAT instance. (b) A wire gadget (vertices of degree four are drawn with dots) and its (c) two truth states (one in bold, the other dotted). Vertices incident to a gray region have degree at least five. This can be achieved by adding edges and vertices (of degree at least five) inside the gray regions – as shown in one gray square. (d) A split gadget with edges for the truth assignment (bold).

For a given 3-SAT formula, we take an embedding of its incident graph (as discussed) and replace its elements by gadgets. Again, we have a *wire gadget* that propagates the truth assignments; see Fig. 5(b–c). It consists of a sequence of similar subgraphs, each containing



(a) (b)

Figure 6 (a) A clause gadget, the three bold segments represent that the corresponding literals are set to true. The central 7-gon can be augmented to a subgraph of degree at least five if and only if at least one literal is true. (b) The three possibilities to augment the 7-gon if one literal is true.

four vertices of degree four (the other vertices have at least degree five). The main idea is that we need to add an edge to each of the vertices of degree four surrounding the big gray squares. But due to blocked visibilities this can only be achieved by a "windmill" pattern which has to align with the neighboring parts. Thus, we have exactly two ways to add edges in order to augment the wire to a graph with minimum degree five. The truth assignment of a wire can be duplicated by the *split gadget* shown in Fig. 5(d). The *clause gadget* is illustrated in Fig. 6(a). The wires enter from left, right and below (respectively above). The 7-gon in the middle of the clause gadget can be augmented to a subgraph with minimum degree five if and only if it is connected to at least one wire in the true state. See also Fig. 6(b). ◀

Acknowledgments. We thank Kevin Buchin, Michael Hoffmann, Wolfgang Mulzer and Nadja Seiferth for helpful discussions.

References

1 Oswin Aichholzer, Sergey Bereg, Adrian Dumitrescu, Alfredo García, Clemens Huemer, Ferran Hurtado, Mikio Kano, Alberto Márquez, David Rappaport, Shakhar Smorodinsky, Diane Souvaine, Jorge Urrutia, and David R. Wood. Compatible geometric matchings. *Comput. Geom.*, 42(6-7):617–626, 2009. doi:10.1016/j.endm.2008.06.040.

2 Oswin Aichholzer, Alfredo García, Ferran Hurtado, and Javier Tejel. Compatible matchings in geometric graphs. In *Proc. XIV Encuentros de Geometría Computacional*, pages 145–148, Alcalá, Spain, 2011.

3 Hugo A. Akitaya, Matias Korman, Mikhail Rudoy, Diane L. Souvaine, and Csaba D. Tóth. Circumscribing polygons and polygonizations for disjoint line segments. In *Proc. of the 35th International Symposium on Computational Geometry (SoCG 2019)*, volume 129 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 9, 17. Schloss Dagstuhl, 2019. doi:10.4230/LIPIcs.SoCG.2019.9.

4 Sang Won Bae, Mark de Berg, Otfried Cheong, Joachim Gudmundsson, and Christos Levcopoulos. Shortcuts for the circle. *Comput. Geom.*, 79:37–54, 2019. doi:10.1016/j.comgeo.2019.01.006.

5 Nathann Cohen, Daniel Gonçalves, Eun Jung Kim, Christophe Paul, Ignasi Sau, Dimitrios M. Thilikos, and Mathias Weller. A polynomial-time algorithm for outerplanar diameter improvement. *J. Comput. System Sci.*, 89:315–327, 2017. doi:10.1016/j.jcss.2017.05.016.

6 Mark de Berg and Amirali Khosravi. Optimal binary space partitions in the plane. In *Proc. of the 16th Annual International Conference on Computing and Combinatorics (COCOON*

2010), volume 6196 of *Lecture Notes in Computer Science*, pages 216–225, 2010. doi:10.1007/978-3-642-14031-0_25.

7 Jean-Lou De Carufel, Carsten Grimm, Stefan Schirra, and Michiel Smid. Minimizing the continuous diameter when augmenting a tree with a shortcut. In *Proc. of the Algorithms and Data Structures Symposium (WADS 2017)*, volume 10389 of *Lecture Notes in Comput. Sci.*, pages 301–312. Springer, Cham, 2017. doi:10.1007/978-3-319-62127-2_26.

8 Ferran Hurtado and Csaba D. Tóth. Plane geometric graph augmentation: a generic perspective. In *Thirty essays on geometric graph theory*, pages 327–354. Springer, New York, 2013. doi:10.1007/978-1-4614-0110-0_17.

9 Mashhood Ishaque, Diane L. Souvaine, and Csaba D. Tóth. Disjoint compatible geometric matchings. *Discrete & Computational Geometry*, 49(1):89–131, 2013. doi:10.1007/s00454-012-9466-9.

10 Klaus Jansen. One strike against the min-max degree triangulation problem. *Comput. Geom.*, 3:107–120, 1993. doi:10.1016/0925-7721(93)90003-O.

11 Wolfgang Mulzer and Günter Rote. Minimum-weight triangulation is NP-hard. *J. ACM*, 55(2):11:1–11:29, 2008. doi:10.1145/1346330.1346336.

12 Alexander Pilz. Augmentability to cubic graphs. In *Proc. 28th European Workshop on Computational Geometry (EuroCG 2012)*, pages 29–32, Assisi, Italy, March 2012.

13 David Rappaport. Computing simple circuits from a set of line segments is NP-complete. *SIAM J. Comput.*, 18(6):1128–1139, 1989. doi:10.1137/0218075.

14 David Rappaport, Hiroshi Imai, and Godfried T. Toussaint. On computing simple circuits on a set of line segments. In Alok Aggarwal, editor, *Proc. of the Proceedings of the 2nd Annual Symposium on Computational Geometry (SoCG 1986)*, pages 52–60. ACM, 1986. doi:10.1145/10515.10521.

Covering a set of line segments with a few squares

Joachim Gudmundsson[1], Mees van de Kerkhof[2], André van Renssen[3], Frank Staals[4], Lionov Wiratma[5], and Sampson Wong[6]

1 **University of Sydney, Australia**
`joachim.gudmundsson@sydney.edu.au`
2 **Utrecht University, Netherlands**
`m.a.vandekerkhof@uu.nl`
3 **University of Sydney, Australia**
`andre.vanrenssen@sydney.edu.au`
4 **Utrecht University, Netherlands**
`f.staals@uu.nl`
5 **Utrecht University, Netherlands**
`l.wiratma@uu.nl`
6 **University of Sydney, Australia**
`swon7907@uni.sydney.edu.au`

Abstract

We study the problem of covering a set of line segments with a few (up to four) axis-parallel, unit-sized squares in the plane. Covering line segments with two squares has been previously studied, however, little is known for three or more squares. Our original motivation for the line segment covering problem comes from trajectory analysis. We study two trajectory covering problems: a data structure on the trajectory that efficiently answers whether a query subtrajectory is coverable, and an algorithm to compute its longest coverable subtrajectory.

1 Introduction

Geometric covering problems are a classic area of research in computational geometry. The traditional *geometric set cover problem* is to decide whether one can place k axis-parallel unit-sized squares to cover n points in the plane. If k is part of the input, the problem is known to be NP-hard [3, 7]. Thus, efficient algorithms are only known for small values of k. For $k = 2$ or 3, there are linear time algorithms [2, 11], and for $k = 4$ or 5, there are $O(n \log n)$ time algorithms [8, 10].

Motivated by trajectory analysis we study a line segment variant of the geometric set cover problem where the input is a set of n line segments, see Figure 1.

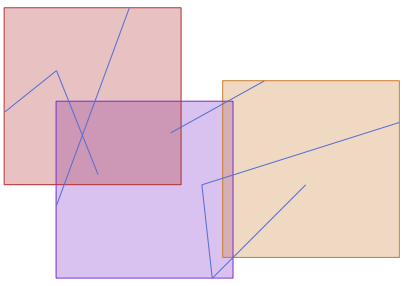


Figure 1 An example a set of n segments which is 3-coverable.

▶ Problem 1. Decide if a set of line segments is k-coverable, for $k = 2, 3, 4$.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

A key difference in the line segment variant is that each segment need not be covered by a single square, as long as the k squares together cover all n segments. Sadhu et al. [9] recently provided a linear time algorithm for $k = 2$. Hoffman [6] provides a linear time algorithm for $k = 3$, however the proof was not included in the extended abstract. We provide a proof for a $k = 3$ algorithm and a new $O(n \log n)$ time algorithm for $k = 4$.

Next, we study trajectory coverings, which was the original motivation for studying line segment coverings. A trajectory is a polygonal curve in the plane parametrised by time, and models the movement of an object through time and space. Trajectory analysis has been used to study data sets in animal ecology [1], meteorology [12] and sports analytics [4]. Our trajectory analysis task is to compute a small region where a moving object spends a large amount of time. Such a region is known as a *hotspot* and has applications in segmentation, clustering, or enriching trajectory data with locations of interest [5].

If the hotspot is modelled by k unit-sized axis-parallel squares, then the subtrajectory that it covers is a k-coverable subtrajectory. Formally, a *subtrajectory* is the trajectory restricted to a contiguous time window. Note that the start and end points of the subtrajectory need not be vertices of the original trajectory, see Figure 2. The two problems we study are:

▶ Problem 2. Construct a data structure on a trajectory, so that given any query subtrajectory, it can efficiently answer whether the subtrajectory is k-coverable, for $k = 2, 3$.

▶ Problem 3. Given a trajectory, compute its longest k-coverable subtrajectory, for $k = 2$.

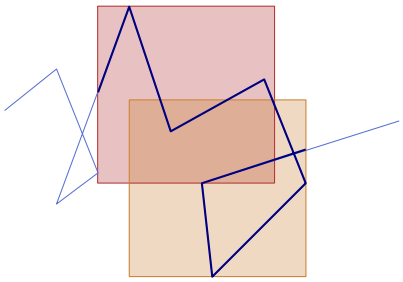


Figure 2 A trajectory (thin) and its longest 2-coverable subtrajectory (thick).

Previous work focuses on hotspots regions modelled by a single square [5]. To the best of our knowledge, this paper is the first to study k-coverable subtrajectories for $k \geq 2$.

The results of this paper and their relevant sections are summarised in Table 1.

	$k = 2$	$k = 3$	$k = 4$
Problem 1	Section 2.1	Section 2.2	Section 2.3
Problem 2	Section 3.1	Section 3.2	
Problem 3	Section 4.1		

Table 1 The results of this paper and their relevant sections.

2 Problem 1: The Decision Problem

2.1 Is a set of segments 2-coverable?

This section restates known results which are useful for the recursive step in Section 2.2 and for the data structure in Section 3.1. Recall that a bounding box of a set of segments is the

smallest axis-aligned rectangle that contains all segments. Observation 1 uses the bounding box to decide if a set of segments is 2-coverable:

▶ **Observation 1.** *A set of segments is 2-coverable if and only if there is a covering with squares in opposite corners of the bounding box of the set of segments.*

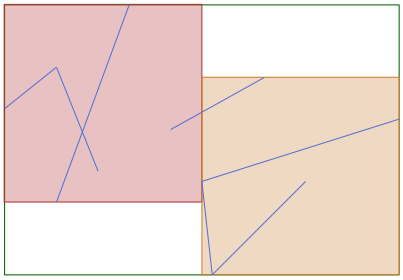


Figure 3 A covering with squares in opposite corners of the bounding box of the set of segments.

Observation 1 is due to Sadhu et al. [9]. See Figure 3 for an illustration. The algorithm is therefore to compute the bounding box, and for each pair of opposite corners, to check if squares placed in these corners cover every segment. This takes linear time in total. Thus:

▶ **Theorem 2.** *One can decide whether a set of segments is 2-coverable in $O(n)$ time.*

2.2 Is a set of segments 3-coverable?

We start off with a similar observation to Observation 1, but for 3-coverable segments:

▶ **Observation 3.** *A set of segments is 3-coverable if and only if there is a covering with a square in a corner of the bounding box of the set of segments.*

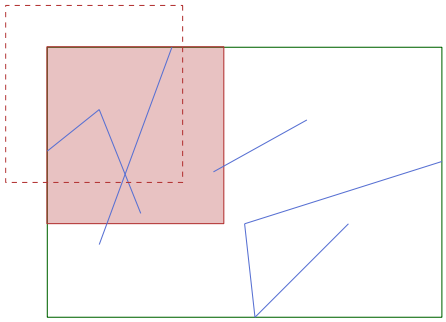


Figure 4 A partial covering with a square in a corner of the bounding box of the set of segments.

Proof. Any 3-covering touches all four sides of the bounding box, in fact, one square in the 3-covering must touch at least two sides. If the two sides are opposite then the bounding box has width less than or equal to one and there is a straightforward one-dimensional algorithm. Otherwise, without loss of generality let the adjacent sides be the top and left sides. If the square is not already in the topleft corner of the bounding box, move the square

to the corner position shown in Figure 4. The new square covers more than the previous position, so the modified positions is 3-covering with a square in a corner. ◀

Now, we can use Observation 3 to place the first square into one of four possible positions. We compute up to two uncovered subsegments for each original segment. The set of uncovered subsegments has linear complexity. We then use Theorem 2 to recursively check if the uncovered subsegments are 2-coverable. All steps in this algorithm take linear time.

▶ **Theorem 4.** *One can decide whether a set of segments is 3-coverable in $O(n)$ time.*

2.3 Is a set of segments 4-coverable?

For any 4-covering, each side of the bounding box is touched by one of the 4-covering squares. We make a similar observation to $k = 2$ and $k = 3$ but we have two cases: either we have a square which touches two sides, or each square touches exactly one side.

▶ **Observation 5.** *A set of segments is 4-coverable if and only if one of the following hold:*

- *There is a covering with a square in a corner of the bounding box.*
- *There is a covering with each square touching exactly one side of the bounding box.*

If the covering has a square in a corner of the bounding box, we can use the same strategy as in the $k = 3$ case and try all four positions of the first square and then recurse. Hence, for the remainder of this section, we assume we are in the second case.

▶ **Definition 6.** Define L, B, T and R to be the square that touches the left, bottom, top and right sides of the bounding box respectively. See Figure 5.

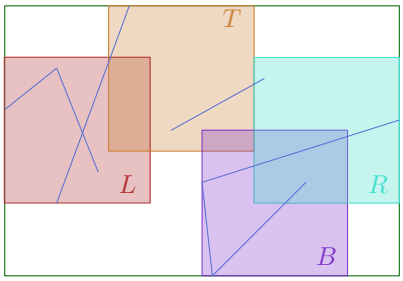


Figure 5 The squares L, T, B, R touch the left, top, bottom and right sides of the bounding box.

Without loss of generality, suppose that T is to the left of B. Then the left to right order of the squares is L, T, B, R. Suppose for now there were a way to compute the initial placement of L. The rest of the algorithm would be as follows. Compute the remaining uncovered subsegments, then place T in its topleft corner of its bounding box. We can do this since T is the leftmost and topmost of the remaining squares. Next, compute the remaining uncovered subsegments, then place B in the bottomright corner of its bounding box. Finally, cover the remaining uncovered subsegments with R, if possible.

Our approach is to simulate the above algorithm for variable positions of L. Let y_L be the y-coordinate of the top side of L, and let x_T be the x-coordinate of the left side of T.

▶ **Lemma 7.** *The variable x_T as a function of variable y_L is piecewise linear and can be computed in $O(n \log n)$ time.*

Proof. We know from our algorithm above that x_T is the leftmost uncovered point of the set of segments after placing L. We divide the uncovered region inside the bounding box into three separate regions: A above L, B below L and C to the right of L. See Figure 6. By definition, all these regions are uncovered. We consider the leftmost points of A, B and C separately, and then compute their minimum to obtain x_T.

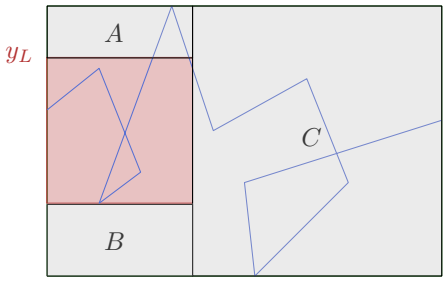


Figure 6 The region above, below, and to the right of L are A, B and C respectively.

As y_L increases, the leftmost point of A moves monotonically to the right. This polygonal curve is shown in Figure 7, and we call this curve the *skyline* of the set of segments. In the full version we show an $O(n \log n)$ divide and conquer algorithm to compute the skyline.

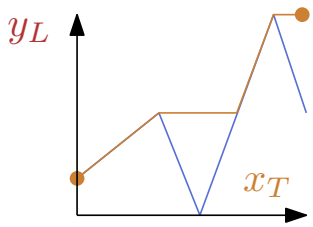


Figure 7 The leftmost point x_T as a function of y_L forming a "skyline".

A similar algorithm for the skyline gives the function for the leftmost point in B as a function of y_L. The leftmost point of C does not depend on y_L and can be computed in linear time. The value of x_T is the minimum value of the leftmost points of A, B and C. Since each is piecewise linear and can be computed in $O(n \log n)$ time, so is their minimum. ◀

We define x_B to be the leftmost uncovered point after placing L at y_L and T at x_T. In the full version of this paper, we show that x_B is a piecewise linear function of y_L and can be computed in $O(n \log n)$ time. Additional skylines need to be computed but a similar approach works. The same is also true for y_{R_1} and y_{R_2}, which are defined to be the topmost and bottommost points after placing L, T and B at y_L, x_T and x_B respectively. Finally, we check if there exists a y_L so that $y_{R_1} - y_{R_2}$ attains a value less than or equal to 1. If so, there is a position for L, T, B, R that covers the set of segments. This yields Theorem 8.

▶ **Theorem 8.** *One can decide whether a set of segments is 4-coverable in $O(n \log n)$ time.*

An open problem is to provide a polynomial time algorithm that decides if a set of segments is k-coverable for $k \geq 5$. Significantly new ideas are required as Observation 5 does not seem to extend to larger values of k.

3 Problem 2: The Subtrajectory Data Structure Problem

In the full version of this paper, we provide the details of the following data structures, i.e., their construction and query procedures and corresponding running times. Given a piecewise linear trajectory of complexity n, we build:

▶ **Tool 9.** *A bounding box data structure that preprocesses a trajectory in $O(n)$ time, so that given a query subtrajectory, returns its bounding box in $O(\log n)$ time.*

▶ **Tool 10.** *An upper envelope data structure that preprocesses a trajectory in $O(n \log n)$ time, so that given a query subtrajectory and a query vertical line, returns the highest intersection between the subtrajectory and the vertical line (if one exists) in $O(\log n)$ time.*

▶ **Tool 11.** *A highest vertex data structure that preprocesses a trajectory in $O(n \log n)$ time, so that given a query subtrajectory and a query axis-parallel rectangle, returns the highest vertex of the subtrajectory inside the rectangle (if one exists) in $O(\log n)$ time.*

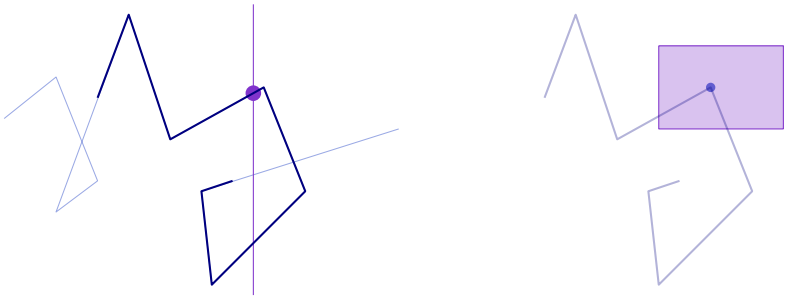


Figure 8 Tool 10 (left) returns the highest intersection between a vertical line and a subtrajectory. Tool 11 (right) returns the highest subtrajectory vertex in a query rectangle.

3.1 Is a query subtrajectory 2-coverable?

Our data structure is built like so. At preprocessing, we construct Tools 9 and 10 in $O(n \log n)$ time. At query time, we use Tool 9 to compute the bounding box of the subtrajectory in $O(\log n)$ time. By Observation 1, the subtrajectory is coverable only if there is a covering with squares in opposite corners of the bounding box. Consider the union of the two squares in opposite corners of the bounding box, as shown in the left diagram of Figure 9.

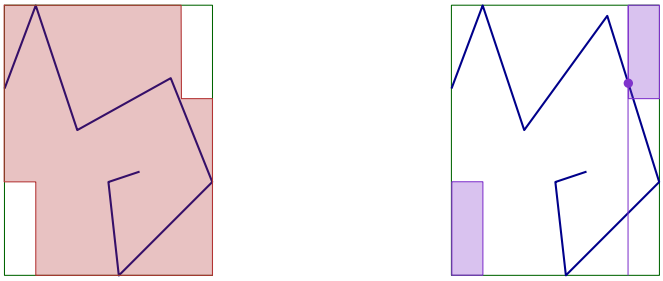


Figure 9 The union of the pair of squares (left) and its complement (right).

To check if this pair of squares covers the subtrajectory, we only need to check if its complement is empty. The complement is shaded in purple in Figure 9. We use Tool 10 to return the highest intersection of the subtrajectory with one of the sides of the purple rectangle, as shown in the same figure. This detects whether there is a subtrajectory edge which pierces any side of any of the purple rectangles. We check all sides of the rectangles in $O(\log n)$ time, and do this for both choices of pairs of opposite corners. Thus:

▶ **Theorem 12.** *There exists a data structure that preprocesses a trajectory in $O(n \log n)$ time, so that given a query subtrajectory, it answers whether the subtrajectory is 2-coverable in $O(\log n)$ time.*

3.2 Is a query subtrajectory 3-coverable?

We provide a sketch of the algorithm for $k = 3$, see the full version for details. In preprocessing time, we construct Tools 9, 10 and 11. At query time, we use Tool 9 to compute the bounding box of the subtrajectory. By Observation 3 there is a square in one of the corners of the bounding box. We then use a combination of Tool 10 and 11 to compute the bounding box of the uncovered subsegments. See Figure 10.

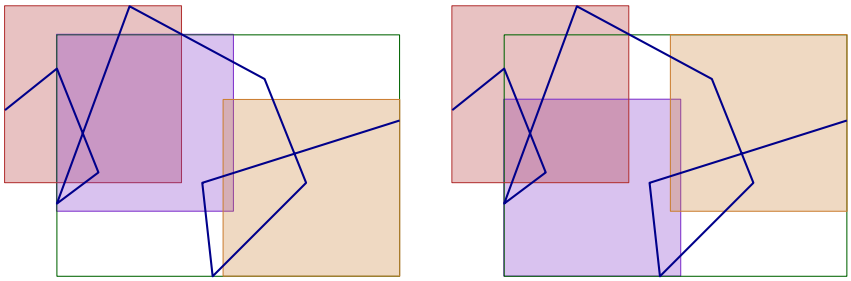


Figure 10 The two choices for the last two squares, as given by their bounding box (thin green).

The bounding box of the uncovered subsegments gives two possible positions for the last two squares. It remains only to check if the three squares cover the subtrajectory. For this final step we use Tool 10 to check for any edges piercing the boundary of the union of the three squares. We require one final check using Tool 11 to detect if the trajectory exits the top of the purple square in the right diagram of Figure 10. Putting this together yields:

▶ **Theorem 13.** *There exists a data structure that preprocesses a trajectory in $O(n \log n)$ time, so that given a query subtrajectory, it answers whether the subtrajectory is 3-coverable in $O(\log n)$ time.*

4 Problem 3: The Longest Coverable Subtrajectory Problem

4.1 The longest 2-coverable subtrajectory

Suppose we were given the starting point of the longest coverable subtrajectory. Then by parametric search in conjunction with the data structure in Theorem 12, we can compute the latest ending point so that the subtrajectory is still coverable.

Hence, the problem reduces to finding a set of points so that the starting point of the longest coverable subtrajectory is guaranteed to be inside the set. In the full version of the paper, we show that there is a set of size $O(n2^{\alpha(n)})$ that can be computed in $O(n2^{\alpha(n)} \log^2 n)$ time that is guaranteed to contain the optimal starting point. Hence, we have:

▶ **Theorem 14.** *There is an $O(n2^{\alpha(n)}\log^2 n)$ time algorithm to compute the longest 2-coverable subtrajectory of the trajectory.*

An open problem is whether there is a polynomial time algorithm to compute the longest 3-coverable subtrajectory of a given trajectory. New ideas are required to compute the set of additional starting points for 3-coverable subtrajectories.

References

1. Maria Luisa Damiani, Hamza Issa, and Francesca Cagnacci. Extracting stay regions with uncertain boundaries from GPS trajectories: A case study in animal ecology. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 253–262, 2014.
2. Zvi Drezner. On the rectangular p-center problem. *Naval Research Logistics (NRL)*, 34(2):229–234, 1987.
3. Robert J. Fowler, Mike Paterson, and Steven L. Tanimoto. Optimal packing and covering in the plane are NP-complete. *Information Processing Letters*, 12(3):133–137, 1981.
4. Joachim Gudmundsson and Michael Horton. Spatio-temporal analysis of team sports. *ACM Computing Surveys (CSUR)*, 50(2):22, 2017.
5. Joachim Gudmundsson, Marc van Kreveld, and Frank Staals. Algorithms for hotspot computation on trajectory data. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 134–143, 2013.
6. Michael Hoffmann. Covering polygons with few rectangles. In *Abstracts 17th European Workshop Computational Geometry*, pages 39–42, 2001.
7. Nimrod Megiddo and Kenneth J. Supowit. On the complexity of some common geometric location problems. *SIAM Journal of Computing*, 13(1):182–196, 1984.
8. Doron Nussbaum. Rectilinear p-piercing problems. In *Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation, ISSAC*, pages 316–323, 1997.
9. Sanjib Sadhu, Sasanka Roy, Subhas C. Nandy, and Suchismita Roy. Linear time algorithm to cover and hit a set of line segments optimally by two axis-parallel squares. *Theoretical Computer Science*, 769:63–74, 2019.
10. Michael Segal. On piercing sets of axis-parallel rectangles and rings. *International Journal of Computional Geometry and Applications*, 9(3):219–234, 1999.
11. Micha Sharir and Emo Welzl. Rectilinear and polygonal p-piercing and p-center problems. In *Proceedings of the 12th Annual Symposium on Computational Geometry*, pages 122–132, 1996.
12. Andreas Stohl. Computation, accuracy and applications of trajectories—A review and bibliography. *Developments in Environmental Science*, 1:615–654, 2002.

Monotone Arc Diagrams with few Biarcs*

Steven Chaplick[†1], Henry Förster[2], Michael Hoffmann[‡3], and Michael Kaufmann[2]

1 **Universität Würzburg, Germany and Maastricht University, the Netherlands**
`s.chaplick@maastrichtuniversity.nl`
2 **Universität Tübingen, Germany**
`{foersth,mk}@informatik.uni-tuebingen.de`
3 **Department of Computer Science, ETH Zürich, Switzerland**
`hoffmann@inf.ethz.ch`

Abstract

We show that every planar graph can be represented by a monotone topological 2-page book embedding where at most $15n/16$ (of potentially $3n-6$) edges cross the spine exactly once.

1 Introduction

Arc diagrams (Figure 1) are drawings of graphs that represent vertices as points on a horizontal line, called *spine*, and edges as *arcs*, consisting of a sequence of halfcircles centered on the spine. A *proper arc* consists of one single halfcircle. In *proper arc diagrams* all arcs are proper. In *plane* arc diagrams no two edges cross. Note that plane proper arc diagrams are also known as 2-*page book embeddings* in the literature. Bernhard and Kainen [2] characterized the graphs admitting plane proper arc diagrams: subhamiltonian planar graphs, i.e., subgraphs of planar graphs with a Hamiltonian cycle. In particular, non-Hamiltonian maximal planar graphs do not admit plane proper arc diagrams.

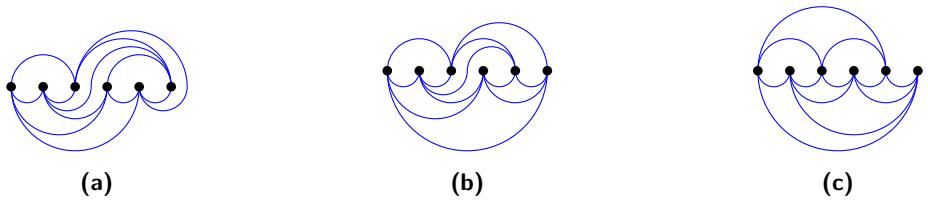


Figure 1 Arc diagram (a), monotone arc diagram (b), proper arc diagram (c) of the octahedron.

To represent all planar graphs, it suffices to allow each edge to cross the spine at most once [9]. The resulting arcs composed of two halfcircles are called *biarcs* (see Figure 1a). Additionally, all edges can be drawn as *monotone* curves w.r.t. the spine [6]; such a drawing is called a *monotone topological (2-page) book embedding*. A monotone biarc is either *down-up* or *up-down*, depending on if the left halfcircle is drawn above or below the spine, respectively. Note that a *monotone topological* 2-page book embedding is not necessarily a 2-page book embedding even though the terminology suggests it.

In general, biarcs are needed, but *some* edges can be drawn as proper arcs. Cardinal et al. [3] gave bounds on the required number of biarcs showing that every planar graph on

* This work started at the workshop *Graph and Network Visualization* 2017. We would like to thank Stefan Felsner and Stephen Kobourov for useful discussions.
† Partially supported by DFG grant WO 758/11-1.
‡ Supported by the Swiss National Science Foundation within the collaborative DACH project *Arrangements and Drawings* as SNSF Project 200021E-171681.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

$n \geq 3$ vertices admits a plane arc diagram with at most $\lfloor (n-3)/2 \rfloor$ biarcs (not necessarily monotone). They also described a family of planar graphs on $n_i = 3i+8$ vertices that cannot be drawn as a plane biarc diagram using less than $(n_i-8)/3$ biarcs for $i \in \mathbb{N}$. However, they use arbitrary biarcs. When requiring only monotone biarcs, Di Giacomo et al. [6] gave an algorithm to construct a monotone plane arc diagram that may create close to $2n$ biarcs for an n-vertex planar graph. Cardinal et al. [3] improved this bound to at most $n-4$ biarcs.

Results. As a main result, we improve the upper bound on the number of monotone biarcs:

▶ **Theorem 1.1.** *Every n-vertex planar graph admits a plane arc diagram with at most $\lfloor \frac{15}{16}n - \frac{5}{2} \rfloor$ biarcs that are all down-up monotone. Such a diagram is computable in $O(n)$ time.*

For general arc diagrams, $\lfloor (n-8)/3 \rfloor$ biarcs may be needed [3], but it is conceivable that this number increases for monotone biarcs. We investigated the lower bound with a SAT based approach (based on [1]), with the following partial result; details will appear in the full version.

▶ **Observation 1.2.** *Every Kleetope on $n' = 3n-4$ vertices derived from triangulations of $n \leq 14$ vertices admits a plane arc diagram with $\lfloor (n'-8)/3 \rfloor$ monotone biarcs.*

Note that a *Kleetope* is derived from a planar triangulation T by inserting a new vertex v_f into each face f of T and then connecting v_f to the three vertices bounding f.

Related Work. Giordano et al. [8] showed that every upward planar graph admits an *upward topological book embedding* where edges are either proper arcs or biarcs. One of their directions for future work is to minimize the number of spine crossings. Note that these embeddings are monotone arc diagrams with at most one spine crossing per edge respecting the orientations of the edges. Everett et al. [7] used monotone arc diagrams with only down-up biarcs to construct small universal point sets for 1-bend drawings of planar graphs. This result was extended by Löffler and Tóth [10] by restricting the set of possible bend positions. They use monotone arc diagrams with at most $n-4$ biarcs to build universal points set of size $6n-10$ (vertices and bend points) for 1-bend drawings of planar graphs on n vertices. Using Theorem 1.1, we can slightly decrease the number of points by approximately $n/16$.

2 Overview of our Algorithm

To prove Theorem 1.1 we describe an algorithm to incrementally construct an arc diagram for a given planar graph G on n vertices. W.l.o.g. we assume that G is a (combinatorial) *triangulation*, i.e., a maximal planar graph. Our algorithm is a (substantial) refinement of the algorithm of Cardinal et al., which is based on the notion of a canonical ordering. A canonical ordering is defined for an *embedded* triangulation. Every triangulation on $n \geq 4$ vertices is 3-connected, so selecting one facial triangle as the *outer face* embeds it into the plane which determines a unique outer face (cycle) for every biconnected subgraph. A *canonical ordering* [5] of an embedded triangulation G is a total order of vertices $v_1, \ldots, v_n$ s.t.

- for each $i \in \{3, \ldots, n\}$, the induced subgraph $G_i = G[\{v_1, \ldots, v_i\}]$ is biconnected and internally triangulated (i.e., every inner face is a triangle);
- for each $i \in \{3, \ldots, n\}$, (v_1, v_2) is an edge of the outer face C_i of G_i;
- for each $i \in \{3, \ldots, n-1\}$, v_{i+1} lies in the interior of C_i and the neighbors of v_{i+1} in G_i form a sequence of consecutive vertices along the boundary of C_i.

Every triangulation admits a canonical ordering [5] and one can be computed in $O(n)$ time [4]. We say that a vertex v_i *covers* an edge e (a vertex v, resp.) if and only if e (v, resp.) is an edge (vertex, resp.) on C_{i-1} but not an edge (vertex, resp.) on C_i.

We iteratively process the vertices in a canonical order $v_1, \ldots, v_n$. Every vertex v_i arrives with α credits that we can either spend to create biarcs (at a cost of one credit per biarc) or distribute on edges of the outer face C_i for later use. We prove our claimed bound by showing that each biarc drawn can be paid for s.t. at least seven credits remain in total.

There are two types of proper arcs: *mountains* (above the spine) and *pockets* (below the spine). The following invariants hold after processing vertex v_i, for every $i \in \{3, \ldots, n\}$.

(I1) Every edge is either a proper arc or a down-up biarc.
(I2) Every edge of C_i is a proper arc. Vertex v_1 is the leftmost and v_2 is the rightmost vertex of G_i. Edge (v_1, v_2) forms the lower envelope of G_i, i.e., no point of the drawing is vertically below it. The other edges of C_i form the upper envelope of G_i.
(I3) Every mountain whose left endpoint is on C_i carries 1 credit.
(I4) Every pocket on C_i carries π credits, for some constant $\pi \in (0, 1)$.
(I5) Every biarc in G_i carries (that is, is paid for with) 1 credit.

Usually, we insert v_i between its leftmost neighbor ℓ_i and rightmost neighbor r_i along C_{i-1}. The algorithm of Cardinal et al. [3] gives a first upper bound on the insertion costs.

▶ **Lemma 2.1.** *If v_i covers at least one pocket, then we can insert v_i maintaining (I1) to (I5) using ≤ 1 credit. If $\deg_{G_i}(v_i) \geq 4$, then $1 - \pi$ credits are enough.*

Proof (Sketch). We place v_i in the rightmost covered pocket and pay for at most 1 mountain; see Figures 2a and 2b. If $\deg_{G_i}(v_i) \geq 4$, at least 1 covered pocket's credits is free. ◀

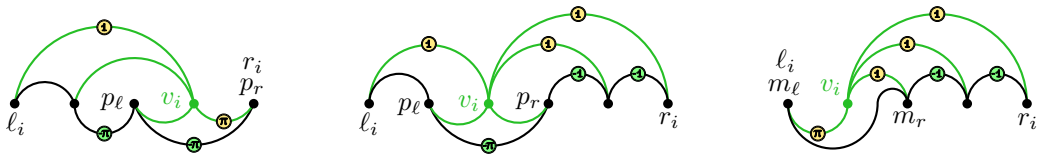


Figure 2 Inserting a vertex v_i using $1 - \pi$, 1, and $1 + \pi$ credits, resp. (Lemma 2.1–2.2).

▶ **Lemma 2.2.** *If v_i covers mountains only, then we can insert v_i maintaining (I1) to (I5) using $\leq 1 + \pi$ credits. If $\deg_{G_i}(v_i) \geq 4$, then $5 - \deg_{G_i}(v_i)$ credits suffice.*

Proof (Sketch). If $\deg_{G_i}(v_i) < 4$, we push down the leftmost mountain and place v_i on the created biarc paying for 1 mountain and 1 pocket each; see Figure 2c. If $\deg_{G_i}(v_i) \geq 4$, we push down the rightmost mountain saving the credit of a covered mountain; see Figure 3. ◀

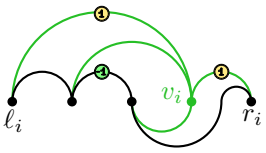


Figure 3 An alternative drawing to insert a degree four vertex.

Full proofs of Lemmas 2.1 and 2.2 will appear in the full version. We only steal credits from arcs on C_{i-1} in both proofs. If left endpoints of proper arcs not on C_{i-1} are covered, there is slack.

In the following, we prove that we can choose $\pi = 1/8$, so that to achieve the bound of Theorem 1.1, we insert a vertex at an average cost of $1-\pi/2$. Lemmas 2.1 and 2.2 guarantee this bound only in certain cases, e.g., a sequence of three degree two (in G_i) vertices stacked onto mountains costs $1+\pi$ per vertex and produces three biarcs, see Figure 4a. A symmetric scheme with up-down biarcs realizes the same graph with one biarc; see Figure 4b.

Figure 4 A sequence of degree two vertices in forward (a) and reverse drawing (b).

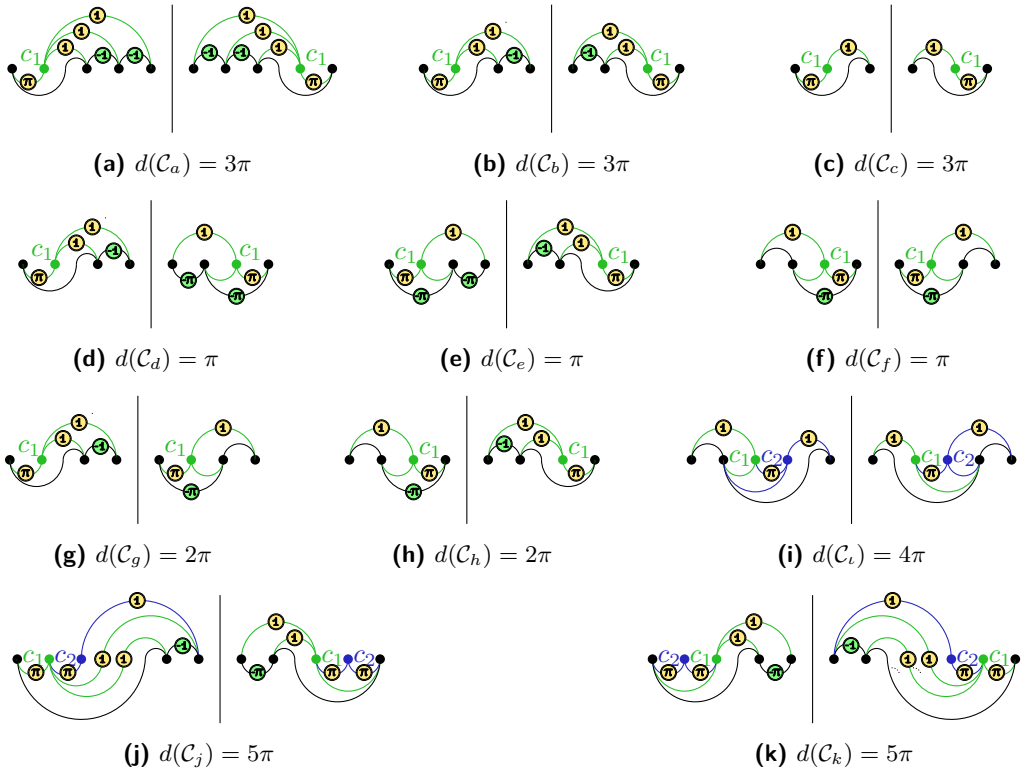
To exploit this behavior, we consider the instance in both a *forward drawing*, using only proper arcs and down-up biarcs, and a *reverse drawing* that uses only proper arcs and up-down biarcs (and so (I1) and (I3) appear in a symmetric formulation). Out of the two resulting arc diagrams, we choose one with a fewest number of biarcs. To prove Theorem 1.1, we need to insert a vertex at an average cost of $\alpha = 2 - \pi$ credits into both diagrams.

The outer face, a sequence of pockets and mountains, can evolve differently in both drawings because edges covered by a vertex may not be drawn the same way in both drawings. Further, it does not suffice to consider a single vertex in isolation. For instance, consider a degree three vertex inserted above two mountains in both the forward and reverse drawings; see Figure 5b. In each drawing, this costs $1 + \pi$ credits, or $2(1 + \pi)$ in total. W.r.t. our target value $\alpha = 2 - \pi$, these costs incur a *debt* of 3π credits. Indeed, there are several such *open configurations*, listed in Figure 5, for which our basic analysis does not suffice.

Each open configuration $\mathcal{C}$ consists of up to two adjacent vertices on the outer face whose insertion incurred a debt and their incident edges. It specifies the drawing of these edges, as pocket, mountain, or biarc in forward and in reverse drawing, as well as the drawing of the edges covered by the vertices of the open configuration. When a vertex v_i covers (part of) an open configuration, we may alter the placement of the vertices and/or draw the edges of the open configuration differently. The associated debt $d(\mathcal{C})$ is the amount of credits paid in addition to α credits per vertex. As soon as any arc of an open configuration is covered, the debt must be paid or transferred to a new open configuration. We enhance our collection of invariants as follows.

(I6) A sequence of consecutive arcs on C_i may be associated with a debt. Each arc is part of at most one open configuration; refer to Figure 5 for a full list of such configurations.

To prove Theorem 1.1 we show that the credit total carried by arcs in both drawings minus the total debt of all open configurations does not exceed $\alpha i - 5$ after inserting v_i.



(a) $d(\mathcal{C}_a) = 3\pi$ **(b)** $d(\mathcal{C}_b) = 3\pi$ **(c)** $d(\mathcal{C}_c) = 3\pi$

(d) $d(\mathcal{C}_d) = \pi$ **(e)** $d(\mathcal{C}_e) = \pi$ **(f)** $d(\mathcal{C}_f) = \pi$

(g) $d(\mathcal{C}_g) = 2\pi$ **(h)** $d(\mathcal{C}_h) = 2\pi$ **(i)** $d(\mathcal{C}_\iota) = 4\pi$

(j) $d(\mathcal{C}_j) = 5\pi$ **(k)** $d(\mathcal{C}_k) = 5\pi$

Figure 5 The set of open configurations. Each subfigure shows the forward drawing (left) and the reverse drawing (right) and is captioned by the debt incurred.

3 Default insertion of a vertex v_i

If v_i does not cover any arc of an open configuration, we use procedures from Lemmas 2.1 and 2.2. If $\deg_{G_i}(v_i) \geq 4$ and v_i covers any pocket in either drawing, by Lemmas 2.1 and 2.2 the insertion costs are at most $2 - \pi = \alpha$. If $\deg_{G_i}(v_i) \geq 5$ and v_i only covers mountains in both drawings, the costs are 0 (Lemma 2.2). If $\deg_{G_i}(v_i) = 4$ and v_i covers mountains only in both drawings, we obtain the open configuration in Figure 5a with cost $2 + 2\pi$ and debt 3π.

If $\deg_{G_i}(v_i) = 2$ and v_i covers a pocket in one drawing, insertion in this drawing costs π resulting in total cost $\leq 1 + 2\pi$ or at most α if $\pi \leq 1/3$. If $\deg_{G_i}(v_i) = 2$ and v_i covers only mountains, we have the open configuration in Figure 5c with cost $2 + 2\pi$ and debt 3π.

It remains to consider $\deg_{G_i}(v_i) = 3$. There are four pocket-mountain configurations for two arcs of G_{i-1} covered by v_i: MM, MP, PM, and PP (using M for mountain and P for pocket). Pattern PP costs $1 - \pi$, pattern MM costs $1 + \pi$. Each drawing has its favorite mixed pattern (PM for forward and MP for reverse) with cost 0; the other pattern costs 1.

There is only one *forward*|*reverse* combination, $MM|MM$, with cost $2+2\pi$ and debt 3π, leading to the open configuration in Figure 5b. Two combinations, $MM|PM$ and $MP|MM$, have cost $2+\pi$ and debt 2π resulting in open configurations in Figure 5g and 5h, resp. Also, the combinations $MM|PP$, $PP|MM$, and $MP|PM$ with costs 2 and a debt π lead to open configurations in Figure 5d, 5e, and 5f, resp. All other combinations cost at most α.

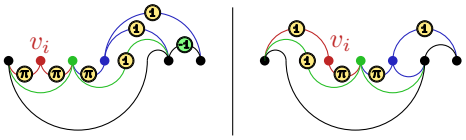


Figure 6 Alternative drawing to handle an open configuration $\mathcal{C}_\iota$ for $\deg_{G_i}(v_i) = 2$.

4 When and how to pay your debts

In this section, we describe the insertion of v_i if it covers an arc of an open configuration. Note that (1) every open configuration contains at least one mountain and at least one pocket in both drawings; (2) the largest debt incurred by one open configuration is 5π. Open configurations $\mathcal{C}_\iota, \mathcal{C}_j, \mathcal{C}_k$ (with highest debts) are introduced in the discussion below.

Case 1: $\deg_{G_i}(v_i) = 2$**.** If v_i covers a pocket of an open configuration $\mathcal{C}$ in either drawing, the insertion costs of $\pi + (1 + \pi)$ cover $d(\mathcal{C})$, as long as $1 + 2\pi + 5\pi \leq \alpha$, that is, $\pi \leq 1/8$. Assume v_i covers a mountain of open configuration $\mathcal{C}$ in both drawings; i.e., $\mathcal{C} \in \{\mathcal{C}_g, \mathcal{C}_h, \mathcal{C}_\iota\}$. If $\mathcal{C} \in \{\mathcal{C}_g, \mathcal{C}_h\}$, we obtain the open configurations in Figure 5j and 5k, resp., with cost $4 + 3\pi$ (for both vertices) and debt 5π. Otherwise $\mathcal{C} = \mathcal{C}_\iota$, and we use the drawings shown in Figure 6 (where v_i is inserted on the left mountain; the other case is symmetric). The costs are $2 + 3\pi$ (forward) and $3 + 2\pi$ (reverse), totaling $5 + 5\pi \leq 3\alpha$, for $\pi \leq 1/8$.

Case 2: $\deg_{G_i}(v_i) \geq 5$ **and Case 3:** $\deg_{G_i}(v_i) \in \{3, 4\}$**.** In the full version, we will discuss both cases in detail while we only mention the main ideas here. Each open configuration includes a mountain that can pay the debt if the configuration is entirely covered. We only focus on the left- and rightmost open configurations $\mathcal{C}_\ell$ and $\mathcal{C}_r$. In both cases, we mainly carefully move vertices v_i, c_1 and c_2 of $\mathcal{C}_r$ to avoid covered mountains from becoming biarcs, i.e., saving their credits.

5 Summary & Conclusions

Proof of Theorem 1.1. As previously shown, if $\pi \leq 1/8$, we maintain all invariants with α credits per vertex. G_3 is a triangle with two pockets on C_3 in both orientations, i.e. G_3 costs 4π. As v_1,v_2 and v_3 contribute 3α credits, there are $6 - 7\pi > 5$ unused credits after drawing G_3. If there remains an open configuration in G_n, there is a mountain with a credit paying its debt. Hence, the 5 unused credits of G_3's drawing remain. As a canonical ordering is computable in $O(n)$ time and we backtrack $O(1)$ steps if needed, the runtime follows. ◀

We proved the first upper bound of the form $c \cdot n$, with $c < 1$, for the total number of monotone biarcs in arc diagrams of n-vertex planar graphs. In our analysis, only three subcases require $\pi \leq 1/8$, i.e., a refinement may provide a better upper bound. Also, it remains open if there is a planar graph that requires more biarcs in a monotone arc diagram than in a general arc diagram. Finally, narrowing the gap between lower $\lfloor \frac{n-8}{3} \rfloor$ and upper $\lfloor \frac{15}{16}n - \frac{5}{2} \rfloor$ bounds would be interesting, particularly from the lower bound side.

References

1 Michael A. Bekos, Michael Kaufmann, and Christian Zielke. The book embedding problem from a SAT-solving perspective. In *Graph Drawing*, volume 9411 of *Lecture Notes in Computer Science*, pages 125–138. Springer, 2015.

2 Frank Bernhart and Paul C. Kainen. The book thickness of a graph. *J. Combin. Theory Ser. B*, 27:320–331, 1979. URL: http://dx.doi.org/10.1016/0095-8956(79)90021-2.

3 Jean Cardinal, Michael Hoffmann, Vincent Kusters, Csaba D. Tóth, and Manuel Wettstein. Arc diagrams, flip distances, and Hamiltonian triangulations. *Comput. Geom. Theory Appl.*, 68:206–225, 2018. URL: https://doi.org/10.1016/j.comgeo.2017.06.001.

4 Marek Chrobak and Thomas H. Payne. A linear-time algorithm for drawing a planar graph on a grid. *Inform. Process. Lett.*, 54:241–246, 1995. URL: http://dx.doi.org/10.1016/0020-0190(95)00020-D.

5 Hubert de Fraysseix, János Pach, and Richard Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10(1):41–51, 1990. URL: http://dx.doi.org/10.1007/BF02122694.

6 Emilio Di Giacomo, Walter Didimo, Giuseppe Liotta, and Stephen K. Wismath. Curve-constrained drawings of planar graphs. *Comput. Geom. Theory Appl.*, 30(1):1–23, 2005. URL: http://dx.doi.org/10.1016/j.comgeo.2004.04.002.

7 Hazel Everett, Sylvain Lazard, Giuseppe Liotta, and Stephen K. Wismath. Universal sets of n points for one-bend drawings of planar graphs with n vertices. *Discrete & Computational Geometry*, 43(2):272–288, 2010. URL: https://doi.org/10.1007/s00454-009-9149-3, doi:10.1007/s00454-009-9149-3.

8 Francesco Giordano, Giuseppe Liotta, Tamara Mchedlidze, Antonios Symvonis, and Sue Whitesides. Computing upward topological book embeddings of upward planar digraphs. *J. Discrete Algorithms*, 30:45–69, 2015. URL: https://doi.org/10.1016/j.jda.2014.11.006, doi:10.1016/j.jda.2014.11.006.

9 Michael Kaufmann and Roland Wiese. Embedding vertices at points: Few bends suffice for planar graphs. *J. Graph Algorithms Appl.*, 6(1):115–129, 2002. URL: http://dx.doi.org/10.7155/jgaa.00046.

10 Maarten Löffler and Csaba D. Tóth. Linear-size universal point sets for one-bend drawings. In *Graph Drawing*, volume 9411 of *Lecture Notes in Computer Science*, pages 423–429. Springer, 2015.

Colouring bottomless rectangles and arborescences

Dömötör Pálvölgyi[1] and Narmada Varadarajan[2]

1 **MTA-ELTE Lendület Combinatorial Geometry Research Group, Institute of Mathematics, Eötvös Loránd University (ELTE), Budapest, Hungary**
dom@cs.elte.hu

2 **MTA-ELTE Lendület Combinatorial Geometry Research Group, Institute of Mathematics, Eötvös Loránd University (ELTE), Budapest, Hungary**
narmadavrn@gmail.com

Abstract

We study problems related to colouring bottomless rectangles. We show that there is no number m and semi-online algorithm to colour a family of nested bottomless rectangles from below with a bounded number of colours such that every m-fold covered point is covered by at least two colours. We also prove several similar results that follow from a more abstract arborescence colouring problem, which is interesting on its own. Our key result is that there is no semi-online algorithm to colour the vertices of an arborescence without producing a long monochromatic path when the vertices are presented in a leaf-to-root order. The lower bounds are complemented with simple optimal upper bounds for semi-online algorithms from other directions.

1 Introduction

The systematic study of polychromatic colourings and cover-decomposition of geometric ranges was initiated by Pach over 30 years ago [5, 6]. The field has gained popularity in the new millennium, with several breakthrough results; for a (slightly outdated) survey, see [7], or see the up-to-date interactive webpage `http://coge.elte.hu/cogezoo.html` (maintained by Keszegh and the first author).

Our paper focuses on the colouring of one particular geometric family, known as *bottomless rectangles*. A subset of $\mathbb{R}^2$ is called a (closed) bottomless rectangle if it consists of the points $\{(x, y) \mid l \leq x \leq r, y \leq t\}$ for some parameters (l, r, t). These range spaces were first defined by Asinowski et al. [1], who showed that for any positive integer k, any finite set of points in $\mathbb{R}^2$ can be k-coloured such that any bottomless rectangle with at least $3k-2$ points contains all k colors. They also showed that the optimal number that can be written in place of $3k-2$ in the above statement is at least $1.67k$. Their upper bound giving $3k-2$ is a very neat *semi-online algorithm*.

Our paper studies the dual of the above problem. Our goal is to find the optimal m_k for which any finite collection of bottomless rectangles can be k-colored such that any m_k-fold covered region is covered by all k colors. About this question much less is known; the best upper bound $m_k = O(k^{5.09})$ is a corollary of a more general result [2] about *octants* (combined with an improvement of the base case [4] that slightly lowered the exponent). The general conjecture, however, is that $m_k = O(k)$ for any family [7]. It was also proved in [2] that there is no semi-online algorithm "from above" for colouring bottomless rectangles. An algorithm is said to colour bottomless rectangles "from above" if it is presented the rectangles in decreasing order of height; "from below" is defined analagously. Similarly, an algorithm is said to colour bottomless rectangles "from the left" if it is presented the rectangles in increasing order of *left* endpoint ($\rightarrow$), and "from the right" if it is presented the rectangles in decreasing order of *right* endpoint ($\leftarrow$). A *semi-online algorithm* is one where the vertices

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

are presented in some order, and the algorithm need not colour a vertex as soon as it is presented, but may not recolour previously coloured vertices. This is a natural generalisation of *online* algorithms, which must colour vertices as soon as they are presented.

Our main result for bottomless rectangles is a generalisation of the non-existence of a semi-online algorithm from above.

▶ **Theorem 1.1.** *For any numbers k and m, for any semi-online algorithm that k-colours bottomless rectangles from above, below, the left, or the right, there is a family of bottomless rectangles that the algorithm colours such that there will be a m-fold covered point that is covered by at most one colour.*
Moreover, the family of the bottomless rectangles can be such that the boundaries of the rectangles are pairwise disjoint.

These are complemented by positive results, where we show that for each of four "natural" bottomless rectangle configurations there is a direction from which there is an optimal online algorithm. Our proof is much more complicated than the one in [2]; while they use an Erdős-Szekeres type incremental argument [3], we need a certain diagonalisation method. In particular, we reduce the semi-online bottomless rectangle colouring problem to a question about semi-online colourings of arborescences, which is interesting in its own right.

▶ **Theorem 1.2.** *For any numbers k and m, and any semi-online algorithm that k-colours the vertices of an arborescence in a leaf-to-root order, there is an arborescence such that the algorithm produces a directed path on m vertices that contains at most one colour.*

In Section 2 we present our main result, and in Section 3 we present our results on polychromatic colourings of bottomless rectangles. The full proofs and other related results can be found in the full version of our paper available at `https://arxiv.org/abs/1912.05251`.

2 Arborescences

An *arborescence* is a directed tree with a distinguished vertex called a *root* such that all the edges are directed away from the root, i.e., there is exactly one directed path to any vertex from the root. (See Figure 1.) A disjoint union of arborescences is an *arborescence forest*, also called a *branching*. We say that an ordering of the vertices of a branching is *root-to-leaf* if every vertex is preceded by its in-neighbors and succeeded by its out-neighbors; in particular, from every component first the root is presented and last a leaf. The following claim easily generalises the notion of a proper colouring for arborescences.

▶ Claim 2.1. The vertices of any arborescence can be k-coloured by an online algorithm in a root-to-leaf order such that any directed path on k vertices contains all k colours.

We call the reversal of a root-to-leaf ordering a *leaf-to-root* ordering; in particular, from every component first a leaf is presented and last the root.

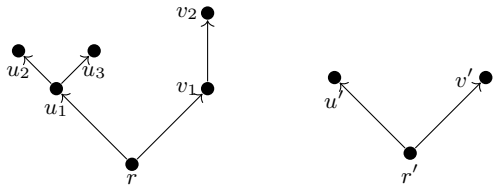


Figure 1 A branching with roots r and r'. A leaf-to-root ordering might present the vertices u', v' and r' before u_3, so it is not necessary that the roots of the branching are the last vertices presented.

Our main result, Theorem 1.2, shows that a similar semi-online polychromatic k-colouring algorithm that takes the vertices in a leaf-to-root order cannot exist, moreover, any such algorithm will even leave an arbitrarily long path monochromatic. To be able to apply this result for bottomless rectangles, we will need (and prove) a slightly stronger notion.

Denote the roots of the branching before a new vertex u is presented by $v_1, v_2, \dots$ indexed in the order in which they were presented. We say that a *leaf-to-root* ordering is *geometric* if the parents of u form an interval in this order, i.e., for every u, $\{v_i \mid u \lessdot v_i\} = \{v_i \mid l < i < r\}$ for some l and r.

▶ **Theorem 2.2.** *There is no semi-online k-colouring algorithm that receives the vertices of an arborescence in a geometric leaf-to-root order, and maintains at any stage (without recolouring any vertices) that all directed paths on m vertices contain at least two colours.*

The key idea of the proof is to exploit that any path of length m must contain at least 2 colours. Further, when a new vertex p is presented, we only need to consider paths of length m rooted at p to check that the colouring is proper. These two conditions show that the algorithm can produce "essentially" only finitely many colourings. More precisely, when p is presented, we "trim" the arborescence of depth m rooted at p to remove any repetitions (see Figure 2 for an example). After this trimming process, we are left with only finitely many different trees. Then we assume that an algorithm has already forced all "achievable" isomorphic trees to appear, and present a new vertex below the root of each. The newly obtained tree must be isomorphic to one of its parents', which leads to a contradiction. For the detailed proof, see the full version of the paper.

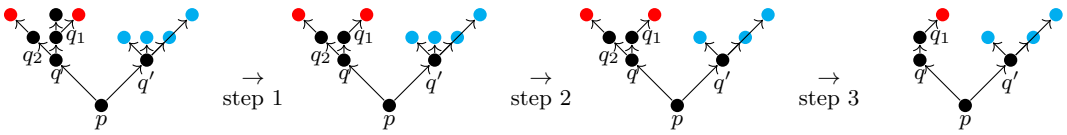


Figure 2 Example for trimming with $m = k = 2$. In step 1, we delete uncoloured points at distance > 2 from p. In step 2, we "trim" the repeated blue parents of q'. In step 3, the subtrees rooted at q_1 and q_2 are isomorphic, so we delete q_2. After this process, we have not lost any information about the paths of length 2 rooted at p.

3 Bottomless rectangles

3.1 Bottomless rectangle configurations

Using the classical result of Erdős and Szekeres [3] that any length $(r-1)(s-1)+1$ sequence of numbers contains either an increasing subsequence of length r or a decreasing subsequence of length s, we will define four configurations of bottomless rectangles as follows.

Ordering the rectangles first by left, then by right endpoints, we obtain that any point contained in $(m-1)^4+1$ rectangles is contained in m bottomless rectangles in an *Erdős Szekeres configuration*. We name these configurations *increasing/decreasing steps, towers* and *nested rectangles*.

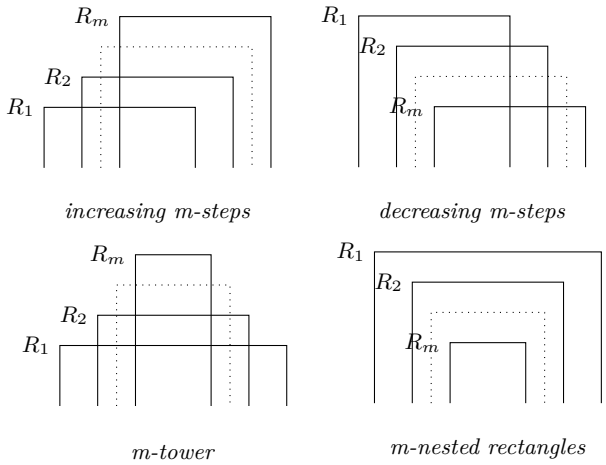


Figure 3 Erdős-Szekeres configurations.

Restricting the colouring problem, we obtain that $m_k = k$ for each fixed configuration. That is, for example, any family of bottomless rectangles can be k-coloured with an online algorithm from the right so that any point covered by increasing k-steps is covered by all k colours. Although these configurations have a seemingly simple structure, we apply the arborescence colouring problem to show that the existence of a semi-online colouring algorithm depends on the order in which the rectangles are coloured.

For a full summary, see the table below.

	left($\rightarrow$)	right($\leftarrow$)	below($\uparrow$)	above($\downarrow$)
inc. steps	∞	$= k$	∞	$= k$
dec. steps	$= k$	∞	∞	$= k$
towers	$= k$	$= k$	$= k$	∞
nested	$= k$	$= k$	∞	$= k$

3.2 Colouring algorithms

▶ **Corollary 3.1.** *There is no semi-online colouring algorithm for towers from above, i.e., for any numbers k and m, for any semi-online algorithm that k-colours bottomless rectangles from above, there is a family of bottomless rectangles such that any two intersecting rectangles form a tower, and the algorithm colours them such that there will be a m-fold covered point that is covered by at most one colour.*

In order to apply Theorem 2.2, we need to show that any branching can be realised as a family of towers so that

1. ordering the rectangles from above corresponds to a geometric leaf-to-root order of the branching, and
2. a semi-online colouring algorithm for towers from above corresponds to an appropriate semi-online k-colouring algorithm for branchings in this order.

We construct this realisation by induction on the number of vertices of the branching. We will need the geometric property of the leaf-to-root ordering to ensure that each isolated vertex is realised as a disjoint rectangle to the right of the former ones. Thus, when a new vertex s is presented, its parents will correspond to some geometrically adjacent rectangles, and s can be realised as a rectangle in the desired configuration.

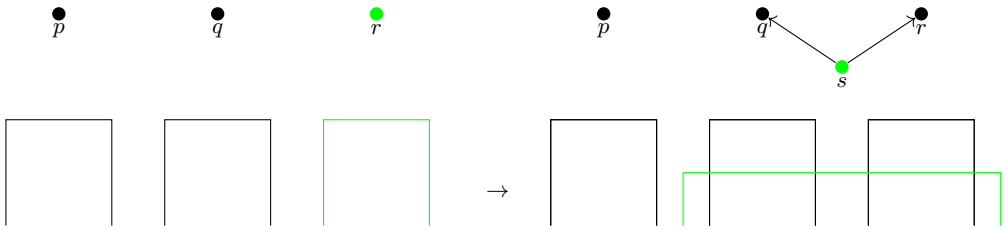


Figure 4 Each time a disjoint element (such as r) is presented, we realise it as a disjoint rectangle to the right. We are then able to realise s as a minimal element that forms a tower with q and r. We need the ordering to be geometric so that the parents of s are consecutive rectangles.

An analagous construction with nested rectangles shows that the same statement holds for nested rectangles from below.

▶ **Corollary 3.2.** *There is no semi-online k-colouring algorithm from the left or from below for increasing steps. More precisely, for any integers k and m, there is no semi-online algorithm to k-colour rectangles from the left (or from below) so that at every step, any point covered by m-increasing steps is covered by at least 2 colours. Similarly, there is no semi-online colouring algorithm for decreasing steps from the right or from below.*

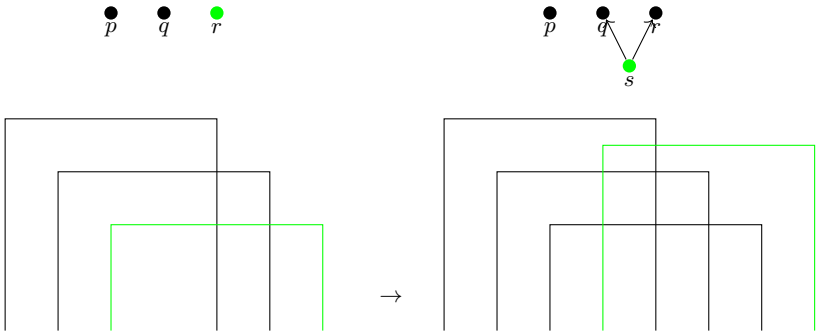


Figure 5 When a disjoint element (r) is presented, we realise it in decreasing steps with the other minimal elements. We are then able to realise s in increasing steps with q and r (once again relying on the geometric ordering).

Note that this statement is slightly weaker than Theorem 1.1 or Corollary 3.1 because we do not exclude the other kind of configurations from the family. Figure 5 shows the modification of this construction for increasing steps from the left.

For online algorithms from other directions, we can prove an optimal upper bound.

▶ **Theorem 3.3.** *Each configuration can be k-coloured so that $m_k = k$.*

These algorithms are simple compared to the proof of non-existence of algorithms from other directions. For example, when colouring with respect to k-towers from below, the

algorithm proceeds as follows. When we present a rectangle R_t, define y_i to be maximal so that any point in R_t below y_i is already covered by colour i (and to be $-\infty$ if this does not exist). Then if $y_1 \geq y_2 \geq \dots y_k$, we colour R_t with colour k. The algorithms for the other configurations are analagous, as depicted in Figures 6 and 7, while the detailed proofs can again be found in the full version.

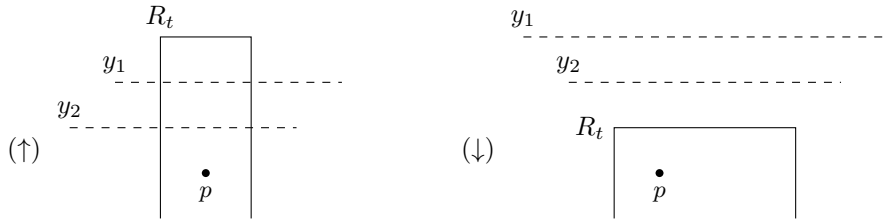


Figure 6 Colouring algorithms for k-towers and k-nested sets, respectively.

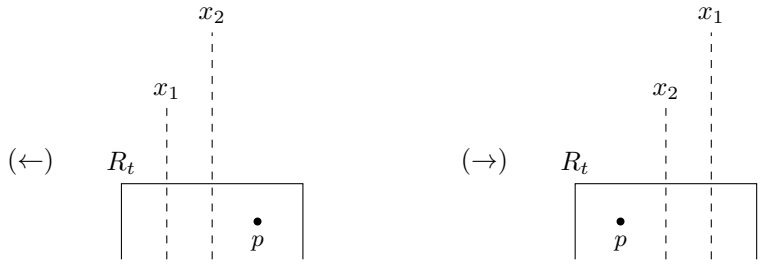


Figure 7 Colouring algorithms for increasing and decreasing k-steps, respectively.

3.3 An improved lower bound

Finally, we prove the following lower bound for general bottomless rectangle families.

▶ **Theorem 3.4.** *$m_k \geq 2k-1$ for bottomless rectangles, i.e., for any k there is a family of bottomless rectangles such that for every k-colouring of the family there is a point of the plane that is contained in at most $k-1$ of the colors, although it is covered by $2k-2$ rectangles.*

Our lower bound construction proceeds in two steps.

1. If $m_k < m_{k-1} + 2$, then every family has a polychromatic k-colouring that is proper (see Figure 8).
2. There is a family so that no polychromatic k-colouring is proper (see Figure 9).

This contradiction shows that $m_k \geq m_{k-1} + 2$, so by induction $m_k \geq 2k-1$. Again, for the details see the full paper.

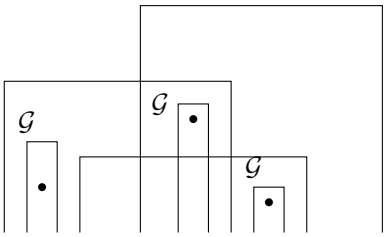


Figure 8 If no polychromatic colouring of $\mathcal{F}$ is proper, place disjoint thin copies of a test family $\mathcal{G}$ around every 2-covered point in $\mathcal{F}$ to obtain a contradiction.

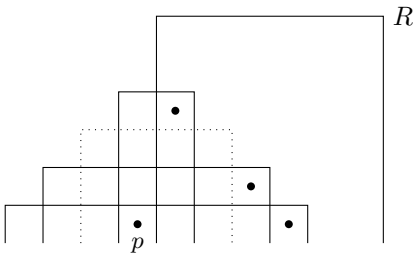


Figure 9 No polychromatic colouring of this family will be proper.

3.4 Concluding remarks

To summarise, our main result shows that by considering arborescences instead of hypergraphs associated to bottomless rectangles, there is no semi-online algorithm to properly colour bottomless rectangles from any of the four "natural" directions. In fact, with a slight modification, we can show that there is no semi-online algorithm to properly colour bottomless rectangles from any other direction either (i.e. along a line).

However, since online algorithms show that $m_k = k$ for each fixed configuration, the next natural step is to attempt to combine these colourings for general families. The strongest such result we have been able to prove is that if a family of bottomless rectangles contains no towers, then it can be k-coloured so that $m_k = O(k^2)$. The crux of the proof is to exploit that (1) adding nested sets to a family can increase m_k by a factor of at most k, and (2) colouring steps with respect to points turns out to be a special case of the primal problem - colouring points with respect to bottomless rectangles. For more details on these and other results, we again refer to the full version of the paper on arXiv.

References

1 A. Asinowski, J. Cardinal, N. Cohen, S. Collette, T. Hackl, M. Hoffmann, K. Knauer, S. Langerman, M. Lason, P. Micek, G. Rote, T. Ueckerdt. "Coloring Hypergraphs Induced by Dynamic Point Sets and Bottomless Rectangles". A*lgorithms and Data Structures*, Lecture Notes in Computer Science Volume 8037. 2013. pp. 73-84.

2 J. Cardinal, K. Knauer, P. Micek, Torsten Ueckerdt. "Making octants colorful and related covering decomposition problems". *SIAM J. on Discrete Math.* 2014. pp. 1948-1959.

3 P. Erdős, G. Szekeres. "A combinatorial problem in geometry". *Compositio Math* 2. 1935. pp. 463-470.

4 B. Keszegh, D. Pálvölgyi. "More on decomposing coverings by octants". *J. of Computational Geometry*, 6(1). 2015. pp. 300-315.

5 J. Pach: "Decomposition of multiple packing and covering" *Diskrete Geometrie, 2. Kolloq.* Math. Inst. Univ. Salzburg. 1980. pp. 169-178.

6 J. Pach. "Covering the plane with convex polygons". *Discrete & Computational Geometry* **1**. 1986. pp. 73-81.

7 J. Pach, D. Pálvölgyi, G. Tóth. "Survey on the Decomposition of Multiple Coverings". *Geometry, Intuitive, Discrete, and Convex* (I. Bárány, K. J. Böröczky, G. Fejes Tóth, J. Pach eds.), Bolyai Society Mathematical Studies, Vol. 24. Springer-Verlag. 2014. pp. 219-257.

On Minimal-Perimeter Lattice Animals*

Gill Barequet[1] and Gil Ben-Shachar[1]

1 **Dept. of Computer Science**
The Technion—Israel Inst. of Technology
Haifa 3200003, Israel
{barequet,gilbe}@cs.technion.ac.il

Abstract

A *lattice animal* is a connected set of cells on a lattice. The *perimeter* of a lattice animal A consists of all the cells that do not belong to A, but that have a least one neighboring cell of A. We consider *minimal-perimeter* lattice animals, that is, animals whose periemeter is minimal for all animals of the same area, and provide a set of conditions that are sufficient for a lattice to have the property that inflating all minimal-perimeter animals of a certain size yields (without repetitions) all minimal-perimeter animals of a new, larger size. We demonstrate this result for polyhexes (animals on the two-dimensional hexagonal lattice). In addition, we provide two efficient algorithms for counting minimal-perimeter polyhexes.

1 Introduction

An *animal* on a d-dimensional lattice is a connected set of lattice cells, where connectivity is through $(d-1)$-dimensional faces of the cells. Specifically, in two dimensions, connectivity is through lattice edges. Two animals are considered identical if one can be obtained from the other by *translation* only, without rotations or flipping.

Lattice animals attracted interest as combinatorial objects [4] and as key players in a model in statistical physics and chemistry [9]. In this paper, we consider lattices in two dimensions, specifically, the hexagonal, triangular, and square lattices, where animals are called polyhexes, polyiamonds, and polyominoes, respectively.

Let $A^{\mathcal{L}}(n)$ denote the number of animals of size n, that is, animals composed of n cells, on the lattice $\mathcal{L}$. A major research problem in the study of lattices is understanding the nature of $A^{\mathcal{L}}(n)$, either by finding a formula for it as a function of n, or by evaluating it for specific values of n. This problem is to this date still open for any nontrivial lattice $\mathcal{L}$. Redelmeier [7] introduced the first algorithm that generates (and counts) all polyominoes of a given size, with no polyomino being generated more than once. The first algorithm for counting lattice animals without generating all of them was introduced by Jensen [6]. Using his method, the number of animals on the 2-dimensional square, hexagonal, and triangular lattices were computed up to size 56, 46, and 75, respectively.

An important property of lattice animals is the size of their *perimeter* (sometimes called "site perimeter"). The perimeter of a lattice animal is defined as the set of empty cells adjacent to the cells of the animal.

In this paper, we consider *minimal-perimeter* animals, that is, animals with the minimal perimeter within all animals of the same size. Altshular et al. [1] and Sieben [8] characterized all polyominoes with maximum size for their perimeter, and provided a formula for the minimum perimeter of a polyomino of size n. Similar results for polyiamonds and polyhexes were given by Fülep and Sieben [5] and by Vainsencher and Bruckstein [10], respectively.

* Work on this paper by both authors has been supported in part by ISF Grant 575/15. Work on this paper by the first author has also been supported in part by BSF Grant 2017684.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

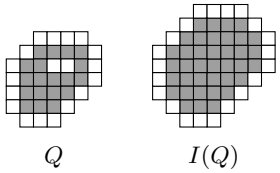


Figure 1 An example of a polyomino Q and its inflated polyomino $I(Q)$. Polyomino cells are colored in gray, perimeter cells are colored in white.

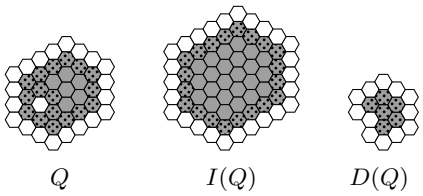


Figure 2 A polyhex Q, its inflated polyhex $I(Q)$, and its deflated polyhex $D(Q)$. The gray cells belong to Q, the white cells are its perimeter, and its border cells are marked with a pattern of dots.

Recently, we studied properties of minimal-perimeter polyominoes [2, 3]. A key notion in our findings was the *inflation* operation. Simply put, inflating a polyomino is adding to it all its perimeter cells (see Figure 1). We showed that inflating all minimal-perimeter polyominoes of some size yields all minimal-perimeter polyominoes of some larger size. In other words, the inflation operation induces a bijection between sets of minimal-perimeter polyominoes. Other results include efficient algorithms for counting minimal-perimeter polyominoes of a given size. In this paper, we provide a nontrivial generalization of this result to any lattice and find a sufficient set of conditions for such a bijection to exist. We also provide efficient counting algorithms for minimal-perimeter polyhexes.

2 Preliminaries

Let $\mathcal{L}$ be a lattice, and Q be an animal on $\mathcal{L}$. The *perimeter* of Q, denoted by $\mathcal{P}(Q)$, is the set of all empty lattice cells that are neighbors of at least one cell of Q. Similarly, the *border* of Q, denoted by $\mathcal{B}(Q)$, is the set of all cells of Q that are neighbors of at least one empty cell.

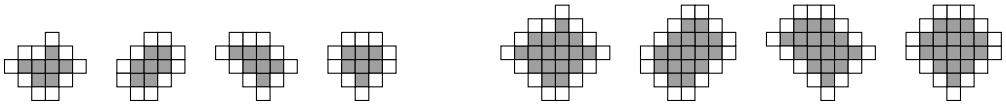
The *inflated* version of Q is defined as $I(Q) := Q \cup \mathcal{P}(Q)$. Similarly, the *deflated* version of Q is defined as $D(Q) := Q \backslash \mathcal{B}(Q)$. These operations are demonstrated in Figure 2.

Denote by $\epsilon^{\mathcal{L}}(n)$ the minimum size (number of cells) of the perimeter of n-cell animals on $\mathcal{L}$, and by $M_n^{\mathcal{L}}$ the set of all minimal-perimeter n-cell animals on $\mathcal{L}$.

Let $\mathcal{S}$ be the two-dimensional square lattice. As mentioned above, animals on $\mathcal{S}$ are usually called *polyominoes*. For this lattice, we know the following.

▶ **Theorem 2.1.** [2, Thm. 4] $\left|M_n^{\mathcal{S}}\right| = \left|M_{n+\epsilon^{\mathcal{S}}(n)}^{\mathcal{S}}\right|$ *(for $n \geq 3$).*

This theorem is a corollary of another theorem that states that the inflation operation induces bijections between sets of minimal-perimeter polyominoes. This is demonstrated in Figure 3. The proof of Theorem 2.1 is based directly on properties of the square lattice. In this paper, we present a nontrivial generalization of this theorem to animals on any lattice which fulfils a set of conditions. This result is stated in Theorem 3.1.



(a) All minimal-perimeter polyominoes of size 7 (up to rotations)

(b) All minimal-perimeter polyominoes of size 17 (up to rotations)

Figure 3 A demonstration of Theorem 2.1.

3 Inflation of Minimal-Perimeter Animals

Our main result consists of a set of conditions, which is sufficient for minimal-perimeter animals to satisfy a claim similar to the one stated in Theorem 2.1. Throughout this section, we consider animals on some specific lattice $\mathcal{L}$.

3.1 A Bijection

▶ **Theorem 3.1.** *Consider the following set of conditions for some lattice $\mathcal{L}$.*

(1) The function $\epsilon^{\mathcal{L}}(n)$ is weakly monotone increasing.
(2) There exists some constant $c \geq 0$, for which, for any minimal-perimeter animal Q on $\mathcal{L}$, we have that $|\mathcal{P}(Q)| = |\mathcal{B}(Q)| + c$ and $|\mathcal{P}(I(Q))| \leq |\mathcal{P}(Q)| + c$.
(3) If Q is a minimal-perimeter animal, then $D(Q)$ is a valid (connected) animal.

If all the above conditions hold for $\mathcal{L}$, then $\left|M_n^{\mathcal{L}}\right| = \left|M_{n+\epsilon^{\mathcal{L}}(n)}^{\mathcal{L}}\right|$. If these conditions are not satisfied for only a finite amount of sizes of animals on $\mathcal{L}$, then the claim holds for all sizes greater than some nominal size n_0. □

Remark. Obviously, no lattice fulfills condition (2) with $c < 0$, and only trivial lattices (e.g., the 1-dimensional lattice) fulfill it with $c = 0$.

The full proof is omitted due to lack of space. However, we detail here the main lemmata which are part of the proof, and which are interesting on their own right.

First, inflating a minimal-perimeter animal preserves this property of the animal.

▶ **Lemma 3.2.** *If Q is a minimal-perimeter animal, then $I(Q)$ is a minimal-perimeter animal as well.* □

Next, under the inflation operation, no two different minimal-perimeter animals are mapped into the same animal.

▶ **Lemma 3.3.** *Let Q_1, Q_2 be two different minimal-perimeter animals. Then, regardless of whether or not Q_1, Q_2 have the same size, the animals $I(Q_1)$ and $I(Q_2)$ are different as well.*

Finally, for any minimal-perimeter animal of size $n + \epsilon^{\mathcal{L}}(n)$ (that is, the size obtained by inflating a minimal-perimeter animal of size n), there exists only a single source animal of size n under the inflation map. Specifically, this source is the animal that is obtained by deflating the original animal.

▶ **Lemma 3.4.** *For any $Q \in M_{n+\epsilon^{\mathcal{L}}(n)}^{\mathcal{L}}$, we also have that $I(D(Q)) = Q$.*

Using the above lemmata, we can wrap up the proof of Theorem 3.1. In Lemma 3.2, we have shown that for any minimal-perimeter animal $Q \in M_n$, we have that $I(Q) \in M_{n+\epsilon^{\mathcal{L}}(n)}$. In addition, Lemma 3.3 states that the inflation of two different minimal-perimeter animals results in two other different minimal-perimeter animals. Combining the two lemmata, we

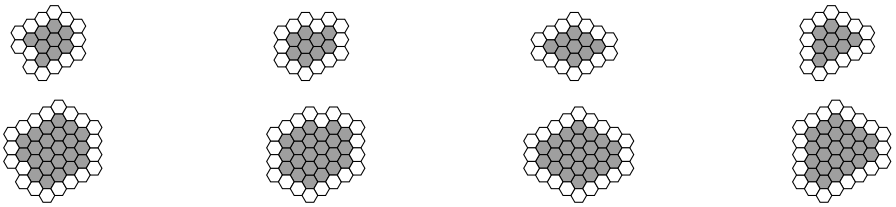


Figure 4 A demonstration of Theorem 3.1 for polyhexes. The top row contains all polyhexes in $M_9^{\mathcal{H}}$ (minimal-perimeter polyhexes of size 9) up to rotations, while the bottom row contains their inflated versions, all members (up to rotations as well) of $M_{23}^{\mathcal{H}}$.

obtain that $|M_n| \leq \left|M_{n+\epsilon^{\mathcal{L}}(n)}\right|$. On the other hand, in Lemma 3.4, we have shown that if $Q \in M_{n+\epsilon^{\mathcal{L}}(n)}$, then $I(D(Q)) = Q$, and, thus, for any animal in $M_{n+\epsilon^{\mathcal{L}}(n)}$, there is a unique source in M_n (specifically, $D(Q)$), whose inflation yields Q. Hence, $|M_n| \geq \left|M_{n+\epsilon^{\mathcal{L}}(n)}\right|$. Combining the two relations, we conclude that $|M_n| = \left|M_{n+\epsilon^{\mathcal{L}}(n)}\right|$.

We can also show that all the premises of Theorem 3.1 are fulfilled for animals on the hexagonal lattice, and, thus, inflating the set of all minimal-perimeter polyhexes of a certain size yields another set of minimal-perimeter polyhexes of another, larger, size. The proofs are omitted due to lack of space. This result is demonstrated in Figure 4.

For polyiamonds, the second condition in Theorem 3.1 does not hold, and indeed, inflating a minimal-perimeter polyiamond does not necessarily result in another minimal-perimeter polyiamond. However, if we slightly modify the triangular lattice to the one in which two cells which share only a vertex are also considered adjacent, then the theorem holds. This is not surprising since under the modified definition, the lattice is homomorphic to the hexagonal lattice. For cubical lattices in three or more dimensions, we know that the second condition does not hold. However, we do not know whether or not the bijection between sets of minimal-perimeter animals on these lattices exists.

3.2 Inflation Chains

Theorem 3.1 implies that there exist infinitely-many chains of sets of minimal-perimeter animals, each one obtained from the previous one by inflation, while the cardinalities of all sets in a single chain are identical. Obviously, there are sets of minimal-perimeter animals that are not created by inflating any other set. We call the size of animals in such sets an *inflation-chain root*. Using the definitions and proofs in the previous section, we are able to characterize which sizes are these roots. The result is stated in the following theorem, while the proof is omitted due to lack of space.

▶ **Theorem 3.5.** *Let $\mathcal{L}$ be a lattice for which the three premises of Theorem 3.1 are satisfied, and, in addition, the following condition holds.*

(4) The inflation operation preserves for an animal the property of having a maximum size for a given perimeter.

Then, if n is the minimum size for a minimal-perimeter size p, or equivalently, if there exists a perimeter size p, such that $n = \min\left\{n \in \mathbb{N} \mid \epsilon^{\mathcal{L}}(n) = p\right\}$, then n is an inflation-chain root. □

As mentioned above, we already provided a similar result for the restricted context of polyominoes [3]. We were not able to identify any lattice for which Theorem 3.1 holds while Theorem 3.5 does not hold, therefore, we suspect that the latter theorem can be inferred from the conditions of the former.

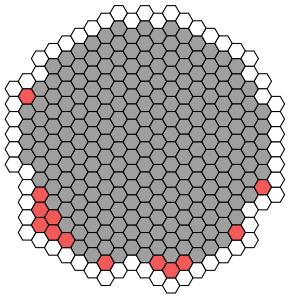


Figure 5 An example of the split of a minimal-perimeter polyhex into its body (grey) and caps (red). Note that the body is composed of 12 "lines" of cells, half of which are aligned with the main three directions (*e.g.*, the left and right edges), and the other half are toggling between two directions, thereby creating three other "lines" (*e.g.*, the top and bottom edges).

4 Counting Minimal-Perimeter Polyhexes

Following Theorem 3.5, one may ask how many minimal-perimeter animals exist for a given inflation-chain root. We describe here two efficient algorithms for counting minimal-perimeter polyhexes of a given size. In fact, it is sufficient to apply these algorithms only to the root of an inflation-chain in order to compute the number of minimal-perimeter polyhexes of the entire chain.

4.1 Bulk Counting

An interesting property of minimal-perimeter polyhexes is the following. Any minimal-perimeter polyhex can be separated into two parts: (1) A dodecagon-like polyhex, with the same perimeter as the original minimal-perimeter polyhex; and (2) Some other cells on the edges of the dodecagon, which do not change the perimeter of the dodecagon. We call these two parts the "body" and the "caps" of the polyhex. This concept is demonstrated in Figure 5.

We can use this property for designing an efficient algorithm for counting minimal-perimeter polyhexes. The algorithm considers all possible bodies with a given perimeter, and calculates the number of different possible caps which yield minimal-perimeter polyhexes. The number of caps fitting each edge is computed by using a rather simple recursive formula, which is quite similar to the formula for Motzkin paths. The time complexity of this algorithm, $O(n^6)$, is pseudo-polynomial. (That is, it is polynomial in n, which is the only input to the algorithm.)

4.2 Column Counting

A polyhex is *convex* if any sequence of its cells along any of the three main directions of the hexagonal lattice is contiguous. A sample convex polyhex is shown in Figure 6.

A better algorithm is based on the following lemma.

▶ **Lemma 4.1.** *Minimal-perimeter polyhexes are convex.* □

Lemma 4.1 implies a simple algorithm for counting minimal-perimeter polyhexes. Simply put, we add cells to the polyhex, one column at a time, in a manner that preserves convexity along the vertical direction, while keeping track of the current area and perimeter. This process is demonstrated in Figure 7.

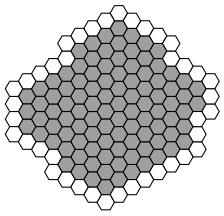


Figure 6 A sample convex polyhex.

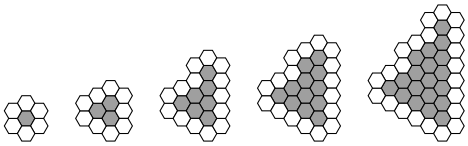


Figure 7 A demonstration of the column-counting algorithm. Each polyhex is generated by one iteration of the algorithm.

Note that in any stage of the algorithm, the possible completions of the animal depend only on the size and perimeter of the animal in the current state, and on the contents of the last two added columns. This property is exemplified in Figure 8. Using this property, we can memoize the results of the computations for each state and reuse them whenever we encounter again the same state. Using this method, we achieve, again, a pseudo-polynomial algorithm with a better running time of $O(n^4)$. The full complexity analysis of the algorithm is omitted due to lack of space.

4.3 Results

Running the column-counting algorithm overnight on a home laptop produced the results shown in Figure 9. One can clearly notice the patterns in the graph, which are very similar to those observed for polyominoes [3]. A natural question which arises from this graph is whether there exists a growth constant for the number of minimal-perimeter polyhexes (when we consider only the roots of the inflation chain). This question is relevant as well for polyominoes, and is still open.

References

1. Y. Altshuler, V. Yanovsky, D. Vainsencher, I.A. Wagner, and A.M. Bruckstein. On minimal perimeter polyminoes. In *Discrete Geometry for Computer Imagery, 13th International Conference, Szeged, Hungary*, pages 17–28. Springer, October 2006.
2. G. Barequet and G. Ben-Shachar. Properties of minimal-perimeter polyominoes. In *Int. Computing and Combinatorics Conference*, pages 120–129, Qingdao, China, July 2018. Springer.
3. G. Barequet and G. Ben-Shachar. Minimal-perimeter polyominoes: Chains, roots, and algorithms. In *Conf. on Algorithms and Discrete Applied Mathematics*, pages 109–123, Kharagpur, India, 2019. Springer.
4. M. Eden. A two-dimensional growth process. *Dynamics of Fractal Surfaces*, 4:223–239, 1961.
5. G. Fülep and N. Sieben. Polyiamonds and polyhexes with minimum site-perimeter and achievement games. *The Electronic J. of Combinatorics*, 17(1):65, 2010.

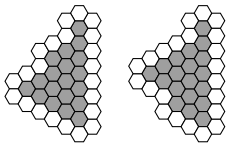


Figure 8 Two possible partial states of the algorithm (created from left to right). Since the size and perimeter, as well as the last two columns of both examples are identical, the possible completions of the polyhex into a minimal-perimeter polyhex are the same.

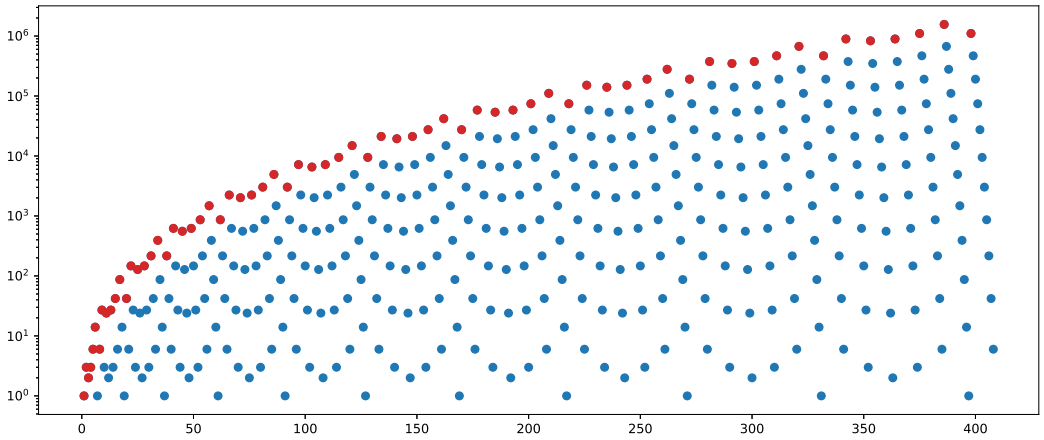


Figure 9 The number of minimal-perimeter polyhexes as a function of the size. Inflation chain roots are colored red.

6 I. Jensen and A.J. Guttmann. Statistics of lattice animals (polyominoes) and polygons. *J. of Physics A: Mathematical and General*, 33(29):L257, 2000.

7 D.H. Redelmeier. Counting polyominoes: Yet another attack. *Discrete Mathematics*, 36(2):191–203, 1981.

8 N. Sieben. Polyominoes with minimum site-perimeter and full set achievement games. *European J. of Combinatorics*, 29(1):108–117, 2008.

9 H.N.V. Temperley. Combinatorial problems suggested by the statistical mechanics of domains and of rubber-like molecules. *Physical Review*, 103(1):1, 1956.

10 D. Vainsencher and A.M. Bruckstein. On isoperimetrically optimal polyforms. *Theoretical Compututer Science*, 406(1-2):146–159, 2008. URL: `https://doi.org/10.1016/j.tcs.2008.06.043`, `doi:10.1016/j.tcs.2008.06.043`.

Shape Formation in a Three-dimensional Model for Hybrid Programmable Matter*

Kristian Hinnenthal[1], Dorian Rudolph[2], and Christian Scheideler[3]

1 **Paderborn University**
`krijan@mail.upb.de`
2 **Paderborn University**
`dorian@mail.upb.de`
3 **Paderborn University**
`scheideler@upb.de`

Abstract

We present the first three-dimensional model for *hybrid programmable matter*, in which small *active* robots with the computational capabilities of finite automata and very limited sensing operate on a structure of *passive* tiles. The model is an extension of the two-dimensional model of Gmyr et al. [DNA 2018], whose hexagonal tiles on the triangular lattice translate to hollow rhombic dodecahedral tiles on the face-centered cubic lattice. Following the idea of Gmyr et al., we initiate the research in this model by considering a *single* robot that can navigate through the structure, pick up tiles, and place them somewhere else to transform the structure. Contrary to the two-dimensional case, where the robot can always find a tile to move while preserving connectivity, we show that such a tile cannot be found in the three-dimensional model without an additional helper tile. We then show how the robot can construct a line with the help of such a tile in $O(n^3)$ rounds, where n is the number of tiles. A line lends itself as an intermediate structure to build more complex objects such as triangles or cubes. Our presentation is accompanied by an implementation in a 3D simulator we specifically developed for our hybrid model.

1 Introduction

Programmable matter is envisioned as a system of small particles that act in coordination to mimic a *programmable physical material* [15]. A variety of models and algorithms for programmable matter has been proposed within the last years, typically focussing on active agents with very limited computational, sensing, and movement capabilities. Notable examples are the *nubot model* [17], *metamorphic robots* [4, 16], and the *amoebot model* [7]. For the amoebot model, problems such as leader election, shape formation, coating, and shape sealing have been investigated (see, e.g., [3, 5, 6, 8]). Very recently, a hybrid variant of the amoebot model has been proposed, in which *active* agents similar to amoebots move on a structure of *passive* hexagonal tiles on the triangular lattice [7, 9, 10]. Contrary to programmable matter only comprised of active elements, here the goal is to have only few active elements that act on a potentially large structure of passive elements. As a potential application, such systems may be used to construct or repair structures in complex or hardly accessible environments such as the human body, narrow pipe systems, or space [11].

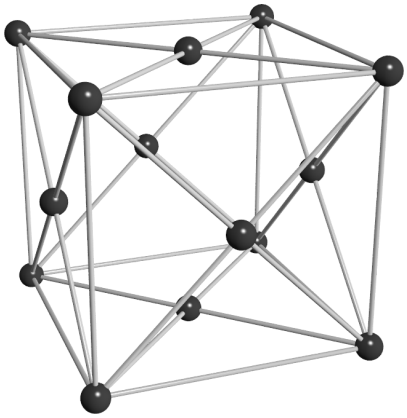
Our Contribution. We present a *three-dimensional* model for hybrid programmable matter. Whereas the three-dimensional case has been intensively studied in related areas such as modular self-reconfigurable robot systems (see, e.g., [13], or [1, 14] and the references

* Supported by DFG Project SCHE 1592/6-1 (Algorithms for Programmable Matter in a Physiological Medium).

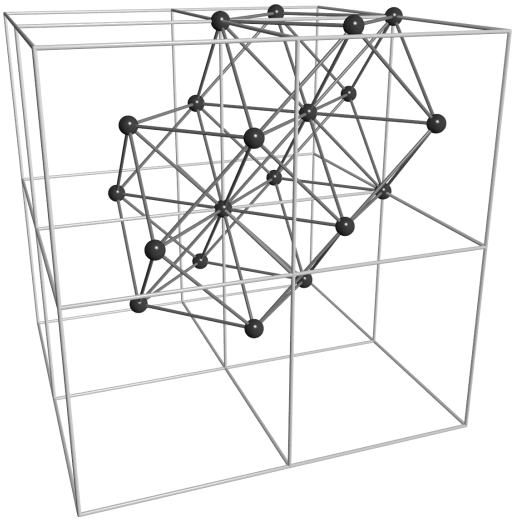
36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

therein) and molecular self-assembly (see, e.g., [12, 2]), to the best of our knowledge, there does not exist a model for our vision of programmable matter. Our model naturally extends the two-dimensional model for hybrid programmable matter presented by Gmyr et al. [10] to three dimensions: instead of robots moving on hexagonal tiles on the triangular lattice, the robots move through (hollow) rhombic dodecahedral tiles on the face-centered cubic (FCC) lattice, i.e., the adjacency graph of the FCC sphere-packing (see Fig. 1).

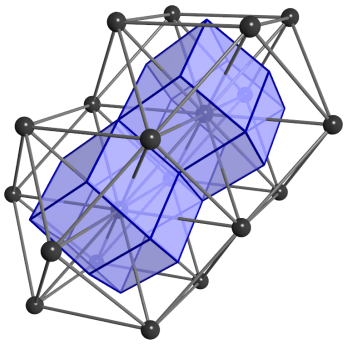
Note that the space-filling tesselation of the rhombic dodecahedron is precisely the Voronoi tesselation of the FCC lattice. Compared to cubic structures as an apparent alternative space-filling tiling, the rhombic dodecahedral structures are stronger connected and therefore more sturdy, as every node has 12 neighbors and each pair of adjacent cells shares a common face. Furthermore, this allows robots to move from a tile to each adjacent tile through the incident face and without having to leave the tile structure. Note that whereas in the two-dimensional model tiles can be moved by carrying them *over* the tile structure, we require the robot to carry tiles *through* existing tiles. Therefore, in a practical setting, the robot needs to be able to fold and unfold carried tiles to squeeze them through other tiles.



(a) Subgraph of the FCC lattice that corresponds to a single cube.



(b) Larger subgraph of the FCC lattice (contained within 8 cubes).



(c) Two voronoi cells (blue rhombic dodecahedra) of the subgraph of (b).

Figure 1 Rhombic dodecahedral tiles in the FCC lattice.

We follow the idea of Gmyr et al. and initiate the research in this model by considering the problem of forming a line with a *single* robot. Specifically, we present an algorithm that transforms any initially connected structure into a line in $O(n^3)$ rounds while preserving connectivity at all times. Similar to the two-dimensional case, a line can be used as an intermediate structure to build a triangle, and simple three-dimensional structures such as pyramids and cubes can be constructed in a very similar fashion; due to space constraints, we leave out the exact description of these algorithms. As our results indicate, moving tiles in three dimensions without violating connectivity requires much more sophisticated behavior. Specifically, we show that the robot cannot locally decide which tile can be moved *at all* without violating connectivity, which we address by providing the robot with an additional helper tile. Our algorithm has been implemented in a three-dimensional hybrid programmable matter simulator that we developed for visualization and evaluation purposes. A few screenshots of the formation of a line, as well as the transformation of the line into a pyramid, can be found in Fig. 2.

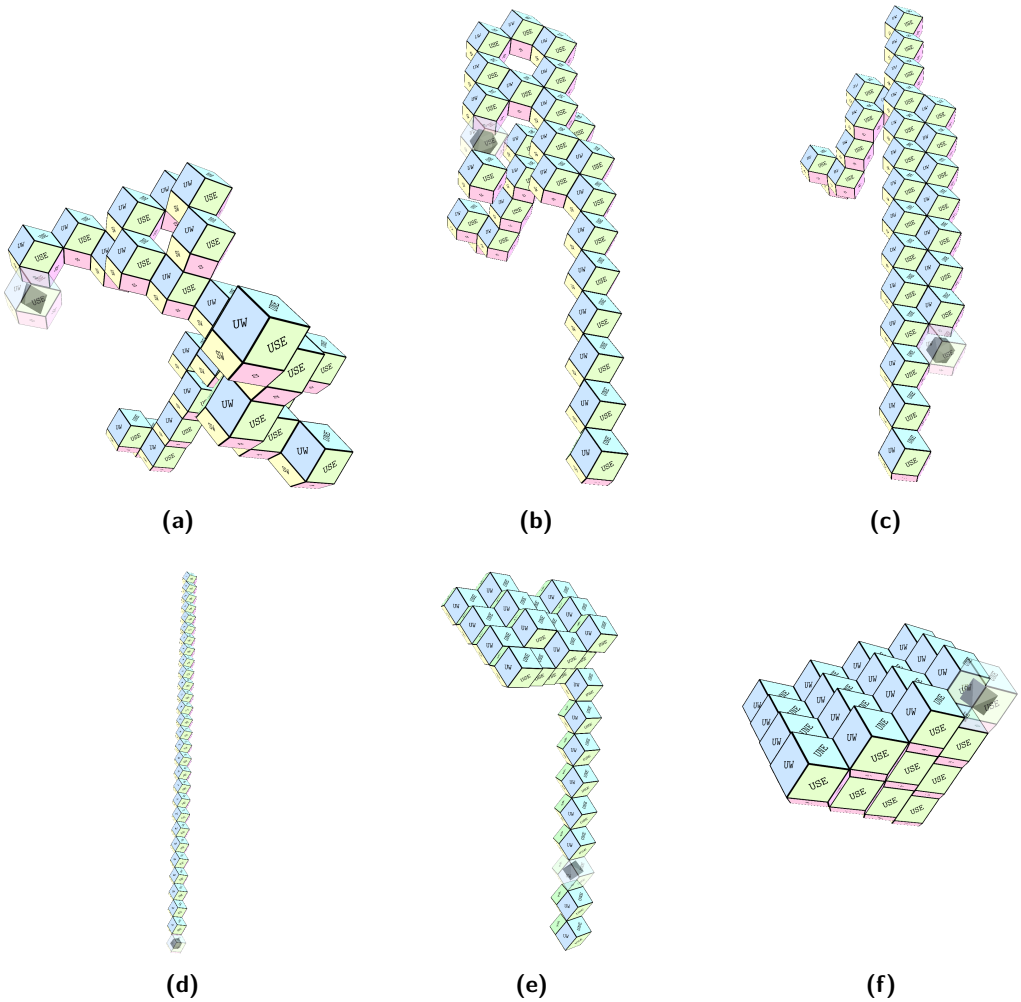


Figure 2 Simulator screenshots of constructing a line (d) and transforming it into a pyramid (f).

Model. As in the two-dimensional model [10], we assume that a single robot r operates on a finite set of n tiles. In our model, each tile forms a hollow rhombic dodecahedron and occupies exactly one node of the infinite FCC lattice $G = (V, E)$. For ease of presentation, we regard G as a stack of hexagonal layers, where each layer can be represented in the triangular lattice (see gray scale layers in Fig. 3).

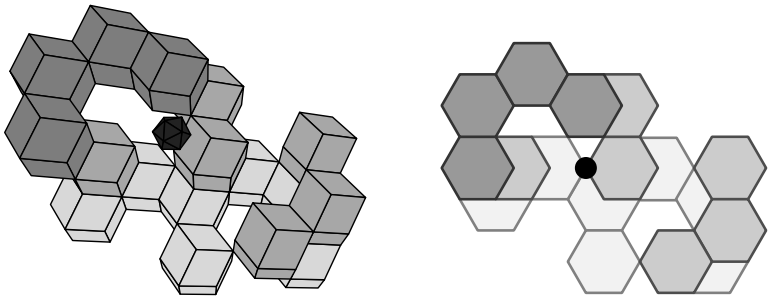


Figure 3 Example configuration with three layers depicted in 3D (left) and as a stack of hexagonal structures in the corresponding triangular lattices (right). The robot is shown as a black circle.

A *configuration* (T, p) consists of a set $T \subseteq V$ of all nodes occupied by tiles, and the robot's position $p \in V$. We assume that the initial position of the robot is occupied by a tile and that the robot initially *carries* one of the n tiles. Every node $u \in V$ is adjacent to twelve neighbors, and, as indicated in Fig. 4, we describe the relative positions of adjacent nodes by the cardinal directions N, NE, SE, S, SW, NW, as well as the "up-directions" USE, UNE, UW, and the "down-directions" DE, DNW, and DSW. The robot must always be on or adjacent to a node occupied by a tile. Additionally, if the robot does not carry a tile, we require the subgraph of G induced by T to be connected; otherwise, the subgraph induced by $T \cup \{p\}$ must be connected. This restriction prevents the configuration from falling apart in a practical scenario.

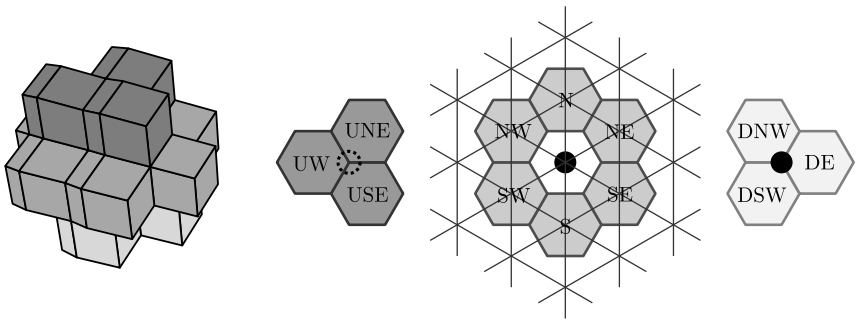


Figure 4 Tiles are labeled with the direction from the robot's point of view. The three layers are depicted separately. The center one also shows the grid.

The robot has only constant memory, which gives it the computational capabilities of a finite automaton, and operates in rounds of *Look-Compute-Move* cycles. In the *Look* phase of a round, the robot can observe its node p and the twelve neighbors of that node. For each of these nodes, it can determine whether the node is occupied or not. In the *Compute* phase, the robot may change its internal state (or terminate) and determines its next move according to the observed information. In the *Move* phase, the robot can either (1) pick up a tile from p, if $p \in T$, (2) place a tile it is carrying at p if $p \notin T$, or (3) move to an adjacent

node while possibly carrying a tile with it. The robot can carry at most one tile.

2 Safely Movable Tiles

As shown by Gmyr et al. for the two-dimensional model [10, Theorem 1], a single robot cannot find a *safely removable tile*, i.e., a tile on a node v such that the subgraph induced by $T \setminus \{v\}$ is connected. Gmyr et al. circumvent this issue by repeatedly locating and moving a *safely movable tile* to transform the structure, i.e., a tile at some node v for which there exists an adjacent node v' such that the subgraph induced by $(T \setminus \{v\}) \cup \{v'\}$ is connected. Specifically, the robot moves tiles to adjacent nodes without even violating connectivity in the tile's *local* neighborhood, which obviously also preserves global connectivity. Formally, we call a tile at node v *locally safely movable* if it has an adjacent node v' such that the graph induced by $N(v) \cup \{v'\}$ is connected, where $N(v) \subseteq T$ is the set of occupied nodes adjacent to v.

As an example, in two dimensions, a hollow hexagon of tiles can be transformed into a triangle by repeatedly moving locally safely movable tiles (i.e., the hexagon's corners) inwards until there is no hole anymore [10]. At this point, it is very easy to find safely removable tiles and rearrange them into the target structure (e.g., a triangle). Since a safely removable tile cannot be found in general [10, Theorem 1], the strategy of moving locally safely movable tiles to create an intermediate structure proves to be very useful. However, the impossibility result for safely removable tiles does not only translate to the three-dimensional case as well, there even exists a configuration that does not have *any* locally safely movable tile in our model (see Fig. 5), making it impossible to use the approach for two dimensions. Even worse, the following theorem shows that the robot cannot find a safely movable tile in general, even though it might exist.

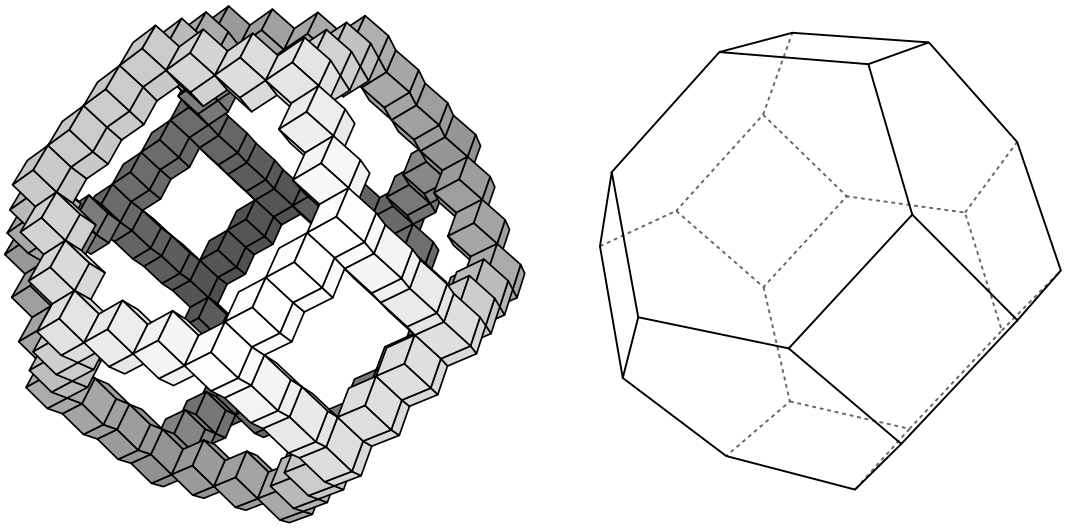


Figure 5 A tile configuration without any *locally* movable tile (i.e., a tile that can be moved without disconnecting its local neighborhood) and the truncated octahedron that it resembles. Note that whereas each tile is in fact safely movable, this cannot be decided locally by the robot.

▶ **Theorem 2.1.** *There does not exist a robot that terminates on a safely movable tile on every initial configuration without ever moving a tile.*

Proof sketch. Construct a tree G (i.e., the graph induced by T is a tree) whose only safely movable tiles are its leaves, and which a robot cannot distinguish from a variation H of Fig. 5 with sufficiently long edges. Roughly speaking, a finite automaton robot cannot "measure" the length of a line, therefore the robot will behave exactly the same on G as it does on H. Since it terminates on H after having traversed a constant number of edges (otherwise it would enter an infinite loop), it must also terminate on G before having reached its leaves (which are the only safely movable tiles in G). ◀

This implies that it is impossible for the robot to select a tile to move in general, which justifies our choice to equip the robot with an initial tile. As we show in the following section, the robot can use the additional tile to transform the structure without violating connectivity.

3 Line Formation Algorithm

In this section, we present our algorithm to transform any initial configuration into a line in $O(n^2 \cdot h) \subseteq O(n^3)$ rounds, where h is the height (i.e., number of layers) of the initial tile configuration. The basic idea is to repeatedly merge *columns*, i.e., maximal lines of occupied nodes in the N/S direction, until only a single column remains. The robot cycles through the following phases, starting with phase *FC*:

- ***FC*** (find column): Find a column without neighbors (i.e., adjacent tiles) above (i.e., USE, UNE or UW) or west (i.e., NW or SW) by traversing columns from N to S and moving up or west whenever possible. Eventually, the robot will find such a column. If that column has a neighbor below (i.e., DE, DNW or DSW), switch to *MCD*; if otherwise it has a neighbor to the east (i.e., NE or SE), switch to *MCE*. Halt once only a single column remains.
- ***MCD*** (move column down): The goal of this phase is to move the column's northernmost remaining tile to the first (from N to S) empty node DE of the column, or, if no such node exists, S of the column. Switch back to *FC* once either all tiles have been moved DE, or the column was merged with another to the south.
- ***MCE*** (move column east): Same as *MCD*, but move tiles SE instead of DE.

We next describe the technically more challenging phases *MCD* and *MCE* in more detail. Our algorithm requires the robot to basically always carry a tile with it, therefore moving a tile t to some node u is actually performed by first placing the carried tile at u and picking up t afterwards (where the movement from u to t is the only time the robot does *not* carry a tile). In the following, let C denote the set of nodes occupied by the column the robot r currently operates on. C_N and C_S denote the column's northernmost and southernmost nodes, respectively. For any node u, denote by, e.g., $u + \mathrm{NW}$ the node NW of u.

***MCD*:** At the beginning of each iteration of this phase the robot's position is $p = C_\mathrm{N} + \mathrm{NE}$ (see Fig. 6a). Recall that before moving the tile t' at C_N, r first has to place its carried tile t at the first empty node DE of C, and then picks up t'. To do so, it follows the path depicted in Fig. 6a until it finds an empty node, and places t. Afterwards, r moves back to C_N and picks up t' (see Fig. 6b–6c). r then distinguishes the following cases:

- If t was placed at $v = C_\mathrm{S} + \mathrm{S}$, and there was a tile at $v + \mathrm{S}$ (see Fig. 6h), r moves to an adjacent tile and switches to *FC* (see Fig. 6i). In this case, r has essentially merged C with the column to its south.
- Else, if r's position is $p = C_N = C_S$, which is the case if $p + \mathrm{S} \notin T$, then r has picked up the column's last tile. The robot then moves to an adjacent tile and switches to *FC*.
- Otherwise, r moves SE and begins the next iteration of phase *MCD* (see Fig. 6c).

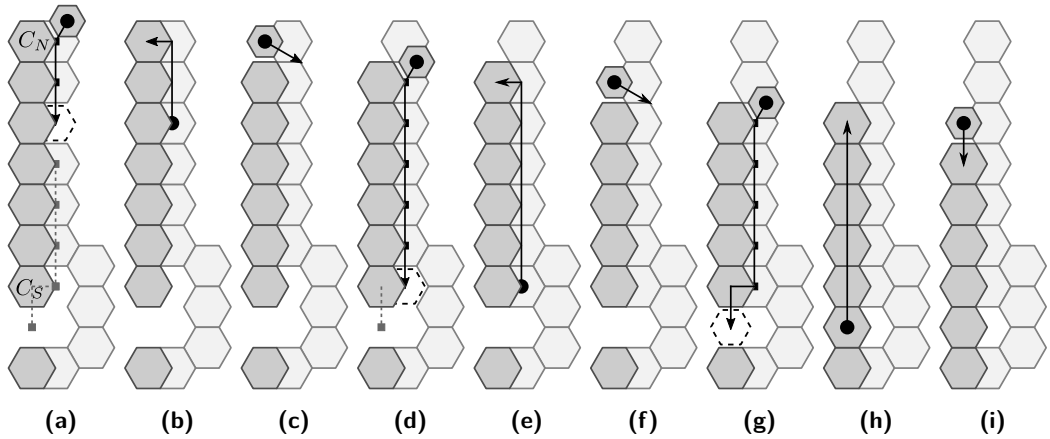


Figure 6 Example of the phase *MCD*. In (a), (d), and (g), r follows the black path until it reaches the first empty node (marked with a dashed outline). The dashed line indicates how the path would proceed if r had not already found an empty node. The carried tile is shown smaller.

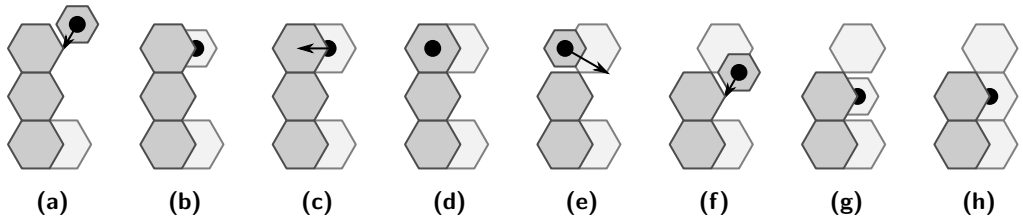


Figure 7 Maintaining connectivity during phase *MCD* using the carried tile.

Fig. 7 illustrates how r makes use of the initially carried tile to maintain connectivity when moving tiles DE (see steps 7e–7g). Note that a new iteration of the phase starts at 7f, where r being at $C_{\mathrm{N}} + \mathrm{NE}$ is essential for the configuration's connectivity.

***MCE*:** The behavior of r in this phase only differs from *MCD* in the path r follows to find the first empty node SE of C (see Fig. 8a). The depicted example shows the case where r moves all tiles SE without merging C with another column to the south (as in Fig. 6).

Using the observation displayed in Fig. 7, it is straightforward to show the next lemma.

▶ **Lemma 3.1.** *The robot does not disconnect the configuration.*

It remains to be proven that r does indeed build a line in $O(n^2 \cdot h)$ rounds.

▶ **Lemma 3.2.** *The robot spends at most $O(n^2)$ rounds in the phases* MCE *and* MCD*.*

Proof sketch. The robot r only moves tiles in the directions S, SE, and DE (assuming the previously discussed interpretation of moving tiles). One can show that if r moves tiles by a total distance of k in phase *MCE* or *MCD*, it spends $O(k)$ rounds in that phase. It can further be shown that the distance between each tile's initial and final positions is bounded by $O(n)$. Hence, tiles are moved a total distance of $O(n^2)$. ◀

▶ **Lemma 3.3.** *The robot spends at most $O(n^2 \cdot h)$ rounds in phase* FC*.*

Proof sketch. r traverses each column at most once during *FC*, since it only exits columns by moving west or up. Hence, it exits *FC* after $O(n)$ rounds. *MCE* always merges C with another column (either S or SE of C), thereby reducing the total number of columns. *MCD*

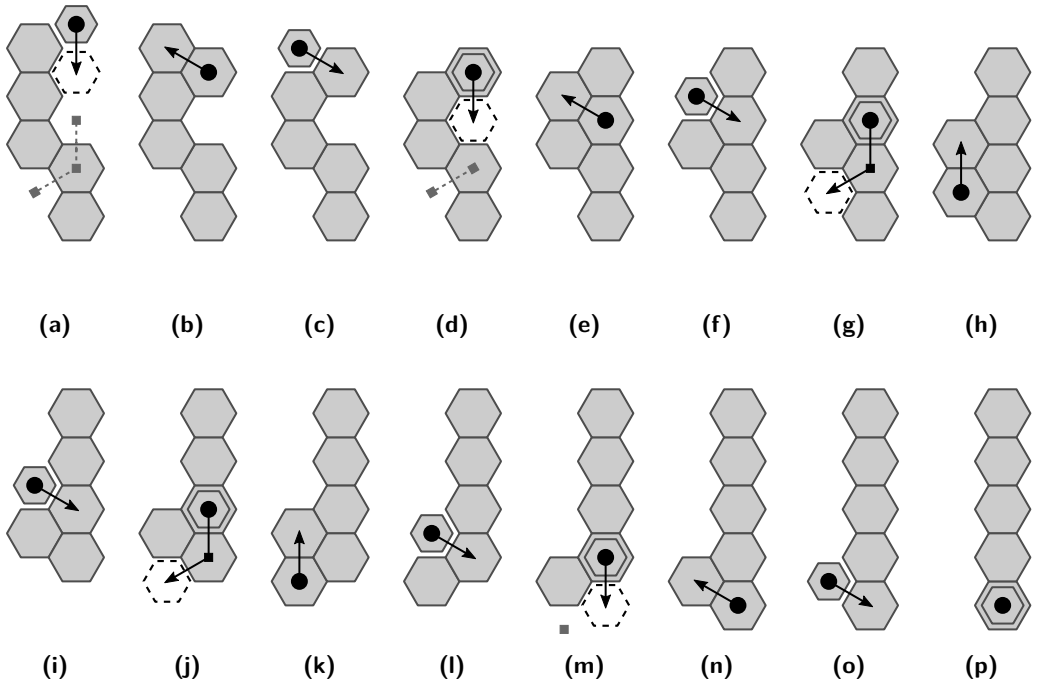


Figure 8 Example of the phase *MCE*.

merges C with another column to its S, or it moves at least one tile down. The former case occurs at most $O(n)$ times since there are at most $O(n)$ columns. The latter occurs at most $O(n \cdot h)$ times, since r only moves a tile down if its column already has a neighbor below. Hence, r enters phases *MCE* and *MCD* (and thus also *FC*) at most $O(n \cdot h)$ times. ◀

Since r only halts once it has built a line, we conclude the following theorem.

▶ **Theorem 3.4.** *The robot builds a line in $O(n^2 \cdot h)$ rounds.*

Acknowledgments. The authors would like to thank Irina Kostitsyna for many helpful discussions from which this work originated.

References

1 Hossein Ahmadzadeh, Ellips Masehian, and Masoud Asadpour. Modular Robotic Systems: Characteristics and Applications. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 81(3-4):317–357, 2016.

2 Faisal A. Aldaye, Alison L. Palmer, and Hanadi F. Sleiman. Assembling materials with dna as the guide. *Science*, 321(5897):1795–1799, 2008.

3 Rida A. Bazzi and Joseph L. Briones. Stationary and Deterministic Leader Election in Self-organizing Particle Systems. In *International Symposium on Stabilizing, Safety, and Security of Distributed Systems*, pages 22–37, 2019.

4 Gregory S. Chirikjian. Kinematics of a metamorphic robotic system. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 449–455, 1994.

5 Joshua J. Daymude, Zahra Derakhshandeh, Robert Gmyr, Alexandra Porter, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. On the runtime of universal coating for programmable matter. *Natural Computing*, 17:81–96, 2018.

6 Joshua J. Daymude, Robert Gmyr, Kristian Hinnenthal, Irina Kostitsyna, Christian Scheideler, and Andréa W. Richa. Convex Hull Formation for Programmable Matter. In *Proc. of the 21st International Conference on Distributed Computing and Networking (ICDCN) (to appear)*, 2020.

7 Joshua J. Daymude, Kristian Hinnenthal, Andréa W. Richa, and Christian Scheideler. Computing by programmable particles. In *Distributed Computing by Mobile Entities: Current Research in Moving and Computing*, pages 615–681. 2019.

8 Zahra Derakhshandeh, Robert Gmyr, Thim Strothmann, Rida Bazzi, Andréa W. Richa, and Christian Scheideler. *Leader Election and Shape Formation with Self-organizing Programmable Matter*, pages 117–132. 2015.

9 Robert Gmyr, Kristian Hinnenthal, Irina Kostitsyna, Fabian Kuhn, Dorian Rudolph, and Christian Scheideler. Shape recognition by a finite automaton robot. In *43rd International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 52:1–52:15, 2018.

10 Robert Gmyr, Kristian Hinnenthal, Irina Kostitsyna, Fabian Kuhn, Dorian Rudolph, Christian Scheideler, and Thim Strothmann. Forming tile shapes with simple robots. *Natural Computing*, 2019.

11 Benjamin Jenett and Daniel Cellucci. A mobile robot for locomotion through a 3d periodic lattice environment. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5474–5479, 2017.

12 Matthew R. Jones, Robert J. Macfarlane, Byeongdu Lee, Jian Zhang, Kaylie L. Young, Andrew J. Senesi, and Chad A. Mirkin. DNA-nanoparticle superlattices formed from anisotropic building blocks. *Nature Materials*, 9(11):913–917, 2010.

13 Ara N. Knaian, Kenneth C. Cheung, Maxim B. Lobovsky, Asa J. Oines, Peter Schmidt-Neilsen, and Neil A. Gershenfeld. The Milli-Motein: A self-folding chain of programmable matter with a one centimeter module pitch. In *IEEE International Conference on Intelligent Robots and Systems*, pages 1447–1453, 2012.

14 Esben Hallundbæk Østergaard, Kristian Kassow, Richard Beck, and Henrik Hautop Lund. Design of the atron lattice-based self-reconfigurable robot. *Autonomous Robots*, 21(2):165–183, 2006.

15 Tommaso Toffoli and Norman Margolus. Programmable matter: Concepts and realization. *Physica D: Nonlinear Phenomena*, 47(1):263–272, 1991.

16 Jennifer E. Walter, Elizabeth M. Tsai, and Nancy M. Amato. Algorithms for fast concurrent reconfiguration of hexagonal metamorphic robots. *IEEE Transactions on Robotics*, 21(4):621–631, 2005.

17 Damien Woods, Ho-Lin Chen, Scott Goodfriend, Nadine Dabby, Erik Winfree, and Peng Yin. Active self-assembly of algorithmic shapes and patterns in polylogarithmic time. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, ITCS '13, pages 353–354, 2013.

Smallest Universal Covers for Families of Triangles

Ji-won Park[*1] **and Otfried Cheong**[2]

1 **Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France**
ji-won.park@inria.fr

2 **School of Computing, KAIST, Daejeon, Korea**
otfried@kaist.airpost.net

Abstract

A universal cover for a family $\mathcal{T}$ of triangles is a convex shape that contains a congruent copy of each triangle $T \in \mathcal{T}$. We conjecture that for any family $\mathcal{T}$ of triangles (of bounded area) there is a *triangle* that forms a universal cover for $\mathcal{T}$ of smallest possible area. We prove this conjecture for all families of two triangles, and for the family of triangles that fit in the unit circle.

1 Introduction

A *universal cover* for a given family $\mathcal{T}$ of objects is a convex set Z that contains a congruent copy of every element $T \in \mathcal{T}$. A *smallest* universal cover is a universal cover of the smallest area (there can be multiple smallest universal covers).

Perhaps the oldest question on universal covers was asked by Lebesgue in 1914: what is the smallest area of a convex set Z that can be used to cover a congruent copy of any set of diameter one in the plane? Lebesgue's problem was first studied by Pál [8], who found that the area of a smallest universal cover is at least 0.8257 and at most 0.8454. Both bounds were improved by several authors, the current best upper bound is around 0.844 [3], the best lower bound is around 0.832 [5], so the problem is still open.

Moser asked for the smallest universal cover for the family of curves of length one. The problem is interesting both for open and closed curves, and both versions are still open. The survey by Wetzel [9] and the book by Brass et al. [4] list these and other results related to universal covers.

Among the few problems on universal covers that are solved are questions where $\mathcal{T}$ is a family of *triangles*. It is known that the smallest universal cover for the family of all triangles of perimeter one, as well as for the family of all triangles of diameter one, is itself a triangle [6, 7]. For the family of all triangles of diameter one, the smallest universal cover similar to a prescribed triangle is also known [10].

We conjecture that this is not a coincidence, and that there is always a triangle that forms a smallest universal cover. More formally, we define a family $\mathcal{T}$ of triangles to be *bounded* if there is a constant D such that no element of $\mathcal{T}$ has diameter larger than D, and state the following conjecture:

▶ Conjecture 1. For any bounded family $\mathcal{T}$ of triangles there is a triangle Z that is a smallest universal cover for $\mathcal{T}$.

If true, this would mirror the situation for translation covers of line segments: the smallest convex translation cover for *any* family of line segments can be chosen to be a triangle [2].

* This research was partially supported by MSIT/NRF (No. 2019R1A2C3002833)

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Our results. Our first result (Theorem 4) describes the triangle T^* that is the unique smallest universal cover for the family $\mathcal{T}_\circ$ of all triangles that fit into the unit circle. This complements the previous results by Kovalev [7] and Füredi and Wetzel [6], as the radius of the circumcircle is, next to diameter and perimeter, a natural "size" for triangles.

It turns out that T^* can be defined by two specific triangles in $\mathcal{T}_\circ$. In other words, T^* is already the smallest universal cover for a *two-element subfamily* of $\mathcal{T}_\circ$. One can notice from the constructions in [6, 7] that the same is true for the family of all triangles of diameter one, and for the family of all triangles of perimeter one. We show that Conjecture 1 holds for any family of triangles with this property, by proving that the smallest universal cover for a family of any *two triangles* can be chosen to be a triangle (Theorem 7).

Hence, if, for an arbitrary triangle family, a smallest universal cover can be determined by a subfamily consisting of two triangles, then Conjecture 1 is implied by Theorem 7. However, we show that not all families of triangles have this property. We give an example of a family $\mathcal{T}_3$ of three triangles such that each proper subfamily has a universal cover smaller than a smallest universal cover of $\mathcal{T}_3$ (Theorem 8).

2 Preliminaries

We will say that a convex shape X *fits into* a convex shape Y if there is a shape $X' \subseteq Y$ congruent to X (that is, X' is the image of X under translation, rotation, and reflection). We say that X *maximally fits into* Y if X fits into Y, but there is no shape X' that is similar to X and larger than X and fits into Y. The following lemma has been well known; see, for instance, Agarwal et al. [1] for a proof.

▶ **Lemma 2.** *If a triangle X maximally fits into a convex polygon Y, then there are at least four incidences between vertices of X and edges of Y.*

Since the triangle X has only three vertices, one of them must be involved in two incidences, that is, it must coincide with a vertex of Y. Of particular interest to us is the special case where Y is a triangle as well. In this case the lemma implies that a vertex of X coincides with a vertex of Y and that an edge of X lies on an edge of Y. There are three cases, depicted in Figure 1. An immediate consequence is the following:

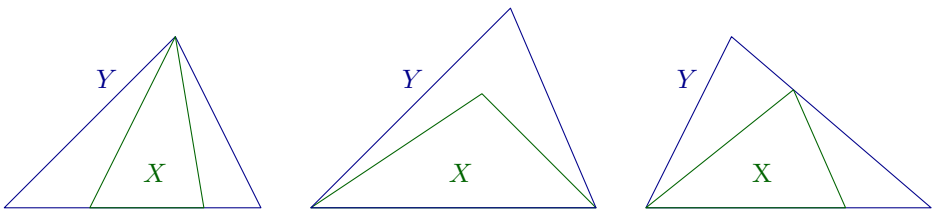


Figure 1 The three cases where X maximally fits into Y.

▶ **Corollary 3.** *If a triangle X fits into a triangle Y, then we can place X in Y such that an edge of X lies on an edge of Y.*

We will let $|PQ|$ denote the length of the segment PQ, while $|X|$ denotes the area of a convex shape X; we also use $|ABC|$ to denote the area of the triangle $\triangle ABC$ and similarly for convex polygons with more than three corners.

3 Triangles contained in the unit circle

Let $T_0 = \triangle ABC$ be the equilateral triangle of side length $\sqrt{3}$. This is the largest equilateral triangle that fits into the unit circle. Then, for $60° \leqslant \theta < 90°$, we let $T_1(\theta) = \triangle DEF$ be the isosceles triangle whose circumradius is one and whose base angles are θ. We place $T_1(\theta)$ such that its long edge DE is aligned with the edge AB of T_0 and its corner F lies on the edge AC; see Figure 2. We define $T_2(\theta)$ as $\triangle ADC$.

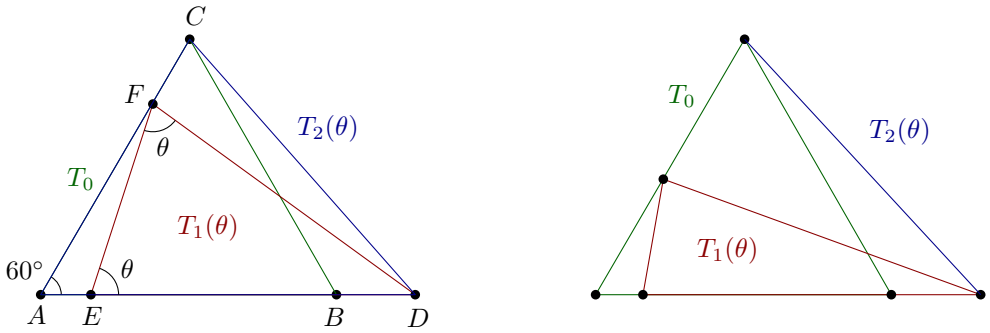


Figure 2 Construction of $T_2(\theta) = \triangle ADC$, for two different values of the angle θ.

We now define $T^* = T_2(80°)$. Our first main result will be the following:

▶ **Theorem 4.** *The triangle T^* is the unique smallest universal cover for the family of all triangles that fit in the unit-radius circle.*

One may wonder what makes 80° special. The reason is that it is for $\theta = 80°$ that $T_1(\theta)$ maximally fits into $T_2(\theta)$ in two distinct ways. To see this, consider the height HD in T^*, and reflect both A and F about the line HD, obtaining points $\tilde{C}$ and G: Calculation shows that $\angle FDC \approx 27.52° > 20°$, resulting in Figure 3. Since $\angle A\tilde{C}D = \angle CAD = 60°$, we obtain an equilateral triangle $\triangle AD\tilde{C}$. We also have $|DG| = |DF|$ and $\angle GDH = \angle FDH = 10°$, so $\triangle FDG$ is congruent to $\triangle EDF = T_1(80°)$.

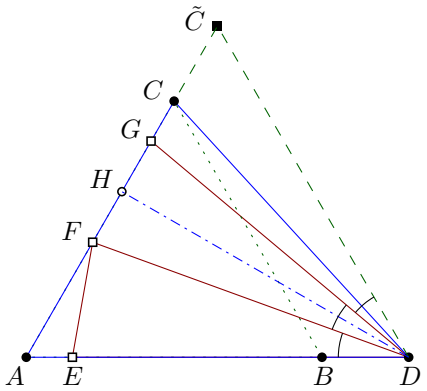


Figure 3 $T^* = \triangle ADC$.

For proving the optimality of T^*, the following lemma is useful, which is an adaptation of a result by Füredi and Wetzel [6, Theorem 5].

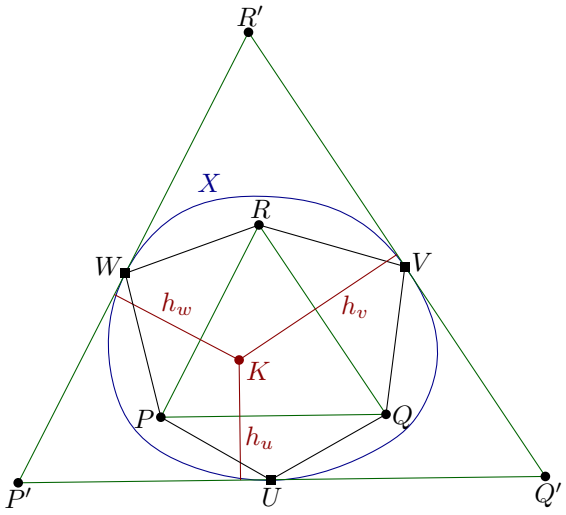


Figure 4 Proof of Lemma 5.

▶ **Lemma 5.** *Let $\mathcal{T}$ be a family of triangles, and let Z be a universal cover for $\mathcal{T}$. Let $S \in \mathcal{T}$, and let S' be the smallest universal cover for $\mathcal{T}$ that is similar to S. If*

$$\frac{|S'|}{|S|} = \Big(\frac{|Z|}{|S|}\Big)^2,$$

then Z is a smallest universal cover for $\mathcal{T}$.

Proof. Let $\triangle PQR = S$, and let X be a universal cover for $\mathcal{T}$. We can assume $S \subseteq X$. We draw tangents to X that are parallel to the edges of S, obtaining a triangle $\triangle P'Q'R'$ that encloses X and that is similar to S; see Figure 4. By the assumption, this implies that $|P'Q'R'| \geqslant |S'|$, and therefore

$$\frac{|P'Q'|}{|PQ|} \geqslant \frac{|Z|}{|S|}.$$

Let U, V, and W be points of X on the three edges of $\triangle P'Q'R'$, let K be any point inside S, and let h_u, h_v, and h_w be the distances from K to the lines $P'Q'$, $Q'R'$, and $R'P'$, respectively. We then have

$$\begin{aligned}
|X| \geqslant |PUQVRW| &= \frac{1}{2}\big(|PQ|h_u + |QR|h_v + |RP|h_w\big) \\
&= \frac{|PQ|}{|P'Q'|} \cdot \frac{1}{2}\big(|P'Q'|h_u + |Q'R'|h_v + |R'P'|h_w\big) \\
&= \frac{|PQ|}{|P'Q'|}|P'Q'R'| = \frac{|PQ|}{|P'Q'|}\Big(\frac{|P'Q'|}{|PQ|}\Big)^2|PQR| \\
&= \frac{|P'Q'|}{|PQ|}|S| \geqslant \frac{|Z|}{|S|}|S| = |Z|.
\end{aligned}$$

◀

The following is a special case of Lemma 5.

▶ **Corollary 6.** *Let S and T be two triangles where T does not fit into S, and let Z be a universal cover for $\{S, T\}$. Let S' be the smallest triangle similar to S such that T fits in S'. If*

$$\frac{|S'|}{|S|} = \Big(\frac{|Z|}{|S|}\Big)^2,$$

then Z is a smallest universal cover for the family $\{S, T\}$.

Now we are ready to sketch the proof of Theorem 4. It is not too hard to show that T^* is indeed a universal cover for triangles in the unit circle. We then notice that the triangle $AD\tilde{C}$ is the smallest equilateral triangle into which $T_1(80°)$ fits. Thus, for optimality, we can apply Corollary 6 with $S = T_0 = \triangle ABC$, $T = T_1(80°) = \triangle DEF$, and $Z = T^* = \triangle ADC$ (recall Figure 3). In fact, there are many smallest universal covers (of the same area) for the family $\{T_0, T_1(80°)\}$. However, we can show that any smallest universal cover that accommodates $T_1(\theta)$ for every θ should coincide with T^*; the proof is omitted. The uniqueness then follows immediately.

4 Two triangles

In the following theorem, we describe how to find a triangle that is a smallest universal cover for a given family of two triangles.

▶ **Theorem 7.** *Let S and T be triangles. Then there is a triangle Z that is a smallest universal cover for the family $\{S, T\}$.*

Proof. If S fits in T or if T fits in S, the statement is true, so we assume that this is not the case. Let S' be the smallest triangle similar to S such that T fits in S'. This implies that T maximally fits into S', so by Lemma 2 there are three cases. We denote S by $\triangle ABC$, S' by $\triangle A'B'C'$, and T by $\triangle PQR$.
Case 1. P and Q lie on the edge $A'B'$, and $R = C'$; see Figure 5. We first observe

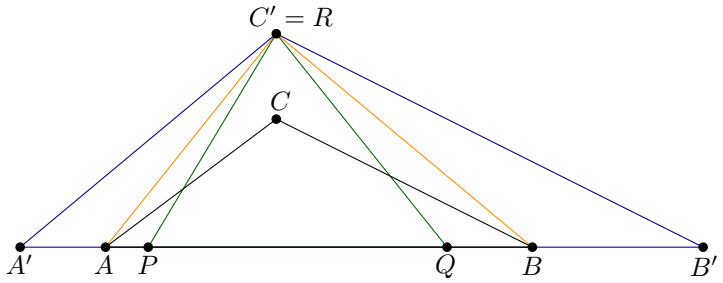


Figure 5 Proof of Theorem 7 - Case 1.

that $|AB| > |PQ|$: otherwise, we can place the segment AB inside the segment PQ, which causes C to fall inside T, and S to fit into T, a contradiction. We can therefore place AB inside $A'B'$ so that it covers PQ and C lies inside T. Then the triangle $Z = \triangle ABR$ is a universal cover for S and T. Then

$$\frac{|Z|}{|S|} = \frac{|ABR|}{|ABC|} = \frac{|A'C'|}{|AC|} \qquad \text{and} \qquad \frac{|S'|}{|S|} = \left(\frac{|A'C'|}{|AC|}\right)^2,$$

so by Corollary 6 Z is a smallest universal cover for $\{S, T\}$.
Case 2. PQ coincides with $A'B'$; see Figure 6. Let h_R be the height of R in T, let h_C be the height of C in S. We have $h_C > h_R$, since otherwise S fits into T by a placement in which $C = R$ and AB is parallel to PQ. We can therefore place $S = \triangle ABC$ such that A and B are on the segment $A'B'$ and C is on the segment RC'. Then $Z = \triangle PQC$ is a smallest universal cover for $\{S, T\}$ by Corollary 6 since

$$\frac{|Z|}{|S|} = \frac{|PQC|}{|ABC|} = \frac{|PQ|}{|AB|} = \frac{|A'B'|}{|AB|} \qquad \text{and} \qquad \frac{|S'|}{|S|} = \left(\frac{|A'B'|}{|AB|}\right)^2.$$

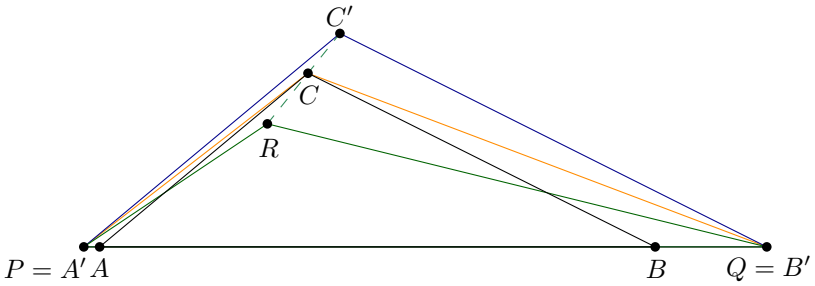


Figure 6 Proof of Theorem 7 - Case 2.

Case 3. P coincides with A', Q lies on the edge $A'B'$, and R lies on the edge $B'C'$. Let again h_R be the height of R in T, let h_C be the height of C in S.

If $h_C \geqslant h_R$, then we can place $S = \triangle ABC$ such that $B = B'$, A lies on $A'B'$, and C lies on the segment RC'; see Figure 7. Then $Z = \triangle PBC$ is a smallest universal cover for $\{S, T\}$

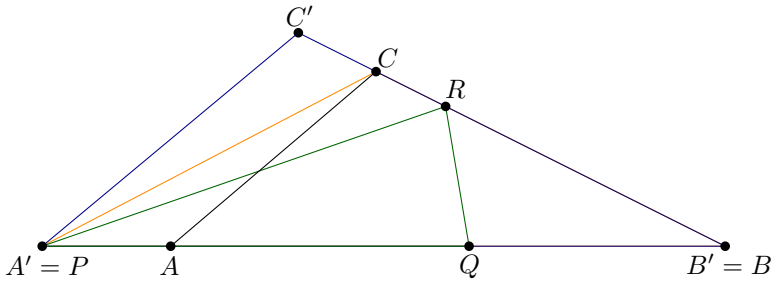


Figure 7 Proof of Theorem 7 - Case 3 when $h_C \geqslant h_R$.

by Corollary 6 since

$$\frac{|Z|}{|S|} = \frac{|PBC|}{|ABC|} = \frac{|PB|}{|AB|} = \frac{|A'B'|}{|AB|} \qquad \text{and} \qquad \frac{|S'|}{|S|} = \left(\frac{|A'B'|}{|AB|}\right)^2.$$

It remains to consider the case where $h_C < h_R$. Then we place $S = \triangle ABC$ such that C is on the edge PR while A and B are on $A'B'$; see Figure 8. We let $Z = \triangle PBR$. We

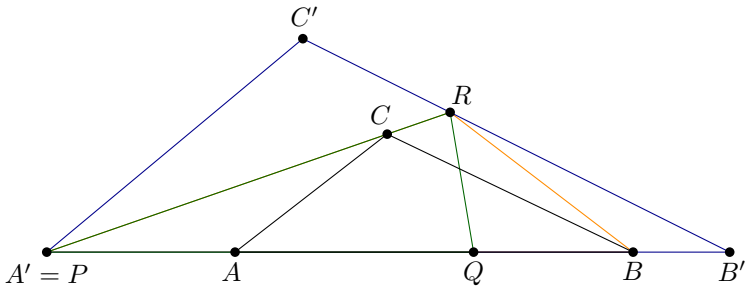


Figure 8 Proof of Theorem 7 - Case 3 when $h_C < h_R$.

observe that $|CBR| = |CBB'|$, since the two triangles have the same base and the same height, as $B'C'$ is parallel to BC. Therefore

$$\frac{|Z|}{|S|} = \frac{|PBR|}{|ABC|} = \frac{|PB'C|}{|ABC|} = \frac{|PB'|}{|AB|} = \frac{|A'B'|}{|AB|}.$$

On the other hand,

$$\frac{|S'|}{|S|} = \Big(\frac{|A'B'|}{|AB|}\Big)^2,$$

so Corollary 6 again implies that Z is a smallest universal cover for $\{S, T\}$. ◀

5 Two triangles are not enough

▶ **Theorem 8.** *There exists a three-element family $\mathcal{T}_3 = \{\triangle ABC, \triangle DEF, \triangle GHI\}$ whose smallest universal cover is larger than a smallest universal cover for any two of the triangles.*

Proof. (Sketch) We start by constructing three triangles as follows:

- $\triangle ABC$ is an equilateral triangle of side length 2 and thus of height $\sqrt{3}$;
- $\triangle DEF$ is an isosceles triangle where $|DF| = |EF|$, $|DE| = 6$, and the height of F is $\sqrt{3}/(1+\varepsilon)$;
- $\triangle GHI$ is an isosceles triangle with $|GI| = |HI|$, the height of G and H is ε, and the projection KI of HI on GI has length $6 - \varepsilon$.

By applying Theorem 7, we can conclude that, for each proper subfamily of $\mathcal{T}_3$, a smallest universal cover is of area at most $3\sqrt{3}$. Now we assume for a contradiction that a universal cover for $\mathcal{T}_3$ of area $3\sqrt{3}$ exists, and proceed as in the proof of Lemma 5. ◀

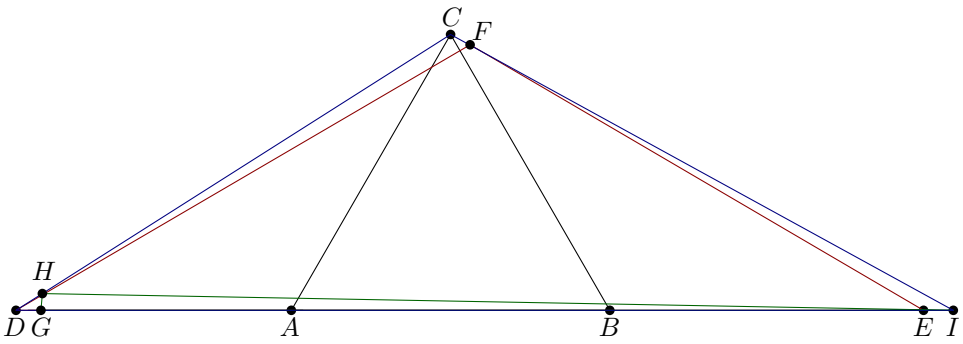


Figure 9 A family of three triangles whose every proper subfamily has a smaller universal cover.

We conjecture that if we arrange the three triangles as in Figure 9 (so that H lies on CD and F lies on CI), then $\triangle CDI$ is the unique smallest universal cover for $\mathcal{T}_3$.

Acknowledgments

This work was initiated during the 18th Korean Workshop on Computational Geometry in Otaru. The authors would like to thank the other participants for suggesting the problem and the interesting discussions during the workshop.

References

1 Pankaj K. Agarwal, Nina Amenta, and Micha Sharir. Largest placement of one convex polygon inside another. *Discrete & Computational Geometry*, 19:95–104, 1998. doi:10.1007/PL00009337.

2 Hee-Kap Ahn, Sang Won Bae, Otfried Cheong, Joachim Gudmundsson, Takeshi Tokuyama, and Antoine Vigneron. A generalization of the convex Kakeya problem. *Algorithmica*, 70:152–170, 2014.

3 John C. Baez, Karine Bagdasaryan, and Philip Gibbs. The Lebesgue universal covering problem. *Journal of Computational Geometry*, 6:288–299, 2015. doi:10.20382/jocg.v6i1a12.

4 Peter Brass, William Moser, and János Pach. *Research Problems in Discrete Geometry*. Springer-Verlag, 2005.

5 Peter Brass and Mehrbod Sharifi. A lower bound for Lebesgue's universal cover problem. *International Journal of Computational Geometry & Applications*, 15:537–544, 2005. doi:10.1142/S0218195905001828.

6 Zoltan Füredi and John E. Wetzel. The smallest convex cover for triangles of perimeter two. *Geometriae Dedicata*, 81:285–293, 2000. doi:10.1023/A:1005298816467.

7 Mikhail D. Kovalev. A minimal convex covering for triangles (in Russian). *Ukrain. Geom. Sb.*, 26:63–68, 1983.

8 Julius Pál. Über ein elementares Variationsproblem. *Math.-fys. Medd., Danske Vid. Selsk.*, 3, 1920.

9 John E. Wetzel. Fits and covers. *Mathematics Magazine*, 76:349–363, 2003.

10 Liping Yuan and Ren Ding. The smallest triangular cover for triangles of diameter one. *Journal of Applied Mathematics and Computing*, 17:39–48, 2005. doi:10.1007/BF02936039.

Between Two Shapes, Using the Hausdorff Distance*

Marc van Kreveld[1], Tillman Miltzow[1], Tim Ophelders[2], Willem Sonke[3], and Jordi Vermeulen[1]

1 **Dep. of Information and Computing Sciences, Utrecht University**
{m.j.vankreveld|t.miltzow|j.l.vermeulen}@uu.nl
2 **Dep. of Computational Mathematics, Science and Engineering, Michigan State University**
ophelder@egr.msu.edu
3 **Dep. of Mathematics and Computer Science, TU Eindhoven**
w.m.sonke@tue.nl

Abstract

Given two shapes A and B in the plane with Hausdorff distance 1, is there a shape S with Hausdorff distance $1/2$ to and from A and B? The answer is always yes, and depending on convexity of A and/or B, S may be convex, connected, or disconnected. We show a generalization of this result and a few others about Hausdorff distances, middle shapes, and related properties. We also show that the implied morph has a bounded rate of change.

1 Introduction

The Hausdorff distance is a widely used distance metric with many applications. For two sets A and B in $\mathbb{R}^2$, we define the *directed Hausdorff distance* as

$$d_{\vec{H}}(A,B) = \sup_{a\in A}\inf_{b\in B} d(a,b),$$

where d denotes the Euclidean distance. The *undirected Hausdorff distance* is defined as

$$d_H(A,B) = \max(d_{\vec{H}}(A,B), d_{\vec{H}}(B,A)).$$

If A and B are closed sets then $d_H(A,B) = r$ is equivalent to saying that $A \subseteq B \oplus D_r$ and $B \subseteq A \oplus D_r$, where $\oplus$ denotes the Minkowski sum, and D_r is a disk of radius r centered at the origin. Recall that the Minkowski sum of sets A and B is the set $\{a+b \mid a \in A,\ b \in B\}$. As we will use this alternative definition throughout the paper, we will only consider closed sets.

Algorithms to compute the Hausdorff distance are available for many types of input sets, such as points, line segments, polylines, polygons and simplices in k-dimensional Euclidean space [1, 2, 4]. However, the question whether a polynomial time algorithm exists to compute the Hausdorff distance between general semialgebraic sets remains open [5].

In this paper, we consider the problem of finding a third set that lies "between" the two input sets, in a Hausdorff sense. We define a class of sets that smoothly interpolate between the two input sets, giving a morph between them. Unlike many existing morphing algorithms [3, 6], our approach does not require any correspondence between features of the

* Research on the topic of this paper was initiated at the 4th Workshop on Applied Geometric Algorithms (AGA 2018) in Langbroek, The Netherlands, supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 639.023.208. The second author is supported by the NWO Veni grant EAGER. The first and fifth authors are supported by the NWO TOP grant no. 612.001.651

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

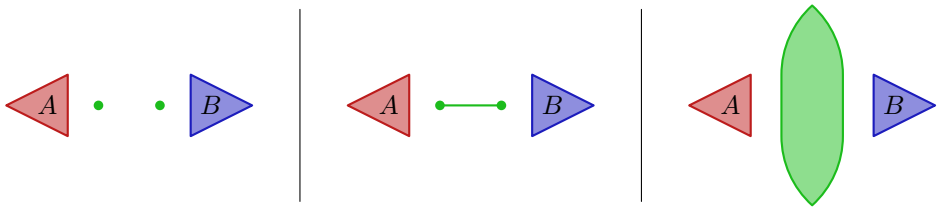


Figure 1 Three possible Hausdorff middles of A and B: two points, a line segment, and $S_{1/2}$.

input to be calculated. However, our approach is unusual in the sense that the intermediate shapes when morphing between e.g. two polygons are not polygons themselves.

Our main contribution is to pioneer the notion of Hausdorff middle and the interpolation between two shapes. We address and solve elementary combinatorial, topological, and algorithmic questions.

2 The Hausdorff middle

Consider two closed compact sets A and B in $\mathbb{R}^2$; we are interested in computing a *Hausdorff middle*: a set C that minimizes the maximum of the undirected Hausdorff distances to A and B. That is,

$$C = \operatorname*{argmin}_{C'} \max(d_H(A, C'), d_H(B, C')).$$

Note that there may be many such sets that minimize the Hausdorff distance; see Figure 1 for a few examples. It might seem intuitive to restrict C to be the minimal set that achieves this distance, but this is ill-defined: the minimal set is not necessarily unique, and the common intersection of all minimal sets is not a solution itself (see Figure 2). However, the maximal set is well-defined and unique. Let $d_H(A, B) = 1$. Then

$$S = (A \oplus D_{1/2}) \cap (B \oplus D_{1/2})$$

is the maximal set satisfying the constraints. We want to show that $d_H(A, S)$ and $d_H(B, S)$ are at most $1/2$. In fact, we can prove something more general where we define maximal sets that may be at other places between A and B, not only halfway.

▶ **Lemma 1.** *Let A and B be two closed sets in the plane with $d_H(A, B) = 1$, and let $S_\alpha = (A \oplus D_\alpha) \cap (B \oplus D_{1-\alpha})$ for $\alpha \in [0, 1]$. Then $d_H(A, S_\alpha) = \alpha$ and $d_H(B, S_\alpha) = 1 - \alpha$.*

Proof. We first show that $d_H(A, S_\alpha) \leq \alpha$ by showing that $d_{\vec{H}}(A, S_\alpha) \leq \alpha$ and $d_{\vec{H}}(S_\alpha, A) \leq \alpha$; the case for $d_H(B, S_\alpha) \leq 1-\alpha$ is analogous and therefore omitted. Then we show equality.

Consider any point $a \in A$; by definition, there is a point $b \in B$ with $d(a, b) \leq 1$. Now consider a point $s \in \text{seg}(a, b)$ with $d(a, s) \leq \alpha$ and $d(b, s) \leq 1 - \alpha$; clearly this point must

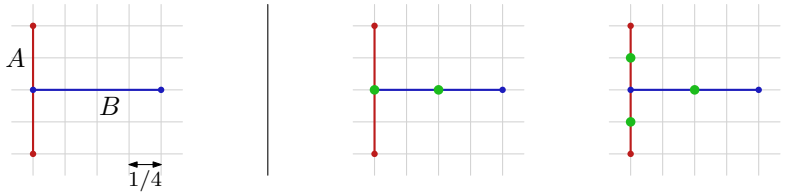


Figure 2 Two different minimal sets achieving minimal Hausdorff distance to A and B.

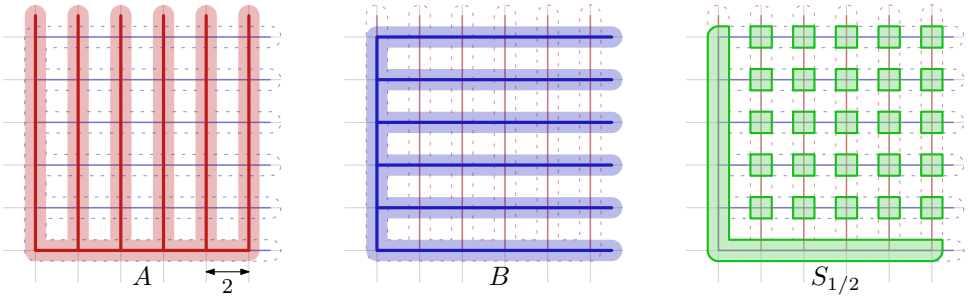


Figure 3 Sets A and B for which $S_{1/2}$ is disconnected. The shaded areas around A and B represent $A \oplus D_{1/2}$ and $B \oplus D_{1/2}$, respectively.

be in S_α, as it is contained in both $A \oplus D_\alpha$ and $B \oplus D_{1-\alpha}$, and it has $d(a, s) \leq \alpha$. As this works for every $a \in A$, it holds that $d_{\vec{H}}(A, S_\alpha) \leq \alpha$. The fact that $d_{\vec{H}}(S_\alpha, A) \leq \alpha$ follows straightforwardly from S_α being a subset of $A \oplus D_\alpha$. Thus, $d_H(A, S_\alpha) \leq \alpha$.

To show equality, assume that the Hausdorff distance is realized by a point $\hat{a} \in A$ with closest point $\hat{b} \in B$, at distance 1. Consider the point $\hat{s} \in \text{seg}(\hat{a}, \hat{b})$ with $d(\hat{a}, \hat{s}) = \alpha$ and $d(\hat{b}, \hat{s}) = 1 - \alpha$. As observed, $\hat{s} \in S_\alpha$. Since $\hat{s}$ is the closest point of S_α to $\hat{a}$, and $\hat{b}$ is the closest point of B to $\hat{s}$, equality follows. ◀

▶ **Lemma 2.** *S_α is the maximal set that satisfies $d_H(A, S_\alpha) = \alpha$ and $d_H(B, S_\alpha) = 1 - \alpha$.*

Proof. Consider any set T for which we have $d_{\vec{H}}(T, A) \leq \alpha$ and $d_{\vec{H}}(T, B) \leq 1-\alpha$. As $A \oplus D_\alpha$ contains all points with distance at most α to A, we have that $T \subseteq A \oplus D_\alpha$; similarly, we have that $T \subseteq B \oplus D_{1-\alpha}$. By the definition of S_α, this implies that $T \subseteq S_\alpha$. As this holds for any T, we conclude that S_α is maximal. ◀

2.1 Properties of S_α

In this section, we study the convexity and connectedness of S_α. Recall that a set $A \subseteq \mathbb{R}^2$ is convex if for any two points $a, b \in A$, the segment $\bar{ab}$ between them is completely contained in A. Also, recall that a set $A \subseteq \mathbb{R}^2$ is connected if for any two points $a, b \in A$, there exists a path from a to b completely contained in A.

1. if A and B are convex, S_α is convex;
2. if A is convex and B is connected, S_α is connected;
3. if A and B are connected, but neither A nor B is convex, S_α may be disconnected.

Property 1 is straightforward: the Minkowski sum of A and B with a disk is convex, and the intersection of convex objects is itself also convex. The example in Figure 3 demonstrates Property 3. We show Property 2 next.

We say a set $A \subset \mathbb{R}^2$ is connected if for any two points $a, b \in A$, there exists a continuous curve $c : [0, 1] \to A$ such that $c(0) = a$ and $c(1) = b$. This type of connectedness is known as path-connectedness. Note that the empty set is trivially connected. The following observation is straightforward:

▶ **Observation 3.** *Let A and B be two connected sets in the plane. If $A \cap B$ is not connected, $A \cup B$ contains a hole.*

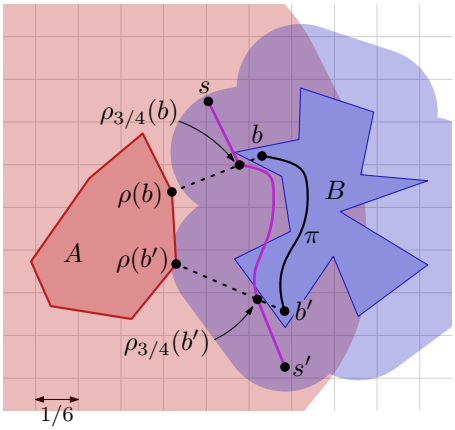


Figure 4 Example of the construction from Lemma 4 showing that S_α is connected if A is convex (sketched here for $\alpha = 3/4$). The shaded areas around A and B represent $A \oplus D_{3/4}$ and $B \oplus D_{1/4}$, respectively, so that the doubly-shaded area is $S_{3/4}$.

The next lemma establishes property 2:

▶ **Lemma 4.** *Let A and B be two connected polygons with Hausdorff distance 1, and A convex. Then $S_\alpha = (A \oplus D_\alpha) \cap (B \oplus D_{1-\alpha})$ is connected.*

Proof. The argument is as follows (see Figure 4 for a sketch):

- Because A is convex, there is a continuous map $\rho\colon B \to A$ that maps each point of B to a closest point (within distance 1) in A.
- For $b \in B$, let $\rho_\alpha(b) = b + \alpha(\rho(b) - b)$. We have $\rho_\alpha\colon B \to S_\alpha$ which is also continuous.
- Now take any two points s and s' in S_α; respectively, they have points b and $b' \in B$ within distance $1 - \alpha$.
- The segments between s and $\rho_\alpha(b)$ and between s' and $\rho_\alpha(b')$ lie completely in S_α.
- Take a path π from b to b' inside B. The image of π under ρ_α connects $\rho_\alpha(b)$ to $\rho_\alpha(b')$ within S_α, so s and s' are connected inside S_α. ◀

We note that S_α may contain holes. Furthermore, S_α is not shape invariant when B is translated with respect to A. For example, let A be the union of the left and bottom sides of a unit square and let B_1 and B_2 be the left and right sides of that same unit square. Then $(A \oplus D_{1/2}) \cap (B_1 \oplus D_{1/2})$ is not a translate of $(A \oplus D_{1/2}) \cap (B_2 \oplus D_{1/2})$. See Figure 5.

2.2 Complexity of S_α

In this section, we describe the complexity of S_α in terms of the number of vertices, line segments, and circular arcs on its boundary, for several types of polygonal input sets. Recall that ∂A denotes the boundary of set A.

▶ **Lemma 5.** *Let A be a convex polygon and B a simple polygon with n, respectively m vertices. Then S_α consists of $\Theta(n+m)$ vertices, line segments and circular arcs in the worst case.*

Proof. There is a trivial worst-case lower bound of $\Omega(n + m)$ by taking $\alpha = 0$ or $\alpha = 1$. Note that if the boundaries of $A^\oplus = A \oplus D_\alpha$ and $B^\oplus = B \oplus D_{1-\alpha}$ would consist of only line segments, the upper bound is easy to show: $A^\oplus$ is convex, and its boundary can therefore intersect each segment of $\partial B^\oplus$ at most twice, making ∂S_α consist of (parts of) segments

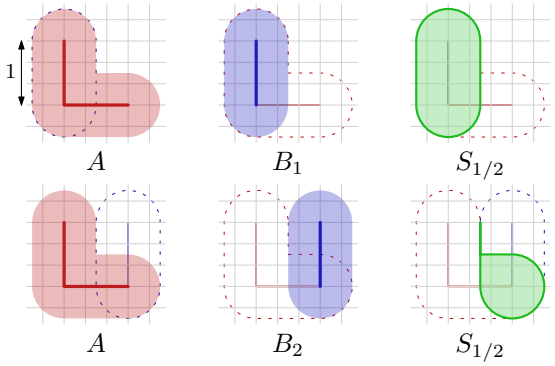


Figure 5 Although B_2 is a translate of B_1, the middle set between A_1 and B_2 is not a translate of the middle set between A_1 and B_1.

from $\partial A^{\oplus}$ and $\partial B^{\oplus}$ and at most $O(m)$ intersection points. The problem is that $\partial A^{\oplus}$ and $\partial B^{\oplus}$ also contain circular arcs, in which case an arc of $\partial B^{\oplus}$ may intersect $\partial A^{\oplus}$ many times.

To show an upper bound of $O(n+m)$, we distinguish two cases. In the first case, we assume $\alpha \geq 1-\alpha$. Note that in this case, the circular arcs that are part of the boundary of $A^{\oplus} = A \oplus D_\alpha$ have a radius larger or equal to those of $B^{\oplus} = B \oplus D_{1-\alpha}$. In this case, we do in fact have that any line segment or circular arc b of $\partial B^{\oplus}$ can intersect $\partial A^{\oplus}$ at most twice: for any circular arc of $\partial A^{\oplus}$, the entire disk that it bounds must be contained in $A^{\oplus}$. This means that b either intersects the same feature of $\partial A^{\oplus}$ twice, or it intersects two different features once.

For the second case, we assume $\alpha < 1-\alpha$. Again, take an arbitrary arc b of $\partial B^{\oplus}$ that intersects some arc a of $\partial A^{\oplus}$. We distinguish two cases: the center point of the disk whose boundary contains a is inside $B^{\oplus}$, or it is outside. If it is outside, b can only intersect $\partial A^{\oplus}$ in two points. If it is inside, $\partial A^{\oplus}$ may intersect b many times. We charge these intersections to the arcs of $\partial A^{\oplus}$. We argue that each arc a of $\partial A^{\oplus}$ is charged at most four times: Consider any α-disk D_α and any $(1-\alpha)$-disk $D_{1-\alpha}$ containing the center of D_α, the latter will cover at least $1/3$ of the perimeter of the former. Hence, the boundary of the union of any number of such $(1-\alpha)$-disks intersects D_α at most four times. The circular arcs of $\partial A^{\oplus}$ cannot be charged more often because they are less than a full circle. ◀

▶ **Lemma 6.** *Let A and B be two simply connected polygons of n and m vertices, respectively. Then S_α consists of $\Theta(nm)$ vertices, line segments and circular arcs in the worst case.*

Proof. We can show a worst-case lower bound of $\Omega(nm)$ by taking A and B to be two "combs" placed at right angles to each other; see Figure 3. In this example, for $\alpha = 1/2$, S_α consists of $\Omega(nm)$ distinct components. The upper bound follows directly from the fact that $A^{\oplus} = A \oplus D_\alpha$ and $B^{\oplus} = B \oplus D_{1-\alpha}$ have complexities $O(n)$ and $O(m)$, respectively. ◀

2.3 S_α as a morph

By increasing α from 0 to 1, S_α morphs from $A = S_0$ into $B = S_1$. The following lemma shows that this morph has a bounded rate of change.

▶ **Lemma 7.** *Let S_α and S_β be two intermediate shapes of A and B with $\alpha \leq \beta$. Then $d_H(S_\alpha, S_\beta) = \beta - \alpha$.*

Proof. We have $d_H(S_\alpha, S_\beta) \geq \beta - \alpha$ because by the triangle inequality, $d_H(A,B) = 1 \leq d_H(A,S_\alpha) + d_H(S_\alpha, S_\beta) + d_H(S_\beta, B) \leq \alpha + d_H(S_\alpha, S_\beta) + 1 - \beta$.

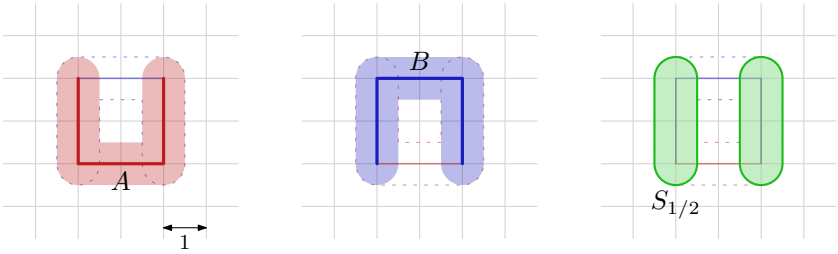


Figure 6 $S_{1/2}$ for the red and blue polygons is shown in green. Any connected Hausdorff middle must cross vertical middle line or stay on one side of it. In both cases, a Hausdorff distance doubles.

It remains to show that $d_H(S_\alpha, S_\beta) \leq \beta - \alpha$. We show that $S_\beta \subseteq S_\alpha \oplus D_{\beta-\alpha}$; the other case is analogous. Let p be some point in S_β. Then, by definition of S_β, there exist some points $a \in A$ and $b \in B$ such that $d(a, p) \leq \beta$ and $d(b, p) \leq 1-\beta$. Let $\bar{p}$ be the point obtained by moving p in the direction of a by $\beta - \alpha$. By the triangle inequality, we then have that $d(a, \bar{p}) \leq \beta - (\beta - \alpha) = \alpha$ and $d(b, \bar{p}) \leq (1 - \beta) + (\beta - \alpha) = 1 - \alpha$. This implies that $\bar{p} \in S_\alpha$. As p was an arbitrary point in S_β, and $d(p, \bar{p}) \leq \beta - \alpha$, we have that $S_\beta \subseteq S_\alpha \oplus D_{\beta-\alpha}$. So $d_H(S_\alpha, S_\beta) \leq \beta - \alpha$. ◀

The lemma implies that, even though the number of connected components of S_α can change when α changes, new components arise by splitting and never 'out of nothing', and the number of components can only decrease through merging and not by disappearance.

2.4 The cost of connectedness

For some applications, it might be necessary to insist that S_α is always connected. However, in the worst case, the cost of connecting all components of S_α can be that its Hausdorff distance to A and B becomes 1. See Figure 6 for an example where this is the case.

3 Conclusion

Besides the maximal middle set, there are other options for a Hausdorff middle. For example, we can choose S_α clipped to the convex hull of $A \cup B$, which is also a valid Hausdorff middle. In Figure 6, the green shape would be reduced to the part inside the square, which may be more natural. This Hausdorff middle can also be used in a morph.

A natural question is whether these results extend to more than two input shapes. The problem would then be to compute some shape S that minimizes the maximum pairwise Hausdorff distance to any of the input shapes, assuming that the pairwaise Hausdorff distance is at most 1. It turns out that in some cases, the minimum Hausdorff distance that

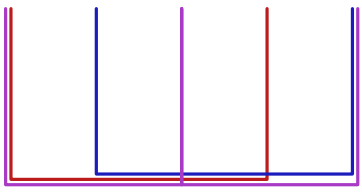


Figure 7 Assuming the pairwise Hausdorff distance is 1, any shape will have Hausdorff distance at least 1 to or from at least one of the three inputs.

can be achieved is 1, even when the input sets are connected; see Figure 7 for an example. When the input shapes are all convex, the worst example we have found so far is three sets of a single point each, forming an equilateral triangle. In this case, the best middle set is the centroid of the triangle, which has Hausdorff distance $1/\sqrt{3}$ to each input set. We conjecture that this is the highest possible optimal value for convex input sets.

References

1 H. Alt, B. Behrends, and J. Blömer. Approximate matching of polygonal shapes. *Annals of Mathematics and Artificial Intelligence*, 13(3):251–265, Sep 1995.

2 H. Alt, P. Braß, M. Godau, C. Knauer, and C. Wenk. Computing the Hausdorff distance of geometric patterns and shapes. In *Discrete and Computational Geometry*, pages 65–76. Springer, 2003.

3 H. Alt and L.J. Guibas. Discrete geometric shapes: Matching, interpolation, and approximation. In *Handbook of Computational Geometry*, pages 121–153. Elsevier, 2000.

4 M. J. Atallah. A linear time algorithm for the hausdorff distance between convex polygons. *Information Processing Letters*, 17(4):207 – 209, 1983.

5 M. G. Dobbins, L. Kleist, T. Miltzow, and P. Rzążewski. $\forall\exists\mathbb{R}$-completeness and area-universality. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 164–175. Springer, 2018.

6 C. Gotsman and V. Surazhsky. Guaranteed intersection-free polygon morphing. *Computers & Graphics*, 25(1):67–75, 2001. Shape Blending.

Representing Graphs by Polygons with Side Contacts in 3D*

Elena Arseneva[1], Linda Kleist[2], Boris Klemz[3], Maarten Löffler[4], André Schulz[5], Birgit Vogtenhuber[6], and Alexander Wolff[7]

1 **Saint Petersburg State University, Russia**
e.arseneva@spbu.ru
2 **Technische Universität Braunschweig, Germany**
kleist@ibr.cs.tu-bs.de
3 **Freie Universität Berlin, Germany**
klemz@inf.fu-berlin.de
4 **Utrecht University, the Netherlands**
m.loffler@uu.nl
5 **FernUniversität in Hagen, Germany**
andre.schulz@fernuni-hagen.de
6 **Technische Universität Graz, Austria**
bvogt@ist.tugraz.at
7 **Universität Würzburg, Germany**
orcid.org/0000-0001-5872-718X

Abstract

A graph has a side-contact representation with polygons if its vertices can be represented by interior-disjoint polygons such that two polygons share a common side if and only if the corresponding vertices are adjacent. In this work we study representations in 3D. We show that every graph has a side-contact representation with polygons in 3D, while this is not the case if we additionally require that the polygons are convex: we show that every supergraph of K_5 and every nonplanar 3-tree does not admit a representation with convex polygons. On the other hand, $K_{4,4}$ admits such a representation, and so does every graph obtained from a complete graph by subdividing each edge once. Finally, we construct an unbounded family of graphs with average vertex degree $12 - o(1)$ that admit side-contact representations with convex polygons in 3D. Hence, such graphs can be considerably denser than planar graphs.

1 Introduction

A graph has a contact representation if its vertices can be represented by interior-disjoint geometric objects[1] such that two objects touch exactly if the corresponding vertices are adjacent. In concrete settings, one usually restricts the set of geometric objects (disks, lines, polygons, ...), the type of contact, and the embedding space. Numerous results about which graphs admit a contact representation of some type are known. Giving a comprehensive overview is out of scope for this extended abstract. We therefore mention only few results. By the Andreev–Koebe–Thurston circle packing theorem [3, 20] every planar graph has a contact representation by touching disks in 2D. Contact representations of graphs in 2D have since been considered for quite a variety of shapes, including triangles [4, 8, 13, 14, 19],

* E.A. was partially supported by RFBR, project 20-01-00488; B.V. was partially supported by the Austrian Science Fund within the collaborative DACH project *Arrangements and Drawings* as FWF project I 3340-N35; A.W. acknowledges support by DFG project WO 758/9-1.

[1] If the considered objects are not fully-dimensional in the considered space then *interior-disjoint* is meant with respect to the relative interior of the objects.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

axis-aligned rectangles [1, 5, 11], curves [15], or line segments [7, 6, 16] in 2D and balls [17], tetrahedra [2] or cubes [12, 18] in 3D. Evans et al. [10] showed that every graph has a contact representation in 3D in which each vertex is represented by a convex polygon and two polygons touch *in a corner* if and only if the corresponding two vertices are adjacent.

In this work we study contact representations with polygons in 3D where a contact between two polygons is realized by sharing a proper side that is not part of any other polygon of the representation. (To avoid confusion with the corresponding graph elements, we consistently refer to polygon vertices as *corners* and to polygon edges as *sides*.) The special case where we require that the polygons are convex is of particular interest. Note that we do not require that the polyhedral complex induced by the contact representation is a closed surface. In particular, not every polygon side has to be in contact with another polygon. By Steinitz's theorem [21], every 3-connected planar graph can be realized as a convex polyhedron, whose dual is also a planar graph. Thus all planar graphs have such a representation with convex polygons.

Results. We show that for the case of nonconvex polygons, every graph has a side-contact representation in 3D. For convex polygons, the situation is more intricate. We show that certain graphs do not have such a representation, in particular all nonplanar 3-trees and all supergraphs of K_5. On the other hand, many nonplanar graphs (for example, $K_{4,4}$) have such a representation. In particular, graphs that admit side-contact representations with convex polygons in 3D can be considerably denser than planar graphs. Due to lack of space, several proofs are only sketched or completely deferred to the full version of this work.

2 Representations with General Polygons

First we show that every graph can be represented by nonconvex polygons; see Figure 1.

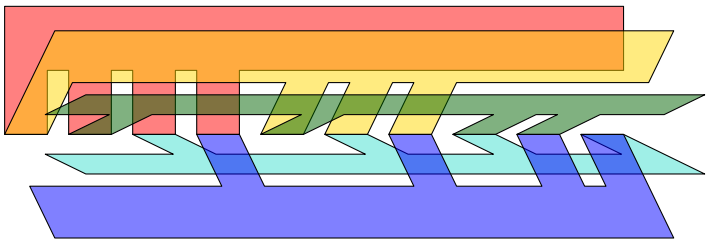


Figure 1 A realization of K_5 by nonconvex polygons with side contacts in 3D.

▶ **Proposition 2.1.** *Every graph can be realized by polygons with side contacts in 3D.*

Proof. To represent a graph G with n vertices, we start with n interior-disjoint rectangles such that there is a line segment s that acts as a common side of all these rectangles. We then cut away parts of each rectangle thereby turning it into a comb-shaped polygon as illustrated in Figure 1. This step ensures that for each pair (P, P') of polygons, there is a subsegment s' of s such that s' is a side of both P and P' that is disjoint from the remaining polygons. The result is a representation of K_n. To obtain a realization of G, it remains to remove side contacts that correspond to unwanted adjacencies, which is easy. ◀

If we additionally insist that each polygon shares all of its sides with other polygons, the polygons describe a closed volume. In this model, K_7 can be realized as the Szilassi

polyhedron; see Figure 2. The tetrahedron and the Szilassi polyhedron are the only two known polyhedra in which each face shares a side with each other face [22]. Which other graphs can be represented in this way remains an open problem.

Figure 2 The Szilassi polyhedron realizes K_7 by nonconvex polygons with side contacts in 3D [22].

3 Representations with Convex Polygons

We now consider the setting where each vertex of the given graph is represented by a convex polygon in 3D and two vertices of the given graph are adjacent if and only if their polygons intersect in a common side. (In most previous work, it was only required that the side of one polygon is contained in the side of the adjacent polygon. For example, Duncan et al. [9] showed that in this model every planar graph can be realized by hexagons in the plane and that hexagons are sometimes necessary.) Note that it is allowed to have sides that do not touch other polygons. Further, non-adjacent polygons may intersect at most in a common corner. We start with some simple observations.

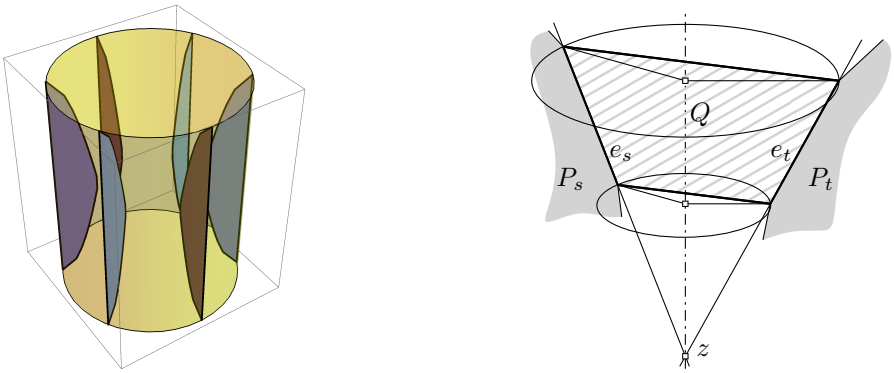
▶ **Proposition 3.1.** *Every planar graph can be realized by convex polygons with side contacts in 2D.*

Proof. Let G be a planar (embedded) graph with at least three vertices (for at most two vertices the statement is trivially true). Add to G a new vertex r and connect it to all vertices of some face. Let G' be a triangulation of the resulting graph. Then the dual $G^\star$ of G' is a cubic 3-connected planar graph. Using Tutte's barycentric method, draw $G^\star$ into a regular polygon with $\deg_{G'}(r)$ corners such that the face dual to r becomes the outer face. Note that the interior faces in this drawing are convex polygons. Hence the drawing is a side-contact representation of $G' - r$. To convert it to a representation of G, we may need to remove some side contacts, which can be easily achieved. ◀

Note that Proposition 3.1 also follows directly from the Andreev–Koebe–Thurston circle packing theorem. So for planar graphs, corner and side contacts behave similarly. For nonplanar graphs (for which we need the third dimension), the situation is different. Here, side contacts are more restrictive. We introduce the following notation. In a 3D representation of a graph G by polygons, we denote by P_v the polygon that represents vertex v of G.

▶ **Lemma 3.2.** *Let G be a graph. Consider a 3D side-contact representation of G with convex polygons. If G contains a triangle uvw, polygons P_v and P_w lie in the same halfspace with respect to the supporting plane of P_u.*

Proof. Due to their convexity, P_v and P_w must lie in the same halfspace with respect to the plane that supports P_u, otherwise P_v and P_w cannot share a side. In this case, the edge vw of G would not be represented; a contradiction. ◀



(a) The arrangement of the polygons $P_1, \ldots, P_n$. **(b)** Quadrilateral Q spanned by e_s and e_t.

Figure 3 Illustration for the proof of Proposition 3.4.

▶ **Proposition 3.3.** *For $n \geq 5$, K_n is not realizable by convex polygons with side contacts in 3D.*

Proof. Assume that K_n admits a 3D side-contact representation. Since every three vertices in K_n are pairwise connected, by Lemma 3.2, for every polygon of the representation, its supporting plane has the remaining polyhedral complex on one side. In other words, the complex we obtain is a subcomplex of a convex polyhedron. Consequently, the dual graph has to be planar, which rules out K_n for $n \geq 5$. ◀

▶ **Proposition 3.4.** *Let K'_n be the subdivision of the complete graph K_n in which every edge is subdivided with one vertex. For every n, K'_n has a side-contact representation with convex polygons in 3D.*

Proof sketch. For $n \leq 4$ the statement is true by Proposition 3.1. We sketch the construction of a representation for $n \geq 5$; see Figure 3. Let P be a convex polygon with $k = 2\binom{n}{2}$ corners, called $v_1, v_2, \ldots, v_k$, such that v_1v_k is a long side and the remaining corners form a flat convex chain connecting v_1 and v_k. We represent each high-degree vertex of K'_n by a copy of P. We arrange those copies in pairwise different vertical planes containing the z-axis such that all copies of v_1v_k are arranged vertically at the same height and at the same distance from the z-axis; and such that the convex chain of each copy of P faces the z-axis but does not intersect it. Consider two different copies P_s and P_t of P in this arrangement. They contain copies e_s and e_t of the same side e of P. It can be shown that e_s and e_t are coplanar. Moreover, they form a convex quadrilateral Q that does not intersect the arrangement except in e_s and e_t. We arbitrarily assign each side $v_{2i-1}v_{2i}$, $1 \leq i \leq k/2 = \binom{n}{2}$, to some edge st of K_n and use the quadrilateral Q spanned by e_s and e_t to represent the subdivision vertex of st in K'_n. As any two such quadrilaterals are vertically separated and hence disjoint, those $\binom{n}{2}$ quadrilaterals together with the n copies of P constitute a valid representation of K'_n. ◀

▶ **Proposition 3.5.** *$K_{4,4}$ is realizable by convex polygons with side contacts in 3D.*

Proof sketch. We sketch how to obtain a realization. Start with a box in 3D and intersect it with two rectangular slabs as indicated in Figure 4 on the left. We can now draw polygons on the faces of this complex such that each of the four vertical faces contains a polygon that has a side contact with a polygon on each of the four horizontal or slanted faces. The polygons

on the slanted faces lie in the interior of the box and intersect each other. To remove this intersection, we pull out one corner of the original box; see Figure 4. ◀

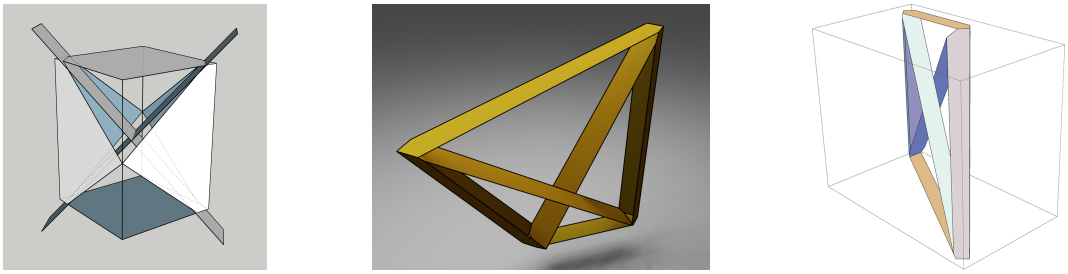


Figure 4 A realization of $K_{4,4}$ by convex polygons with side contacts in 3D.

In contrast to Proposition 3.5, we believe that the analogous statement does not hold for all bipartite graphs, i.e., we conjecture the following.

▶ **Conjecture 3.6.** *There exist values n and m such that the complete bipartite graph $K_{m,n}$ is not realizable by convex polygons with side contacts in 3D.*

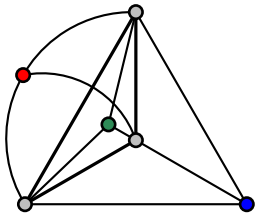
By Proposition 3.1, all planar 3-trees can be realized by convex polygons with side contacts (even in 2D). On the other hand, we can show that no nonplanar 3-tree has a realization in 3D. To this end, we prove the following two propositions, the first of which easily follows from the definition of 3-trees.

▶ **Lemma 3.7.** *A 3-tree is nonplanar if and only if it contains the graph depicted in Figure 5a as a subgraph.*

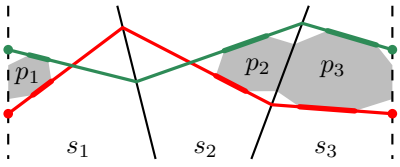
▶ **Lemma 3.8.** *The 3-tree depicted in Figure 5a cannot be realized by convex polygons with side contacts in 3D.*

▶ **Theorem 3.9.** *A 3-tree admits a side-contact representation with convex polygons in 3D if and only if it is planar.*

It is an intriguing question how dense graphs that admit a side-contact representation with convex polygons in 3D can be. In contrast to the results for corner contacts [10] and nonconvex polygons (Proposition 2.1) in 3D, we could not find a construction with a superlinear number of edges. The following construction yields the densest graphs we know.



(a) A 3-tree that is not realizable by convex polygons with side contacts in 3D. The gray vertices form a 3-cycle.

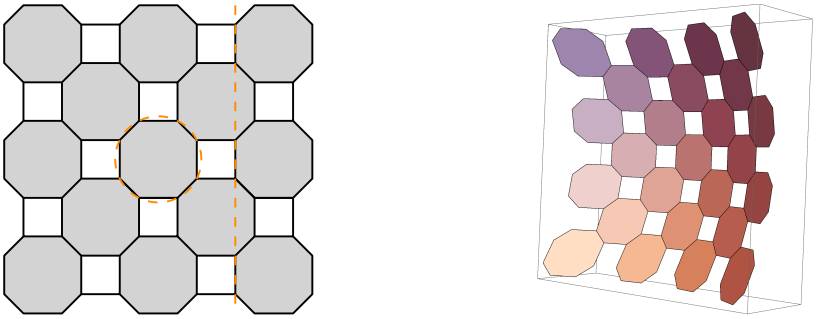


(b) Schematic drawing of a potential realization. Net of the three gray polygons and traces of the planes that contain the red and green polygons, which must touch each of the gray polygons. The line of intersection between two of the gray polygons is drawn twice (dashed).

Figure 5 Illustrations for the proof of Lemma 3.8.

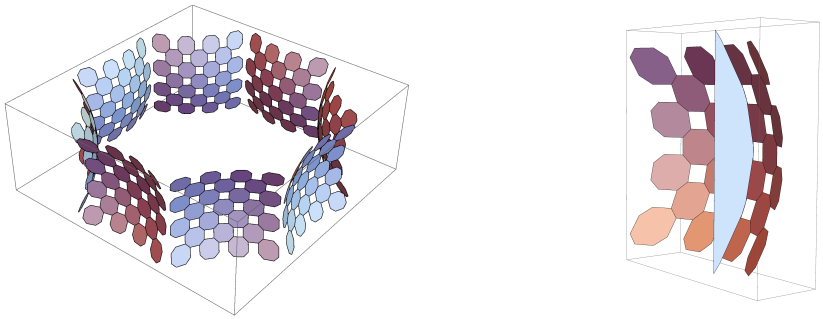
▶ **Theorem 3.10.** *There is an unbounded family of graphs with average vertex degree* $12-o(1)$ *that admit side-contact representations with convex polygons in 3D.*

Proof sketch. We first construct a contact representation of $m = \lceil\sqrt{n}\rceil$ regular octagons arranged as in a truncated square tiling; see Figure 6(a). Since the underlying geometric graph of the tiling is a Delaunay tessellation, we can lift the points to the paraboloid such that each octagon is lifted to coplanar points. We call the corresponding (scaled and rotated) polyhedral complex Γ; see Figure 6(b). Next we place $\lfloor\sqrt{n}\rfloor$ copies of Γ in a cyclic fashion as



(a) The plane octagon structure. **(b)** The complex Γ.

Figure 6 Proof of Theorem 3.10: construction of the complex Γ.



(a) The placement of the copies of Γ. **(b)** One vertical polygon.

Figure 7 Proof of Theorem 3.10: placement of the copies of Γ and vertical polygons.

shown in Figure 7(a) and we add vertical polygons in to generate a contact with the $\Theta(\sqrt{m})$ vertical sides of the octagons; see Figure 7(b). Finally, we introduce horizontal polygons in the "inner space" of our construction such that each of these polygons touches a specific side in each copy of Γ, as illustrated in Figure 8(a). A slight perturbation fixes the following two issues: First, many of the horizontal polygons lie on the same plane and intersect each other. Second, many sides of vertical polygons run along the faces of Γ. To fix these problems we modify the initial grid slightly; see Figure 8(b). ◀

4 Conclusion and Open Problems

Applying Turán's theorem [23] to Proposition 3.3 yields that the maximum number $e_{\text{cp}}(n)$ of edges in an n-vertex graph that admits a side-contact representation with convex polygons is at most $\frac{3}{8}n^2$. Theorem 3.10 gives a lower bound of $6n - o(n)$ for $e_{\text{cp}}(n)$. We tend to believe

(a) The location of a horizontal polygon as seen in a cross section.

(b) The modifications for Γ to separate nondisjoint faces.

Figure 8 Proof of Theorem 3.10: horizontal polygons and final modifications.

that the latter is closer to the truth than the former and conclude with the following open problem.

▶ **Question 4.1.** *What is the maximum number $e_{\text{cp}}(n)$ of edges that an n-vertex graph admitting a side-contact representation with convex polygons can have?*

Acknowledgements. This work has been initiated at the Dagstuhl Seminar 19352 "Computation in Low-Dimensional Geometry and Topology". We thank all the participants for the great atmosphere and fruitful discussions. Arnaud de Mesmay raised the question about side contacts. We also thank the anonymous referees for helpful comments, especially with respect to Theorem 3.9.

References

1 Md. Jawaherul Alam, Therese C. Biedl, Stefan Felsner, Andreas Gerasch, Michael Kaufmann, and Stephen G. Kobourov. Linear-time algorithms for hole-free rectilinear proportional contact graph representations. *Algorithmica*, 67(1):3–22, 2013. doi:10.1007/s00453-013-9764-5.

2 Md. Jawaherul Alam, Muriel Dulieu, Justin Iwerks, and Joseph O'Rourke. Tetrahedron contact graphs. In *Fall Workshop Comput. Geom.*, 2013.

3 E. M. Andreev. Convex polyhedra in Lobačevskiĭ spaces. *Mat. Sb. (N.S.)*, 81 (123)(3):445–478, 1970. doi:10.1070/SM1970v010n03ABEH001677.

4 Melanie Badent, Carla Binucci, Emilio Di Giacomo, Walter Didimo, Stefan Felsner, Francesco Giordano, Jan Kratochvíl, Pietro Palladino, Maurizio Patrignani, and Francesco Trotta. Homothetic triangle contact representations of planar graphs. In Prosenjit Bose, editor, *Proc. Canadian Conf. Comput. Geom. (CCCG'07)*, pages 233–236, 2007. URL: http://cccg.ca/proceedings/2007/09b4.pdf.

5 Adam L. Buchsbaum, Emden R. Gansner, Cecilia M. Procopiuc, and Suresh Venkatasubramanian. Rectangular layouts and contact graphs. *ACM Trans. Algorithms*, 4(1), 2008. doi:10.1145/1328911.1328919.

6 Hubert de Fraysseix and Patrice Ossona de Mendez. Representations by contact and intersection of segments. *Algorithmica*, 47(4):453–463, 2007. doi:10.1007/s00453-006-0157-x.

7 Hubert de Fraysseix, Patrice Ossona de Mendez, and János Pach. Representation of planar graphs by segments. *Intuitive Geometry*, 63:109–117, 1991. URL: https://infoscience.epfl.ch/record/129343/files/segments.pdf.

8 Hubert de Fraysseix, Patrice Ossona de Mendez, and Pierre Rosenstiehl. On triangle contact graphs. *Combinatorics, Probability and Computing*, 3:233–246, 1994. doi:10.1017/S0963548300001139.

9 Christian A. Duncan, Emden R. Gansner, Y. F. Hu, Michael Kaufmann, and Stephen G. Kobourov. Optimal polygonal representation of planar graphs. *Algorithmica*, 63(3):672–691, 2012. doi:10.1007/s00453-011-9525-2.

10 William Evans, Paweł Rzążewski, Noushin Saeedi, Chan-Su Shin, and Alexander Wolff. Representing graphs and hypergraphs by touching polygons in 3D. In Daniel Archambault and Csaba Tóth, editors, *Proc. Graph Drawing & Network Vis. (GD'19)*, volume 11904 of *LNCS*, pages 18–32. Springer, 2019. URL: http://arxiv.org/abs/1908.08273, doi:10.1007/978-3-030-35802-0_2.

11 Stefan Felsner. Rectangle and square representations of planar graphs. In János Pach, editor, *Thirty Essays on Geometric Graph Theory*, pages 213–248. Springer, 2013. doi:10.1007/978-1-4614-0110-0_12.

12 Stefan Felsner and Mathew C. Francis. Contact representations of planar graphs with cubes. In Ferran Hurtado and Marc J. van Kreveld, editors, *Proc. 27th Ann. Symp. Comput. Geom. (SoCG'11)*, pages 315–320. ACM, 2011. doi:10.1145/1998196.1998250.

13 Emden R. Gansner, Yifan Hu, and Stephen G. Kobourov. On touching triangle graphs. In Ulrik Brandes and Sabine Cornelsen, editors, *Proc. Graph Drawing (GD'10)*, volume 6502 of *LNCS*, pages 250–261. Springer, 2010. doi:10.1007/978-3-642-18469-7.

14 Daniel Gonçalves, Benjamin Lévêque, and Alexandre Pinlou. Triangle contact representations and duality. *Discrete Comput. Geom.*, 48(1):239–254, 2012. doi:10.1007/s00454-012-9400-1.

15 Petr Hliněný. Classes and recognition of curve contact graphs. *J. Combin. Theory Ser. B*, 74(1):87–103, 1998. doi:10.1006/jctb.1998.1846.

16 Petr Hliněný. Contact graphs of line segments are NP-complete. *Discrete Math.*, 235(1):95–106, 2001. doi:10.1016/S0012-365X(00)00263-6.

17 Petr Hliněný and Jan Kratochvíl. Representing graphs by disks and balls (a survey of recognition-complexity results). *Discrete Math.*, 229(1–3):101–124, 2001. doi:10.1016/S0012-365X(00)00204-1.

18 Linda Kleist and Benjamin Rahman. Unit contact representations of grid subgraphs with regular polytopes in 2D and 3D. In Christian Duncan and Antonios Symvonis, editors, *Proc. Graph Drawing (GD'14)*, volume 8871 of *LNCS*, pages 137–148. Springer, 2014. doi:10.1007/978-3-662-45803-7_12.

19 Stephen G. Kobourov, Debajyoti Mondal, and Rahnuma Islam Nishat. Touching triangle representations for 3-connected planar graphs. In Walter Didimo and Maurizio Patrignani, editors, *Proc. Graph Drawing (GD'12)*, volume 7704 of *LNCS*, pages 199–210. Springer, 2013. doi:10.1007/978-3-642-36763-2_18.

20 Paul Koebe. Kontaktprobleme der konformen Abbildung. *Berichte über die Verhandlungen der Sächsischen Akad. der Wissen. zu Leipzig. Math.-Phys. Klasse*, 88:141–164, 1936.

21 Ernst Steinitz. Polyeder und Raumeinteilungen. In *Encyclopädie der mathematischen Wissenschaften*, volume 3-1-2 (Geometrie), chapter 12, pages 1–139. B. G. Teubner, Leipzig, 1922. URL: https://gdz.sub.uni-goettingen.de/id/PPN360609767?tify={"pages":[839],"view":"toc"}.

22 Szilassi polyhedron. Wikipedia entry. Accessed 2019-10-08. URL: https://en.wikipedia.org/wiki/Szilassi_polyhedron.

23 Paul Turán. Eine Extremalaufgabe aus der Graphentheorie. *Mat. Fiz. Lapok*, 48:436–452, 1941.

Headerless Routing in Unit Disk Graphs*

Wolfgang Mulzer[1] and Max Willert[1]

1 **Institut für Informatik, Freie Universität Berlin, 14195 Berlin, Germany**
{mulzer,willert}@inf.fu-berlin.de

Abstract

Let $V \subset \mathbb{R}^2$ be a set of n sites in the plane. The *unit disk graph* $\mathrm{DG}(V)$ of V is the graph with vertex set V in which two sites v and w are adjacent if and only if their Euclidean distance is at most 1.

We develop a *compact routing scheme* $\mathcal{R}$ for $\mathrm{DG}(V)$. The routing scheme $\mathcal{R}$ preprocesses $\mathrm{DG}(V)$ by assigning a *label* $\ell(v)$ to every site v in V. After that, for any two sites s and t, the scheme $\mathcal{R}$ must be able to route a packet from s to t as follows: given the label of a *current vertex* r (initially, $r = s$) and the label of the target vertex t, the scheme determines a neighbor r' of r. Then, the packet is forwarded to r', and the process continues until the packet reaches its desired target t. The resulting path between the source s and the target t is called the *routing path* of s and t. The *stretch* of the routing scheme is the maximum ratio of the total Euclidean length of the routing path and of the shortest path in $\mathrm{DG}(V)$, between any two sites $s, t \in V$.

We show that for any given $\varepsilon > 0$, we can construct a routing scheme for $\mathrm{DG}(V)$ with diameter D achieving stretch $1 + \varepsilon$ and label size $O(\log D \log^3 n / \log\log n)$ (the constant in the O-Notation depends on ε). In the past, several routing schemes for unit disk graphs have been proposed. Our scheme is the first one to achieve poly-logarithmic label size and arbitrarily small stretch without storing any additional information in the packet.

1 Introduction

The *routing problem* is a well-known problem in distributed graph algorithms [10, 13]. We are given a graph G and want to preprocces it by assigning *labels* to each node of G such that the following task can be solved: a data packet is located at a source node and has to be routed to a target node. A routing scheme should have several properties. First, routing must be *local*: a node can only use the label of the target node as well as its own local information to compute a neighbor to which the packet is sent next. Second, the routing should be *efficient*: the ratio of the routed path and the shortest path — the *stretch factor* — should be close to 1. Finally, the routing scheme should be *compact*: the size of the labels (in bits) must be small.

Many routing schemes use additional *headers*. The header contains mutable information and is stored in the data packet. Thus, the header moves with the data packet through the graph. The usage of an additional header makes it possible to implement recursive routing strategies or to remember information from past positions of the packet.

A trivial solution to solve the routing problem is to store the complete shortest path tree in every label. Then it is easy to route the data packets along a shortest path. However, such a routing scheme is not compact. Moreover, Peleg and Upfal [13] proved that in general graphs, any routing scheme that achieves a constant stretch factor must store a polynomial number of bits for each node.

Nevertheless, there is a rich collection of routing schemes for general graphs [1, 2, 5, 7, 8, 14, 15]. For example, the scheme by Roditty and Tov [15] uses labels of size $mn^{O(1/\sqrt{\log n})}$

* Partially supported by ERC STG 757609.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

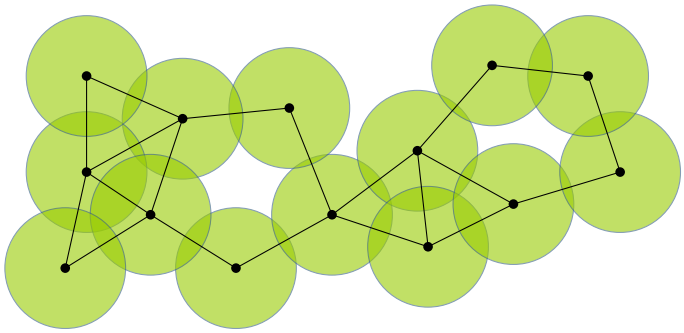


Figure 1 The disks in the unit disk graph have diameter 1 and there is an edge between two midpoints if and only if their corresponding disks intersect.

and routes a packet from s to t on a path of length $O\big(k\Delta + m^{1/k}\big)$, where Δ is the shortest path distance between s and t, $k > 2$ is any fixed integer, n is the number of nodes, and m is the number of edges. Their routing scheme needs headers of poly-logarithmic size. There are routing schemes for special graph classes that achieve better bounds on the label size and stretch factor [3, 9, 16–18].

Our graph class of interest comes from the study of mobile and wireless networks. These networks are usually modeled as *unit disk graphs* [6] with diameter D. There are already several routing schemes for unit disk graphs [11, 20]. We present the first headerless routing scheme with label size $O(\varepsilon^{-4} \log D \log^3 n / \log\log n)$ that achieves stretch $1 + \varepsilon$.

2 Preliminaries

Let $G = (V, E)$ be a *simple*, *undirected*, and *connected* graph with n vertices. In our model the graph G is embedded in the Euclidean plane and an edge uv is weighted according to the Euclidean distance $|uv|$ of its endpoints, for all , $uv \in V$. We write $d(u, v)$ for the (weighted) *shortest path distance* between the vertices $u, v \in V$. Moreover, every vertex v has a unique identifier $v_{\mathrm{id}} \in \{0, \ldots, n-1\}$. In a *unit disk graph* $\mathrm{DG}(V)$ of V, there is an edge between two nodes $u, v \in V$ if and only if $|uv| \leq 1$, see Figure 1. We use D to denote the diameter $\max_{u,v \in V} d(u, v)$ of $\mathrm{DG}(V)$. We assume that $\mathrm{DG}(V)$ is connected. Hence, $D \leq n - 1$.

Routing Schemes. Let $G = (V, E)$ be a graph. A *routing scheme* $\mathcal{R}$ for G consists of a *labeling function* $\ell(v) \in \{0, 1\}^+$. It serves as the address of the node v in G and might contain some more information about the topology of G. Furthermore, $\mathcal{R}$ has a *routing function* $\sigma : \ell(V) \times \ell(V) \to V$. The routing function σ describes the behavior of the routing scheme, as follows: assume a data packet is located at a vertex $s \in V$ and must be routed to a destination $t \in V$. Then, $\sigma(\ell(s), \ell(t))$ has to compute a vertex to which the data packet is forwarded. Now, let $v_0 = s$ and $v_{i+1} = \sigma(\ell(v_i), \ell(t))$, for $i \geq 0$. The sequence $(v_i)_{i \in \mathbb{N}}$ is called *routing sequence*. The routing scheme $\mathcal{R}$ is *correct*, if and only if for all distinct $s, t \in V$, there is a number $m \in \mathbb{N}$ such that $v_j = t$, for all $j \geq m$, and $v_j \neq t$, for all $j = 0, \ldots, m-1$. If $\mathcal{R}$ is correct for G, then $\delta(s, t) = \sum_{i=1}^{m} |v_{i-1} v_i|$ is called the *routing length* between s and t (in G). The *stretch* of the routing scheme is the largest ratio $\delta(s, t)/d(s, t)$ over all distinct vertices $s, t \in V$. The goal is to achieve a routing scheme that minimizes the stretch factor as well as the number of bits stored in the labels.

3 Building Blocks

The Distance Oracle of Chan and Skrepetos. Our routing scheme is based on the recent approximate distance oracle for unit disk graphs by Chan and Skrepetos [4]: we are given an n-vertex unit disk graph with diameter D and a parameter $\varepsilon \geq D^{-1}$. Chan and Skrepetos show how to compute an efficient data structure that can answer *approximate distance queries* in $\mathrm{DG}(V)$: given two vertices $s, t \in V$, compute a number $\theta \in \mathbb{R}$ with $d(s,t) \leq \theta \leq d(s,t) + O(\varepsilon D)$. The main tool for this data structure is a *decomposition tree* $\mathcal{T}$ for $\mathrm{DG}(V)$ with the following properties.

- Every node μ of $\mathcal{T}$ is assigned two sets $\mathrm{port}(\mu)$ and $V(\mu)$ such that $\mathrm{port}(\mu) \subseteq V(\mu) \subseteq V$. The subgraph of $\mathrm{DG}(V)$ induced by $V(\mu)$ is connected and the vertices in $\mathrm{port}(\mu)$ are called *portals*.
- If μ is the root, then $V(\mu) = V$. If μ is a leaf, then $V(\mu) = \mathrm{port}(\mu)$.
- If μ is an inner node with k children $\sigma_1, \ldots, \sigma_k$, the sets $\mathrm{port}(\mu), V(\sigma_1), \ldots, V(\sigma_k)$ are pairwise disjoint, and we have $V(\sigma_i) \subseteq V(\mu)$, for $1 \leq i \leq k$.
- The height of $\mathcal{T}$ is in $O(\log n)$, and for every node μ of $\mathcal{T}$, we have $|\mathrm{port}(\mu)| \in O(1/\varepsilon)$.

To state the final (and most important) property of $\mathcal{T}$, we need some additional notation. The properties of $\mathcal{T}$ so far imply that the portal sets of two different nodes in $\mathcal{T}$ are disjoint. For every portal p, we let $\mu(p)$ be the unique node in $\mathcal{T}$ with $p \in \mathrm{port}(\mu(p))$. Moreover, let μ be a node of $\mathcal{T}$ and $s, t \in V(\mu)$. We denote by $d_\mu(s,t)$ the shortest path distance between s and t in the subgraph of $\mathrm{DG}(V)$ induced by $V(\mu)$. Now, the decomposition tree of Chan and Skrepetos has the property that for every pair of vertices $s, t \in V$, if we set

$$\theta(s,t) = \min_{\substack{p \text{ portal} \\ s,t \in V(\mu(p))}} d_{\mu(p)}(s,p) + d_{\mu(p)}(p,t)$$

then

$$\theta(s,t) \leq d(s,t) + O(\varepsilon D). \tag{1}$$

Simple Routing Schemes. Moreover, we need the following known routing schemes. The first routing scheme is due to Fraigniaud and Gavoille [9] as well as Thorup and Zwick [18]. The second routing scheme is based on an idea described by Kaplan et al. [11]. For the proof, we refer to [12].

▶ **Lemma 1.** *Let T be an n-vertex tree with arbitrary edge weights. There is a routing scheme for T with label size $O(\log^2 n / \log\log n)$ and stretch 1.*

▶ **Lemma 2.** *Let $\mathrm{DG}(V)$ be an n-vertex unit disk graph with diameter D and $0 < \varepsilon \leq D^{-1}$. There is a routing scheme for $\mathrm{DG}(V)$ with label size $O\left(\varepsilon^{-4} \log n\right)$ and stretch $1 + O(\varepsilon)$.*

4 A Routing Scheme with Additive Stretch

In this section we present a routing scheme that is efficient for $\mathrm{DG}(V)$ if the diameter D is large. Let $\varepsilon > D^{-1}$ and $c = n \cdot (\varepsilon D)^{-1}$. We define $x_c = \lfloor x \cdot c \rfloor$, for each $x \in \mathbb{R}_0^+$. Let $\mathcal{T}$ be the decomposition tree of $\mathrm{DG}(V)$, as explained in Section 3.

The idea of the routing scheme is as follows: We use $\mathcal{T}$ to compute a set of shortest path trees whose union covers $\mathrm{DG}(V)$ such that every vertex is contained in at most $O(\varepsilon^{-1} \log n)$ trees. In each step of the routing process, we use the source and target labels to select a good shortest path tree and route in this tree.

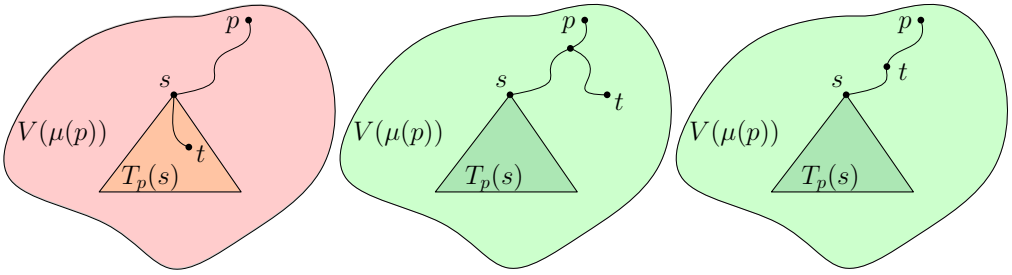


Figure 2 Left: If t is in $T_p(s)$, i.e., $\theta(s,t;p) = d_{\mu(p)}(t,p) - d_{\mu(p)}(p,s)$, we route away from p. Middle and Right: If t is not in $T_p(s)$, i.e., $\theta(s,t;p) = d_{\mu(p)}(t,p) + d_{\mu(p)}(p,s)$, we route towards p. The right picture suggests to define $\theta(s,t;p)$ as $d_{\mu(p)}(s,p) - d_{\mu(p)}(t,p)$. This does not influence the guarantees of our routing scheme but would lead to more cases.

The Labels. Let $v \in V$, and let p be a portal with $v \in V(\mu(p))$. We compute the shortest path tree T_p for p in $V(\mu(p))$ and enumerate its vertices in postorder. The postorder number of v in T_p is denoted by $r_p(v)$. Next, the subtree of T_p rooted at v is called $T_p(v)$ and we use $l_p(v)$ to denote the smallest postorder number in $T_p(v)$. Thus, a vertex $w \in V(\mu(p))$ is in the subtree $T_p(v)$ if and only if $r_p(w) \in [l_p(v), r_p(v)]$. Finally, we apply the tree routing from Lemma 1 to T_p and denote by $\ell_p(v)$ the corresponding label of v. We store $(p_{\mathrm{id}}, d_{\mu(p)}(v,p)_c, l_p(v), r_p(v), \ell_p(v))$ in $\ell(v)$, for every portal p. The rounding of the distances is necessary since we are not allowed to store real values. For the proof of the next lemma see [12].

▶ **Lemma 3.** *For every vertex $v \in V$, we have $|\ell(v)| \in O\left(\frac{\log^3 n}{\varepsilon \log\log n}\right)$.*

The Routing Function. We are given the labels $\ell(s)$ and $\ell(t)$ for the current vertex s and the target vertex t. First, we identify all portals p with $s,t \in V(\mu(p))$. We can do this by identifying all vertices p such that the entry $(p_{\mathrm{id}}, d_{\mu(p)}(s,p)_c, l_p(s), r_p(s), \ell_p(s))$ is in $\ell(s)$ and the entry $(p_{\mathrm{id}}, d_{\mu(p)}(t,p)_c, l_p(t), r_p(t), \ell_p(t))$ is in $\ell(t)$. Next, let $\theta(s,t;p) = d_{\mu(p)}(t,p) + d_{\mu(p)}(p,s)$, if t is not in the subtree $T_p(s)$, and $\theta(s,t;p) = d_{\mu(p)}(t,p) - d_{\mu(p)}(p,s)$, otherwise; see Figure 2 for an illustration of the two cases. Let p_{opt} be the portal that minimizes $\theta(s,t;p)$ among all portals p. Then, it is easy to see, that $\theta(s,t;p_{opt}) \leq \theta(s,t)$. Hence, $\theta(s,t;p_{opt})$ is a good approximation for the distance between s and t and we would like to route in $T_{p_{opt}}$. However, the routing function cannot compute the optimal portal p_{opt}, since we do not have direct access to the exact distances. Instead, we use the rounded distances to compute a near-optimal portal. We define $\theta_c(s,t;p) = d_{\mu(p)}(t,p)_c + d_{\mu(p)}(p,s)_c$, if t is not in the subtree $T_p(s)$, and $\theta_c(s,t;p) = d_{\mu(p)}(t,p)_c - d_{\mu(p)}(p,s)_c$, otherwise. Let p^* be the portal that lexicographically minimizes $(\theta_c(s,t;p), p_{\mathrm{id}})$, among all portals p. We call p^* the *s-t-portal* and set $\theta_c(s,t) = \theta_c(s,t;p^*)$. Observe that the s-t-portal can be computed by using only the labels of s and t. The routing function now uses the labels $\ell_{p^*}(s)$ and $\ell_{p^*}(t)$ to compute the next vertex in T_{p^*} and forwards the data packet to this vertex.

Analysis. The following lemma shows that we make progress after each step.

▶ **Lemma 4.** *Let s be the current vertex, t the target vertex, and suppose that the routing scheme sends the packet from s to v. Moreover, let p be the s-t-portal and q be the v-t-portal. We have*

1. $\theta_c(s,t) \geq \theta_c(v,t) + |sv|_c$,
2. *if* $\theta_c(s,t) = \theta_c(v,t)$ *then* $p_{\mathrm{id}} \geq q_{\mathrm{id}}$, *and*
3. *if* $\theta_c(s,t) = \theta_c(v,t)$ *and* $p_{\mathrm{id}} = q_{\mathrm{id}}$ *then* v *is on a shortest path from* s *to* t *in* T_q.

The intuition is as follows. First of all, the rounded approximate distance to the target will never increase (1st statement). If this values does not change, then the next tree in which we route can not have a larger index (2nd statement). If this index does not change, then we decrease the hop distance to our target (3rd statement). Hence, we made progress. The first statement of Lemma 4 can now be used to obtain the stretch factor and therefore the main theorem. The proof uses Inequality 1 and can be found in [12].

▶ **Theorem 5.** *Let* $\mathrm{DG}(V)$ *be an* n*-vertex unit disk graph with diameter* D *and* $\varepsilon > D^{-1}$. *There is a routing scheme for* $\mathrm{DG}(V)$ *with label size* $O\left(\varepsilon^{-1} \log^3 n / \log\log n\right)$ *and additive stretch* $O(\varepsilon D)$.

Theorem 5 and Lemma 2 can now be used as building blocks to get a routing scheme with stretch factor $1+\varepsilon$. To achieve this we use a well-known technique that groups the vertices of $\mathrm{DG}(V)$ using a hierarchy of sparse covers with exponentially increasing diameter [4]. In the end, each node is contained in at most $O(\log D)$ different groups. Each group gives a connected subgraph of $\mathrm{DG}(V)$ on which we apply one of the two routing schemes, depending on whether $\varepsilon \geq D^{-1}$ or not. It is then easy to route in these subgraphs. For the details and the analysis we refer to [12]. Nevertheless, we claim the following.

▶ **Theorem 6.** *Let* $\mathrm{DG}(V)$ *be an* n*-vertex unit disk graph with diameter* D *and* $\varepsilon > 0$. *There is a routing scheme for* $\mathrm{DG}(V)$ *with label size* $O\left(\varepsilon^{-4} \log D \log^3 n / \log\log n\right)$ *and stretch* $1+\varepsilon$.

5 Conclusion

We presented an efficient, compact, and headerless routing scheme for unit disk graphs. It achieves near-optimal stretch $1+\varepsilon$ and uses $O(\log D \log^3 n / \log\log n)$ bits in the label.

It would be interesting to see if this result can be extended to disk graphs in general. If the radii of the disks are unbounded, the decomposition of Chan and Skrepetos cannot be applied immediately. However, the case of bounded radii is still interesting, and even there, it is not clear how the method by Chan and Skrepetos generalizes.

Finally, let us compare our routing scheme to the known schemes. The model of the routing scheme of Kaplan et al. [11] is very close to ours. They claim that the neighborhood can be checked locally without wasting storage. We also use this assumption in the details of Lemma 2. The scheme was generalized to non-unit disk graphs with constant bounded radii [19]. Nevertheless, in unit disk graphs, we achieve the same stretch factor and still have additional information of poly-logarithmic size. The main advantage of our routing scheme is that we do not use any additional headers. Therefore, whenever a data packet arrives at a node, it is not necessary to know what happened before or where the packet came from. In the routing scheme of Kaplan et al., a data packet visits a node more than once.

The routing scheme of Yan et al. [20] uses headers as well, but they are only computed in the first step and do not change again. The idea of their routing scheme is similar to ours: the graph is covered by $O(\log n)$ different trees. When the routing starts, the labels of the source and the target are used to determine the identity of a tree and an $O(\log n)$-bit label of the target within this tree. Finally, they completely forget the original labels and route within this tree until they reach t. Their stretch is bounded by a constant. Our routing scheme can also be turned into this model, but we have $O(\log D \log n)$ different trees that cover the

unit disk graph and the label of a vertex in one of the trees has size $O(\log^2 n/\log\log n)$. Nevertheless, we achieve the near optimal stretch $1+\varepsilon$. A more thorough analysis of their and our model will lead to a more complicated comparison. This does not fit here, so we refer the reader to have a look into [12].

References

1 Ittai Abraham and Cyril Gavoille. On approximate distance labels and routing schemes with affine stretch. In *Proc. 25th Int. Symp. Dist. Comp. (DISC)*, pages 404–415, 2011.

2 Baruch Awerbuch, Amotz Bar-Noy, Nathan Linial, and David Peleg. Improved routing strategies with succinct tables. *J. Algorithms*, 11(3):307–341, 1990.

3 Bahareh Banyassady, Man-Kwun Chiu, Matias Korman, Wolfgang Mulzer, André van Renssen, Marcel Roeloffzen, Paul Seiferth, Yannik Stein, Birgit Vogtenhuber, and Max Willert. Routing in polygonal domains. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 92. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

4 Timothy M. Chan and Dimitrios Skrepetos. Approximate shortest paths and distance oracles in weighted unit-disk graphs. *J. of Computational Geometry*, 10(2):3–20, 2019. doi:10.20382/jocg.v10i2a2.

5 Shiri Chechik. Compact routing schemes with improved stretch. In *Proc. ACM Symp. Princ. Dist. Comp. (PODC)*, pages 33–41, 2013.

6 Brent N Clark, Charles J Colbourn, and David S Johnson. Unit disk graphs. *Discrete mathematics*, 86(1-3):165–177, 1990.

7 Lenore J Cowen. Compact routing with minimum stretch. *J. Algorithms*, 38(1):170–183, 2001.

8 Tamar Eilam, Cyril Gavoille, and David Peleg. Compact routing schemes with low stretch factor. *J. Algorithms*, 46(2):97–114, 2003.

9 Pierre Fraigniaud and Cyril Gavoille. Routing in trees. In *Proc. 28th Internat. Colloq. Automata Lang. Program. (ICALP)*, pages 757–772, 2001.

10 Silvia Giordano and Ivan Stojmenovic. Position based routing algorithms for ad hoc networks: A taxonomy. In *Ad hoc wireless networking*, pages 103–136. Springer-Verlag, 2004.

11 Haim Kaplan, Wolfgang Mulzer, Liam Roditty, and Paul Seiferth. Routing in unit disk graphs. *Algorithmica*, 80(3):830–848, 2018.

12 Wolfgang Mulzer and Max Willert. Routing in unit disk graphs without dynamic headers, 2020. arXiv:2002.10841.

13 David Peleg and Eli Upfal. A trade-off between space and efficiency for routing tables. *J. ACM*, 36(3):510–530, 1989.

14 Liam Roditty and Roei Tov. New routing techniques and their applications. In *Proc. ACM Symp. Princ. Dist. Comp. (PODC)*, pages 23–32, 2015.

15 Liam Roditty and Roei Tov. Close to linear space routing schemes. *Distributed Computing*, 29(1):65–74, 2016.

16 Nicola Santoro and Ramez Khatib. Labelling and implicit routing in networks. *The Computer Journal*, 28(1):5–8, 1985.

17 Mikkel Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *J. ACM*, 51(6):993–1024, 2004.

18 Mikkel Thorup and Uri Zwick. Compact routing schemes. In *Proc. 13th ACM Symp. Par. Algo. Arch. (SPAA)*, pages 1–10, 2001.

19 Max Willert. Routing schemes for disk graphs and polygons. Master's thesis, Freie Universität Berlin, 2016.

20 Chenyu Yan, Yang Xiang, and Feodor F Dragan. Compact and low delay routing labeling scheme for unit disk graphs. *Comput. Geom. Theory Appl.*, 45(7):305–325, 2012.

A $(1+\varepsilon)$-approximation for the minimum enclosing ball problem in $\mathbb{R}^d$ *

Sang-Sub Kim and Barbara Schwarzwald

Institute of Computer Science, University Bonn
schwarzwald@uni-bonn.de

Abstract

Given a set of points P in $\mathbb{R}^d$ for an arbitrary d, the 1-center problem or minimum enclosing ball problem (MEB) asks to find a ball B^* of minimum radius r^* which covers all of P. Kumar et. al. [5] and Bădoiu and Clarkson [1] simultaneously developed core-set based $(1+\varepsilon)$-approximation algorithms. While Kumar et al. achieve a slightly better theoretical runtime of $O(nd/\varepsilon + 1/\varepsilon^{4.5} \log 1/\varepsilon)$, Bădoiu and Clarkson have a stricter bound of $\lceil 2/\varepsilon \rceil$ on the size of their core-set, which strongly affects run-time constants.

We give a gradient-descent based algorithm running in time $O(nd/\varepsilon)$ based on a geometric observation that was used first for a 2-center streaming algorithm by Kim and Ahn [4]. Our approach can be extended to the k-center problem to obtain a $(1+\varepsilon)$-approximation in time $O(nd\,k\,2^{k/\varepsilon})$.

1 Introduction

Given a set of points $P \subset \mathbb{R}^d$, the *minimum enclosing ball problem* (MEB), also known as the 1-center problem, asks to find a ball B^* of minimum radius r^* containing all of P and is an important subproblem in clustering. While it can be solved in worst-case linear time for fixed d [6], the dependence on d is exponential and hence not practical for high dimensional real-world applications. However, Bădoiu et al. [3] presented a $(1+\varepsilon)$-approximation algorithm for arbitrary d running in time $O(nd/\varepsilon^2 + 1/\varepsilon^{10} \log 1/\varepsilon)$ using core-sets of size at most $1/\varepsilon^2$ independent of d. An ε-core-set is a subset $S \subset P$, such that a ball of radius $(1+\varepsilon)r^*$ around the center of a minimum enclosing ball of S covers P. Their algorithm can be extended to approximate the k-center problem, but the running time is then exponential in k and the size of the core-set; so having a tight bound on the size of the core-set is paramount. In fact, no polynomial time approximation scheme for the k-center problem in high dimensions can exist if P $\neq$ NP, see [7].

Kumar et al. [5] improved these results to finding ε-core-sets of size $O(1/\varepsilon)$ in time $O(nd/\varepsilon^2 + 1/\varepsilon^{4.5} \log 1/\varepsilon)$. Indepedently Bădoiu and Clarkson [1] achieved an algorithm with a similar running time of $O(nd/\varepsilon + 1/\varepsilon^5)$ while having a stricter bound of $\lceil 2/\varepsilon \rceil$ on the size of their core-set, which significantly affects run-time especially when extending to the k-center problem. Bădoiu and Clarkson [1] also gave a simple gradient-descent algorithm obtaining a $(1+\varepsilon)$-approximation in time $O(nd/\varepsilon^2)$ and later showed that a tight bound of $\lceil 1/\varepsilon \rceil$ on the size of ε-core-sets exists, see [2]. The gradient-descent algorithm has the advantage of not computing minimum enclosing balls for several subsets of P of size $O(1/\varepsilon)$ which improves the constants involved in the calculation of each step and simplifies implementation. Bădoiu and Clarkson [2] also performed runtime experiments on both the gradient-descent and the different core-set-based algorithms which results showed that the gradient-descent algorithm is competitive in reality, as it converges significantly faster than its theoretical bound suggests.

* This work has been supported by DFG grant Kl 655/19 as part of a DACH project.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

We combine the analysis of these core-set-based algorithms with the ideas of the gradient-descent algorithm and extend structural observations made by Kim and Ahn [4] for the Euclidean 2-center problem in a streaming model to obtain a new efficient gradient-descent algorithm that converges to a $(1+\varepsilon)$-approximation in time $O(nd/\varepsilon)$. It can be applied to the k-center problem in a similar fashion as the core-set based algorithms. Hence, it gives an alternative $(1+\varepsilon)$-approximation in time $O(nd\,k\,2^{k/\varepsilon})$ for that problem without the use of core-sets and, possibly, a faster algorithm on real-world data.

2 An Algorithm for the Euclidean 1-center problem

In this section we present an algorithm for the Euclidean 1-center problem for high dimensions and show the following theorem:

▶ **Theorem 2.1.** *Given a set $P \in \mathbb{R}^d$, one can compute a $(1+\varepsilon)$-approximation of the minimum enclosing ball in time $O(nd/\varepsilon)$ with the gradient-descent-algorithm* GradientMEB.

Let $P \subset \mathbb{R}^d$ for any $d \geq 2$ be a set of n points. Let $B(c,r)$ denote a ball of radius r centered at c and let $r(B)$ and $c(B)$, denote the radius and center of a ball B, respectively. We denote by pq the straight line segment between two points p and q and by $|pq|$ the length of pq. Finally, we denote the boundary of a closed set A by ∂A.

Let $B^* = B(c^*, r^*)$ be the optimal solution of the 1-center problem for a set P. The core idea of the algorithm is a follows. We will start with an arbitrary point p_1 from P as a starting center m_1. For any center m_i constructed, the radius r_i necessary to cover all of P with a ball $B(m_i, r_i)$ is defined by the farthest point in P from m_i. Therefore, in every subsequent step we pick that farthest point as p_{i+1} and construct a new center m_{i+1} on the line segment between p_{i+1} and m_i to reduce r_i. We will use a central structural property proven in Lemma 2.2 to show how to construct $m_i = m(p_{i+1}, m_i)$ in such a way that we can give a bound on its distance $|m_i c^*|$ to the optimal center c^* decreasing with every step i. The exact definition of $m(p_{i+1}, m_i)$ will hence be given after that Lemma.

Algorithm 1 GradientMEB

Input: Set of points $P \subset \mathbb{R}^d$.
Output: A center c such that $B(c, (1+\varepsilon)r^*)$ covers P

```
p_1 ← arbitrary point from P
m_1 ← p_1
bestRadius ← ∞
for i = 1 to ⌊2/ε⌋ do
    p_{i+1} ← farthest point from m_i in P
    if |m_i p_{i+1}| < bestRadius then
        bestCenter ← m_{i-1}
        bestRadius ← |m_{i-1} p_i|
    m_{i+1} ← m(p_{i+1}, m_i)
return bestCenter
```

We will start with the proof of the central structural property and the construction of $m(p_{i+1}, m_i)$ and then show that after at most $k = \lfloor 2/\varepsilon \rfloor$ such steps, $|m_k c^*| < \varepsilon r^*$ and hence $B(m_k, (1+\varepsilon)r^*)$ covers all of P.

Both together will proof the correctness of GradientMEB. As the algorithm runs for $\lfloor 2/\varepsilon \rfloor$ rounds, finding p_i each round takes $O(nd)$ and the computation of m_i takes $O(d)$, this will also proof Theorem 2.1.

▶ **Lemma 2.2.** *Given two d-dimensional balls B and B' with radii r and r' around the same center point c with $r > r'$ with $d \geq 2$. Let $p \in \partial B$ and $p' \in \partial B'$ with $|pp'| = l \geq r$. Let B'' be the d-dimensional ball centered around c that is tangential to pp'. We denote that tangential point with m and the distances $|p'm|$ with l_1 and $|pm|$ with l_2, so $l = l_1 + l_2$. Consider any line segment p_1p_2 with $|p_1p_2| > l$, $p_1 \in B'$ and $p_2 \in B$. Then any point m^* on p_1p_2 with $|p_1m^*| \geq l_1$ and $|p_2m^*| \geq l_2$ lies inside B''.*

Proof. For $d = 2$ we first show that $|p_1p_2|$ intersects B'' at all. It is clear that we can rotate and reflex pp' without changing B'' as long as its length l stays the same. Hence we can assume without loss of generality, that p', c and p_1 are collinear with $p_1 \in cp'$. Then p_2 must lie in $B \setminus B(p', l)$ which means p_1p_2 intersects B'' as illustrated in Figure 1.

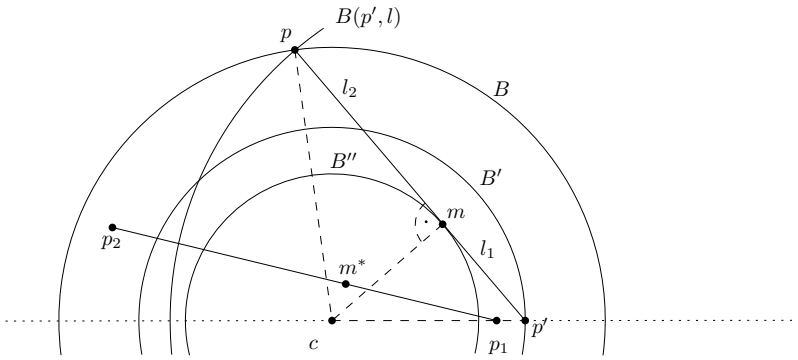


Figure 1 Construction of m and B''. p_1p_2 must intersect B'' if $|p_1p_2| \geq l = |pp'|$.

Now consider a point m^* on p_1p_2 with $|p_1m^*| \geq l_1$ and $|p_2m^*| \geq l_2$. Assume m^* is not contained in B''. Then either $p_1m^* \cap B'' = \emptyset$ and $p_2m^* \cap B'' \neq \emptyset$ or the other way around as p_1p_2 intersects B'' somewhere.

Let's first assume, $p_1m^* \cap B'' = \emptyset$ and $p_2m^* \cap B'' \neq \emptyset$. In this case $p_1 \in B' \setminus B''$. Let p_t be a point on the boundary of B'' such that p_1p_t is tangential to B'' and m^* lies within the triangle p_1p_tc. Clearly $|p_1p_t| > |p_1m^*|$. But by construction of B'', $|p'm| \geq |p_1p_t|$, which contradicts $|p'm| = l_1 \leq |p_1m^*|$. This is illustrated in Figure 2 (assuming without loss of generality that p' is collinear with p_1p_2, as this does not not affect the construction of B'').

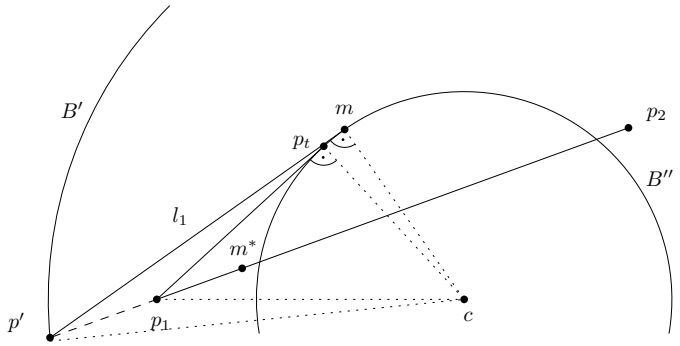


Figure 2 Assume $p_1m^* \cap B'' = \emptyset$. $|p_1m^*| < |p_1p_t| \leq |p'm|$. This contradicts $|p_1m^*| \geq l_1 = |p'm|$.

The other case can be shown equivalently by switching p_1 and p_2 and replacing p' and B' and l_1 with p and B and l_2, respectively.

For $d > 2$ we can show the lemma by choosing the 2-dimensional plane passing through c and the line segment p_1p_2 and then follow the same arguments as for $d = 2$. ◀

Note, that the point m' on p_1p_2 with $\frac{|m'p_1|}{|p_2p_1|} = \frac{l_1}{l} = \frac{|mp'|}{|pp'|}$ fulfils $|m'p_1| \geq l_1$ and $|m'p_2| \geq l_2$. The following corollary follows from that observation, Lemma 2.2 and the Pythagorean theorem and defines a way to calculate that point without actually knowing r^*.

▶ **Corollary 2.3.** *Let B and B' be two balls in $\mathbb{R}^d$ with $c(B) = c(B')$ and radii $r(B) = r^*$ and $r(B') = r' = \delta r^*$ for some $0 < \delta \leq 1$. Then the line segment pp' between any two points $p \in \partial B$ and $p' \in \partial B'$ with distance $|pp'| = l = (1+\varepsilon)r^*$ is tangential to $B'' = B(c, r_m)$ with*

$$r_m \leq r^* \sqrt{1 - \left(\frac{1 + (1+\varepsilon)^2 - \delta^2}{2(1+\varepsilon)}\right)^2} \tag{1}$$

at a point m^* with $l_1 \coloneqq |m^*p'|$.

Let $p_1 \in B'$ and $p_2 \in B$ with $|p_1p_2| \geq l = (1+\varepsilon)r^*$.

Then

$$m(p_1, p_2) \coloneqq p_1 + (p_2 - p_1)\frac{l_1}{l} = p_2 + (p_1 - p_2)\frac{\delta^2 + (1+\varepsilon)^2 - 1}{2(1+\varepsilon)^2} \tag{2}$$

lies in the ball $B(c, r_m)$ and can be calculated independent of r^*, only knowing p_1, p_2, δ and ε.

One can extend this definition to a sequence $m_1, m_2, \ldots, m_k$ based on a sequence of points $p_1, \ldots, p_k$ with $p_i \in P$ with $|p_j m_{j-1}| \geq (1+\varepsilon)r^*$ for all $i \geq j > 1$.

Let

$$\delta_i \coloneqq \begin{cases} 1, & \text{if } i = 1. \\ \sqrt{1 - \left(\frac{1+(1+\varepsilon)^2 - \delta_{i-1}^2}{2(1+\varepsilon)}\right)^2}, & \text{otherwise.} \end{cases} \tag{3}$$

and

$$m_i \coloneqq \begin{cases} p_1, & \text{if } i = 1. \\ m(p_i, m_{i-1}) = m_{i-1} + (p_i - m_{i-1})\frac{\delta_{i-1}^2 + (1+\varepsilon)^2 - 1}{2(1+\varepsilon)^2}, & \text{otherwise.} \end{cases} \tag{4}$$

As all points in P lie in the ball $B(c^*, r^*)$, it follows by induction from Corollary 2.3 that m_i lies in the ball $B(c^*, \delta_i r^*)$.

GradientMEB starts with an arbitrary point p_1 from P as m_i and always uses the farthest point from m_{i-1} in P as p_i. That way, at each round we either have $|m_i p_{i+1}| > (1+\varepsilon)r^*$ or m_i is already a $(1+\varepsilon)$-approximation. As we do not know, which of both holds at any round, we just return the best m_i out of all rounds.

It remains to prove, that any sequence of points with $|p_j m_{j-1}| \geq (1+\varepsilon)r^*$ for all $i \geq j > 1$ contains at most $k \leq \lfloor 2/\varepsilon \rfloor$ points before $m_i \in B(c^*, \varepsilon r^*)$.

If at step i, $|m_{i-1}p_i| = (1+\varepsilon)r^*$, $p_i \in \partial B^*$ and $m_{i-1} \in \partial B(c^*, \delta_{i-1}r^*)$, then $m_i \in \partial B(c^*, \delta_i r^*)$ by Lemma 2.2. In that case, $m_i \in B(c^*, \varepsilon r^*)$ if and only if $\delta_i < \varepsilon$. As this is the worst-case, we can assume we were given our sequence of points $p_i \in P$ by an adversary, always fulfilling $|m_{i-1}p_i| = (1+\varepsilon)r^*$ and $p_i \in \partial B^*$, which gives $m_{i-1} \in \partial B(c^*, \delta_{i-1})$ by induction.

We use a similar proof as [1] for their core-set based algorithm. For this we consider the line segments $a_i = m_{i-1}m_i$ with $|a_i| = \alpha_i(1+\varepsilon)r^*$ and $b_i = m_i p_i$ with $|b_i| = \beta_i(1+\varepsilon)r^*$ that together form $m_{i-1}p_i$ as illustrated in Figure 3.

As m_i converges towards c^* with increasing i, β_i increases and α_i decreases. However, β_i can be at most $1/(1+\varepsilon)$ by construction as b_i forms a right-angled triangle with c^*p_i as the hypotenuse, so $\beta_i(1+\varepsilon)r^* = |b_i| < |c^*p_i| = r^*$. We will now show a lower bound on β_i and prove that it exceeds $1/(1+\varepsilon)$ for $i \geq 2/\varepsilon - 1$. In that case, there does not exist a point p_i with $|m_{i-1}p_i| = (1+\varepsilon)r^*$ and $p_i \in \partial B^*$ that our adversary could have given us. This

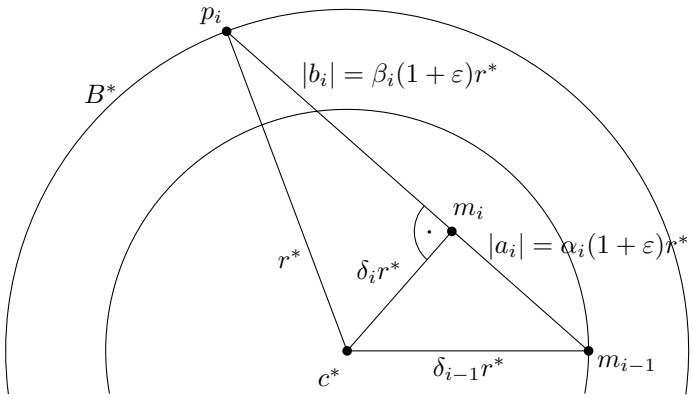


Figure 3 Construction of m_i based on m_{i-1} and p_i.

can only happen if the intersection of ∂B^* and $\partial B(m_{i-1}, (1+\varepsilon)r^*)$ is empty, and therefore, $\partial B(m_{i-1}, (1+\varepsilon)r^*)$ covers B^*.

▶ **Lemma 2.4.** *$\beta_i \geq 1/(1+\varepsilon)$ for $i \geq 2/\varepsilon - 1 > \lfloor 2/\varepsilon \rfloor$.*

Proof. By our definition and our worst-case assumption

$$|b_i| = (1+\varepsilon)r^* - |a_i| \qquad \Rightarrow \qquad \beta_i = 1 - \alpha_i. \tag{5}$$

In addition, by the construction of m_i as illustrated in Figure 3 and the Pythagorean theorem, it holds

$$\begin{aligned} \beta_i^2(1+\varepsilon)^2 &= 1^2 - \delta_i^2 \\ &= 1 - (\delta_{i-1}^2 - \alpha_i^2(1+\varepsilon)^2) \\ &= 1 - ((1-\beta_{i-1}^2(1+\varepsilon)^2) - \alpha_i^2(1+\varepsilon)^2) \\ \Rightarrow \qquad \beta_i^2 &= \alpha_i^2 + \beta_{i-1}^2. \end{aligned} \tag{6}$$

Combining these two equations we get

$$\begin{aligned} 1 - \alpha_i &= \sqrt{\beta_{i-1}^2 + \alpha_i^2} \\ 1 - 2\alpha_i + \alpha_i^2 &= \beta_{i-1}^2 + \alpha_i^2 \\ \Rightarrow \qquad \alpha_i &= \frac{1 - \beta_{i-1}^2}{2}. \end{aligned} \tag{7}$$

Applying Equation 5 again we obtain the recurrence

$$\beta_i = \frac{1 + \beta_{i-1}^2}{2}. \tag{8}$$

If we substitute $\gamma_i = \frac{1}{1-\beta_i} \Leftrightarrow \beta_i = \frac{\gamma_i - 1}{\gamma_i}$ in Equation 8, we get

$$\gamma_i = \frac{\gamma_{i-1}}{1 - 1/(2\gamma_{i-1})} = \gamma_{i-1}(1 + \frac{1}{2\gamma_{i-1}} + \frac{1}{4\gamma_{i-1}^2} + \cdots) \geq \gamma_{i-1} + \frac{1}{2}. \tag{9}$$

As we have $\beta_1 = 1/2$ and hence $\gamma_1 = 2$, we know $\gamma_i \geq (3+i)/2$ and hence $\beta_i \geq 1 - \frac{2}{3+i}$. To obtain $\beta_i \geq 1/(1+\varepsilon)$ it suffices to have $i \geq 2/\varepsilon - 1$. ◀

This also concludes the proof of Theorem 2.1.

2.1 Extension to the 2-center problem

We employ a strategy quite similar to the approach in [1]. We aim to construct two series of centers $m_{1,j}$ and $m_{2,k}$ based on two series of points from the two optimal balls B_1^* and B_2^*.

We start with an arbitrary point p_1 and set $m_{1,1} = p_1$ as we can assume $p_1 \in B_1^*$ without loss of generality. In every further step, we pick a point p_i farthest from the two current centers $m_{1,j}$ and $m_{2,k}$. As long as we have no center for B_2^*, we pick the point furthest from $m_{1,j}$. We then employ a guessing oracle that tells us whether p_i belongs to B_1^* or B_2^*. Depending on its answer, we add the point to the sequence for the respective ball then calculate a new center $m_{1,j+1}$ or $m_{2,k+1}$ as in our 1-center algorithm.

After at most $2\lfloor 2/\varepsilon \rfloor$ picks, we obtain a $(1+\varepsilon)$-approximation. As we do not have a guessing oracle, we just exhaust all possible guesses and return the best solution encountered, which results in a running time of $O(nd\,2^{1/\varepsilon})$.

2.2 Extension to the k-center problem for $k > 2$

The k-center algorithm is a straight-forwarded extension of the 2-center algorithm. As we need to guess at most $k\lfloor 2/\varepsilon \rfloor$ points to obtain $(1+\varepsilon)$-approximation and have to exhaust k possibilities each, our algorithm runs in time $O(nd\,k\,2^{k/\varepsilon})$.

3 Conclusion

We provided a new efficient gradient-descent $(1+\varepsilon)$ approximation algorithm for MEB in arbitrary dimensions running in time $O(nd/\varepsilon)$, which is strictly better than previous core-set based approaches with running times $O(nd/\varepsilon + 1/\varepsilon^{4.5} \log 1/\varepsilon)$ as long as $nd \in o(1/\varepsilon^{3.5} \log 1/\varepsilon)$. Like the core-set based algorithms it can be extended to the k-center problem with a running time of $O(nd\,k\,2^{k/\varepsilon})$, which makes the gradient-descent based algorithm theoretically equivalent to the core-set based approached with possibly better run-time constants by combining similar analysis with new geometric observations.

References

1 Mihai Bădoiu and Kenneth L. Clarkson. Smaller core-sets for balls. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '03, page 801–802, USA, 2003. Society for Industrial and Applied Mathematics. doi:10.5555/644108.644240.

2 Mihai Bădoiu and Kenneth L. Clarkson. Optimal core-sets for balls. *Computational Geometry*, 40(1):14 – 22, 2008. doi:10.1016/j.comgeo.2007.04.002.

3 Mihai Bădoiu, Sariel Har-Peled, and Piotr Indyk. Approximate clustering via core-sets. In *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*, STOC '02, page 250–257, New York, NY, USA, 2002. Association for Computing Machinery. doi:10.1145/509907.509947.

4 Sang-Sub Kim and Hee-Kap Ahn. An improved data stream algorithm for clustering. In Alberto Pardo and Alfredo Viola, editors, *LATIN 2014: Theoretical Informatics*, pages 273–284, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. doi:10.1007/978-3-642-54423-1_24.

5 Piyush Kumar, Joseph S. B. Mitchell, and E. Alper Yildirim. Approximate minimum enclosing balls in high dimensions using core-sets. *J. Exp. Algorithmics*, 8:1.1–es, December 2004. doi:10.1145/996546.996548.

6 Nimrod. Megiddo. "linear-time algorithms for linear programming in r^3 and related problems". *SIAM Journal on Computing*, 12(4):759–776, 1983. doi:10.1137/0212052.

7 Nimrod Megiddo. On the complexity of some geometric problems in unbounded dimension. *Journal of Symbolic Computation*, 10(3):327 – 334, 1990. doi:10.1016/S0747-7171(08)80067-3.

Disjoint tree-compatible plane perfect matchings*

Oswin Aichholzer[1], Julia Obmann[1], Pavel Paták[2], Daniel Perz[1], and Josef Tkadlec[2]

1 **Graz University of Technology, Graz, Austria**
oaich@ist.tugraz.at, julia.obmann@student.tugraz.at, daperz@ist.tugraz.at

2 **IST Austria, Klosterneuburg, Austria**
patak@kam.mff.cuni.cz, josef.tkadlec@ist.ac.at

Abstract

Two plane drawings of geometric graphs on the same set of points are called disjoint compatible if their union is plane and they do not have an edge in common. For a given set S of $2n$ points two plane drawings of perfect matchings M_1 and M_2 (which do not need to be disjoint nor compatible) are *disjoint tree-compatible* if there exists a plane drawing of a spanning tree T on S which is disjoint compatible to both M_1 and M_2.

We show that the graph of all disjoint tree-compatible perfect geometric matchings on $2n$ points in convex position is connected if and only if $2n \geq 10$. Moreover, in that case the diameter of this graph is either 4 or 5, independent of n.

1 Introduction

Two plane drawings of geometric graphs on the same set S of points are called *compatible* if their union is plane. The drawings are *disjoint compatible* if they are compatible and do not have an edge in common. For a fixed class $\mathcal{G}$, e.g. matchings, trees, etc., of plane geometric graphs on S the (disjoint) *compatibility graph* of S has the elements of $\mathcal{G}$ as the set of vertices and an edge between two elements of $\mathcal{G}$ if the two graphs are (disjoint) compatible. For example, it is well known that the (not necessarily disjoint) compatibility graph of plane perfect matchings is connected [4, 5]. Moreover, in [2] it is shown that there always exists a sequence of at most $O(\log n)$ compatible (but not necessarily disjoint) matchings between any two plane perfect matchings of a set of $2n$ points in general position, that is, the graph of perfect matchings is connected with diameter $O(\log n)$. On the other hand, Razen [8] provides an example of a point set where this diameter is $\Omega(\log n / \log \log n)$.

Disjoint compatible (perfect) matchings have been investigated in [2] for sets of $2n$ points in general position. The authors show that for odd n there exist isolated matchings and pose the following conjecture: For every perfect matching with an even number of edges there exists a disjoint compatible perfect matching. This conjecture was answered in the positive by Ishaque et al. [7] and it was mentioned that for even n it remains an open problem

* Research on this work was initiated at the 6th Austrian-Japanese-Mexican-Spanish Workshop on Discrete Geometry and continued during the 16th European Geometric Graph-Week, both held near Strobl, Austria. We are grateful to the participants for the inspiring atmosphere. We especially thank Alexander Pilz for bringing this class of problems to our attention and Birgit Vogtenhuber for inspiring discussions. D.P. is partially supported by the FWF grant I 3340-N35 (Collaborative DACH project *Arrangements and Drawings*). The research stay of P.P. at IST Austria is funded by the project CZ.02.2.69/0.0/0.0/17_050/0008466 Improvement of internationalization in the field of research and development at Charles University, through the support of quality projects MSCA-IF.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 734922.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

whether the disjoint compatibility graph is always connected. In [1] it is shown that for sets of $2n \geq 6$ points in convex position this disjoint compatibility graph is (always) disconnected.

Both concepts, compatibility and disjointness, are also used in combination with different geometric graphs. For example, in [5] it is shown that the flip-graph of all triangulations that admit a (compatible) perfect matching, is connected. It has also been shown that for every graph with an outerplanar embedding there exists a compatible plane perfect matching [3]. Considering plane trees and simple polygons, the same work provides bounds on the minimum number of edges a compatible plane perfect matching must have in common with the given graph. See also the survey [6] on the related concept of compatible graph augmentation.

In a similar spirit we can define a bipartite disjoint compatible graph, where the two sides of the bipartition represent two different graph classes. For example, let one side be all plane perfect matchings of S, $|S| = 2n$, while the other side consists of all plane spanning trees of S. Edges represent the pairs of matchings and trees whose union results in a plane, edge-disjoint drawing. Considering connectivity of this bipartite graph there trivially exist isolated vertices on the tree side - consider a spanning star, which can not have any disjoint compatible matching. Thus, the question remains whether there exists a bipartite connected subgraph which contains all vertices representing plane perfect matchings.

This point of view leads us to a new notion of adjacency for matchings. For a given set S of $2n$ points two plane drawings of perfect matchings M_1 and M_2 (which do not need to be disjoint nor compatible) are *disjoint tree-compatible* if there exists a plane drawing of a spanning tree T on S which is disjoint compatible to both, M_1 and M_2. The idea is that in the disjoint tree-compatible graph G_{2n} we have an edge between M_1 and M_2 as they are only two steps apart if we consider a disjoint and compatible transformation via T. Rephrasing the above question we ask whether G_{2n} is connected? Recall that the disjoint compatible graph for matchings alone is not connected (see [1, 2]) and that without disjointness the result of [5] already implies that the tree-compatible graph of matchings is connected.

In this paper we show that the disjoint tree-compatible graph of perfect geometric matchings on $2n$ points in convex position is connected if and only if $2n \geq 10$. Moreover, in that case the diameter of this graph is either 4 or 5, independent of n.

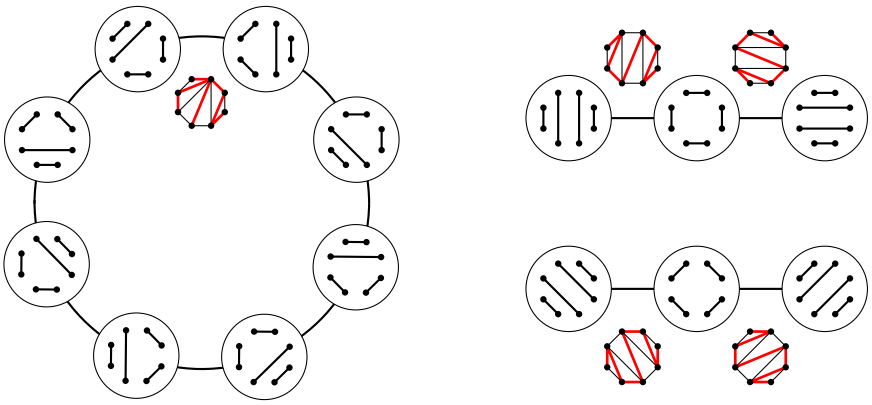


Figure 1 The disjoint tree-compatible graph G_8. Each vertex represents a plane matching on a convex point set of eight points. Two matchings are connected by an edge if they are tree-compatible (the trees are drawn in red).

1.1 Basic Definitions

Throughout this paper the point set S consists of $2n$ points in convex position. For simplicity we use the terms *matching* and *tree* for plane perfect matchings and plane spanning trees.

▶ **Definition 1.1.** Edges spanned by two neighbouring points on the boundary of the convex hull of S are called *perimeter edges*; all other edges spanned by S are called *diagonals*.

▶ **Definition 1.2.** A *perimeter matching* is a matching containing no diagonal. We label the sides of the convex hull of S alternately 'odd' and 'even'. Then the perimeter matching consisting of only odd perimeter edges is called *odd perimeter matching*, the one consisting of only even perimeter edges is called *even perimeter matching*.

2 Upper bound

We show that any matching on $2n \geq 10$ points in convex position has small distance to one of the two perimeter matchings which are themselves close to each other in the disjoint tree-compatible graph G_{2n}. When done carefully, this gives an upper bound of 5 on the diameter of G_{2n}.

First we introduce key notions of a semicycle, a cycle of edges and a rotation (see Figure 2).

▶ **Definition 2.1.** Let M be a matching on S. A set X of $k \geq 2$ matching edges is called a *k-semicycle* if the interior of the convex hull of X does not intersect any edges of M. Given a k-semicycle X, the perimeter of its convex hull (including the non-matching edges) is called its *k-cycle* and denoted by $\bar{X}$. A k-cycle $\bar{X}$ is called an *inside k-cycle* (or just an *inside cycle*) if $\bar{X}$ contains at least two diagonals, otherwise it is called a *k-ear* (or just an *ear*). Finally, given a semicycle X, we can obtain a matching $M' = \mathrm{rot}(\bar{X})$ by *rotating* the cycle $\bar{X}$, that is, by omitting from M edges in X and including edges in $\bar{X} \setminus X$.

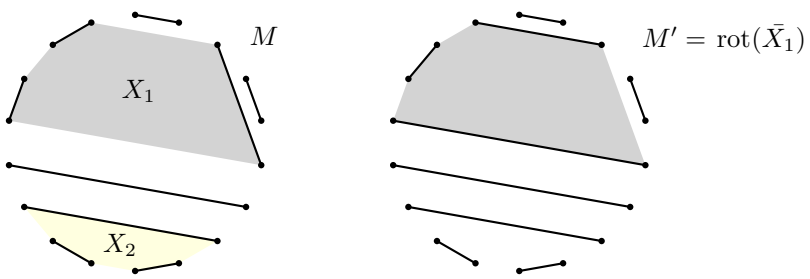


Figure 2 A matching M with convex hulls of two of its 3-semicycles X_1, X_2 shaded. The cycle $\bar{X}_1$ corresponding to X_1 is an inside cycle, since the boundary of the grey region contains at least two (in fact three) diagonals. The cycle $\bar{X}_2$ is an ear. Rotating $\bar{X}_1$, we obtain a matching $M' = \mathrm{rot}(\bar{X}_1)$.

With this notation in place we show that any number of inside cycles can be simultaneously rotated in one step and that sufficiently long ears can be rotated in at most 3 steps.

▶ **Lemma 2.2.** *Let M, M' be two matchings whose symmetric difference is a union of disjoint inside cycles. Then M and M' are tree-compatible to each other.*

Proof idea. The idea is that the union of M and M' can be extended to a triangulation such that every inside cycle has at most two vertices of S which are neighbored in this triangulation to only two other points of the inside cycle. We argue that the added edges form a connected graph spanning all the nodes of S. Removing edges one by one we create a spanning tree edge-disjoint to both matchings (see Figure 3). See full version for a full proof.

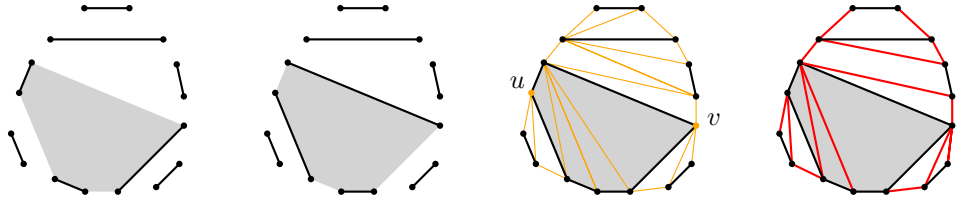


Figure 3 The union of two matchings differing at a single inside cycle (shaded) is extended to a suitable triangulation using yellow edges such that the yellow edges contain a spanning tree (red).

▶ **Lemma 2.3.** *Let M, M' be two matchings whose symmetric difference is a k-ear with $k \geq 6$. Then M and M' have distance at most 3 (in G_{2n}).*

Proof idea. The idea is to do three rotations as in Figure 4. See full version for a full proof.

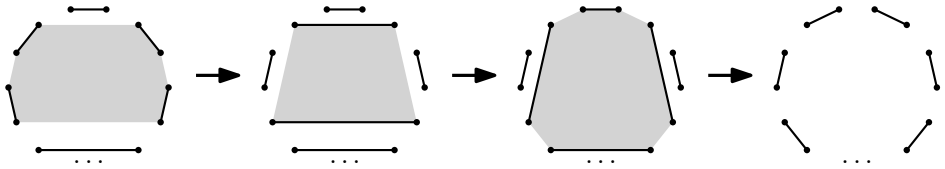


Figure 4 A 6-ear can be rotated in 3 steps (in each step we rotate the grey inside cycle).

▶ **Theorem 2.4.** *For $2n \geq 10$, the graph G_{2n} is connected and* $\mathrm{diam}(G_{2n}) \leq 5$.

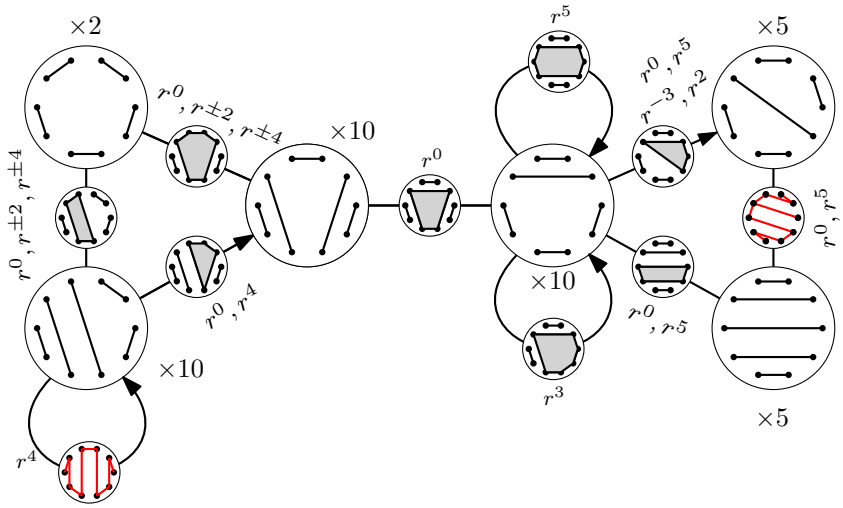


Figure 5 All 42 nodes of G_{10}, the letter r stands for a possible rotation by $2\pi/10$. Going against the arrows rotates in opposite direction.

Proof idea. When $2n = 10$, the claim can be checked in Figure 5. When $2n \geq 12$, the idea is to show that all matchings can be quickly transformed either to the odd perimeter matching O or to the even perimeter matching E (or to both – by Lemma 2.3 we have $\mathrm{dist}(O, E) \leq 3$). In particular, for a fixed matching M we denote by $\mathrm{d}_{\min}(M)$ (resp. $\mathrm{d}_{\max}(M)$) the distance from M to the closer (resp. further) perimeter matching. Then we prove that the non-perimeter matchings can be split into three classes S_1, S_2, S_3 with the following properties:

1. $\forall M \in S_1$ we have $\mathrm{d}_{\min}(M) \leq 1$ (and hence $\mathrm{d}_{\max}(M) \leq 1+3=4$);
2. $\forall M \in S_2$ we have $\mathrm{d}_{\min}(M) \leq 2$ and $\mathrm{d}_{\max}(M) \leq 3$;
3. $\forall M \in S_3$ we have $\mathrm{d}_{\max}(M) \leq 3$ and $\forall M, M' \in S_3$ we have $\mathrm{dist}(M, M') \leq 4$.

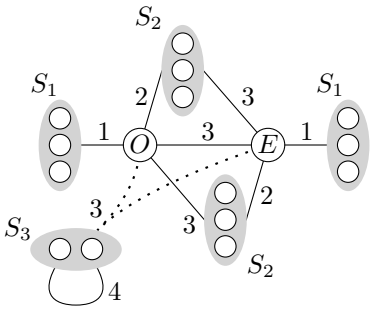


Figure 6 A partitioning of the non-perimeter matchings into sets S_1, S_2, S_3.

This guarantees that $\mathrm{diam}(G_{2n}) \leq 5$. See full version for a full proof.

3 Lower bound

In this chapter we prove the following theorem by constructing two matchings with distance at least 4.

▶ **Theorem 3.1.** *The diameter of the disjoint tree-compatible graph G_{2n} for $2n \geq 10$ can be lower bounded by 4.*

In the same way as we defined inside cycles and ears for cycles, we now introduce notions of inside semicycles and semiears for semicycles.

▶ **Definition 3.2.** Let M be a matching on S. A k-semicycle X is called an *inside k-semicycle* (or just an *inside semicycle*) if $\bar{X}$ contains at least two diagonals, otherwise it is called a *k-semiear* (or just a *semiear*).

▶ **Definition 3.3.** Let M and M' be two matchings in S. A *boundary area with k points* is an area within the convex hull of S restricted by edges in M and M' such that the matching edges intersect at least once and the points on the boundary of the area are adjacent on the boundary of the convex hull of S; see Figure 7.

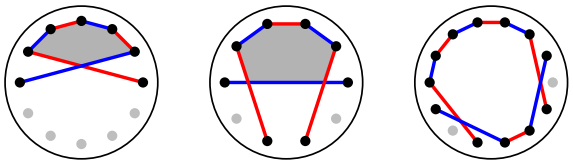


Figure 7 Boundary areas with five points (left) and four points (middle). The drawing on the right does not show a boundary area; not all points are neighbouring on the convex hull of S.

▶ **Definition 3.4.** A matching M on a set of $4k$ points is called a *2-semiear matching* if it consists of exactly k 2-semiears and an inside k-semicycle. A matching M on a set of $4k+2$ points is called a *near-2-semiear matching* if it consists of exactly k 2-semiears and an inside $(k+1)$-semicycle.

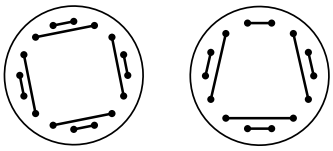


Figure 8 Left: A 2-semiear matching. Right: A near-2-semiear matching.

▶ Remark. Analogous to perimeter matchings we can distinguish between odd and even 2-semiear matchings, according to the values taken by the respective perimeter edges.

▶ **Lemma 3.5.** *Let M, M' be two matchings whose symmetric difference is an ear or a boundary area with at least three points. Then M and M' are not tree-compatible to each other.*

Proof idea. The idea is to apply a counting argument, a detailed proof is in the full version.

▶ **Lemma 3.6.** *Let M be a matching tree-compatible to an even 2-semiear-matching. Then M contains no odd perimeter edge.*

Proof idea. Adding an odd perimeter edge always yields either an ear or a boundary area with at least three points (cf. Figure 9), a detailed proof is in the full version.

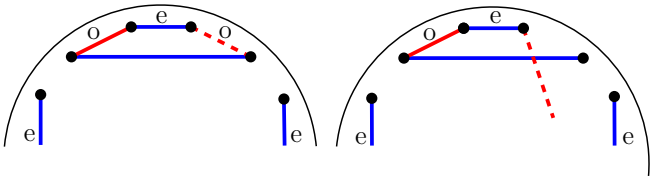


Figure 9 An (even) 2-semiear matching (in blue) and a (red) matching with at least one odd perimeter edge; the matchings create an ear (left) or a boundary area with three points (right).

▶ **Lemma 3.7.** *Let M be a matching tree-compatible to a near-2-semiear-matching M' consisting of k even and one odd perimeter edge. Then M contains at most one odd perimeter edge (the one in M').*

The proof works analogously to the proof of Lemma 3.6.

▶ **Lemma 3.8.** *Let M and M' be two tree-compatible matchings. Then M and M' have at least two perimeter edges in common.*

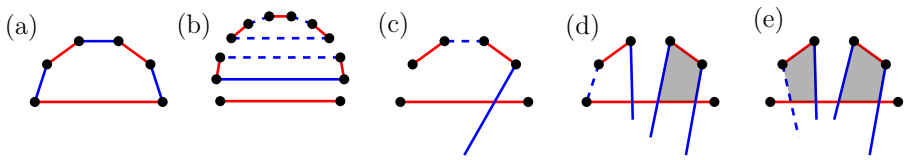


Figure 10 All possible cases for a semiear of size $k \geq 3$ in a matching M (depicted in red) and a second matching M' (depicted in blue) which does not use any of the perimeter edges in M.

Proof idea. We focus on proving that M and M' have one perimeter edge in common. Since we only use local arguments we can extend these to show that M and M' have at least two perimeter edges in common.

If M contains an ear of size at least three, then one of the perimeter edges of this ear is also in M'. Otherwise the union of the two matchings would give something like in Figure 10 which forbids a disjoint spanning tree.

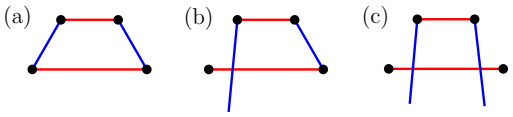


Figure 11 All possible cases for a 2-ear in a matching M (depicted in red) and a second matching M' (depicted in blue) which does not use the perimeter edges in M

So we can assume that M only has 2-semiears. If M' contains the perimeter edge of a 2-semiear of M, then we are done. So assume this is not the case. If we have a union of M and M', which looks locally like Figure 11(a) or Figure 11(b), then M and M' are not disjoint tree-compatible. So the only possibility that M and M' are disjoint tree-compatible and do not share a perimeter edge of a 2-semiear is depicted in Figure 11(c). Out of the 2-semiears of M we choose the one with no further semiear of M on one side of a diagonal d in M'. This is possible since the number of semiears is finite and the diagonals in M' cannot intersect each other, therefore there is an ordering of the 2-semiears in M. Since d is a diagonal, there exists a semiear E' on this side of the drawing in M'. Every edge of M on this side of d is a perimeter edge or intersects d, since there does not exist an semiear to this side of d in M. If E' is a 2-semiear and one diagonal in M intersects d, we get another blocking structure. This means that the perimeter edge of E' is also in M.

▶ **Corollary 3.9.** *Let S be of size $2n \geq 10$. For even n, the distance between an even 2-semiear matching and an odd 2-semiear matching is at least 4.*
For odd n, let M be a near-2-semiear matching with a single odd perimeter edge and M' be a near-2-semiear matching with a single even perimeter edge such that those two edges are incident in S. Then the distance between M and M' is at least 4.

Proof idea. We obtain the statement by applying Lemma 3.6 (for n even) or 3.7 (for n odd), respectively, and Lemma 3.8, cf. Figure 12. A detailed proof is in the full version.

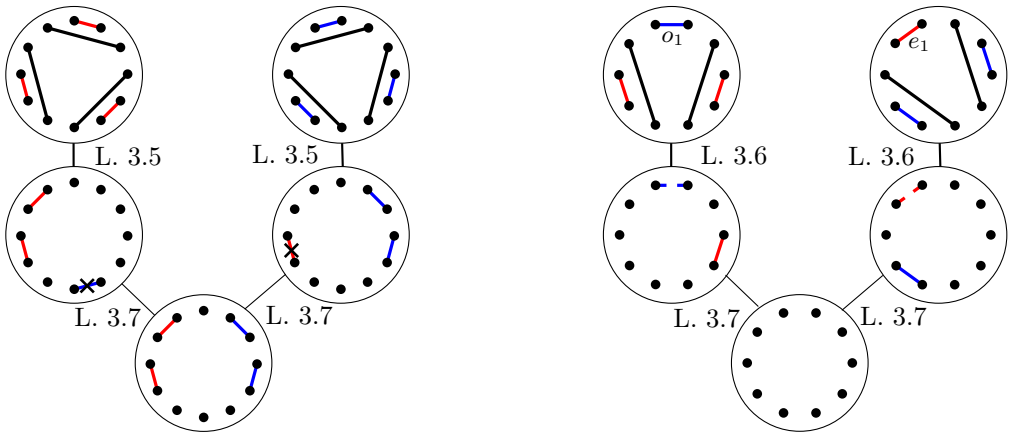


Figure 12 Illustrations, that the distance between two special 2-semiear matchings (left) and between two special near-2-semiear matchings (right) is at least 4. Even perimeter edges are drawn in red, odd ones are drawn in blue. The numbers next to the edges indicate which Lemma is applied.

4 Conclusion

We have shown that the diameter of the disjoint tree-compatible graph G_{2n} of disjoint tree-compatible matchings for points in convex position is 4 or 5 when $2n \geq 10$. Due to further computations, we conjecture that the diameter for all $2n \geq 18$ is 4. Further we obtained some results for the clique number of G_{2n}. Still, the question whether G_{2n} is connected for general point sets is open.

References

1 Oswin Aichholzer, Andrei Asinowski, and Tillmann Miltzow. Disjoint compatibility graph of non-crossing matchings of points in convex position. *The Electronic Journal of Combinatorics*, 22:1–65, 2015. URL: `http://www.combinatorics.org/ojs/index.php/eljc/article/view/v22i1p65`.

2 Oswin Aichholzer, Sergey Bereg, Adrian Dumitrescu, Alfredo García, Clemens Huemer, Ferran Hurtado, Mikio Kano, Alberto Márquez, David Rappaport, Shakhar Smorodinsky, Diane L Souvaine, Jorge Urrutia, and David Wood. Compatible Geometric Matchings. *Computational Geometry: Theory and Applications*, 42(6-7):617–626, 2009.

3 Oswin Aichholzer, Alfredo García, Ferran Hurtado, and Javier Tejel. Compatible matchings in geometric graphs. In *Proc. XIV Encuentros de Geometría Computacional*, pages 145–148, Alcalá, Spain, 2011.

4 Carmen Hernando, Ferran Hurtado, and Marc Noy. Graphs of non-crossing perfect matchings. *Graphs and Combinatorics*, 18(3):517–532, 2002.

5 Michael E Houle, Ferran Hurtado, Marc Noy, and Eduardo Rivera-Campo. Graphs of triangulations and perfect matchings. *Graphs and Combinatorics*, 21(3):325–331, 2005.

6 Ferran Hurtado and Csaba D Tóth. Plane geometric graph augmentation: a generic perspective. In *Thirty Essays on Geometric Graph Theory*, pages 327–354. Springer, 2013.

7 Mashhood Ishaque, Diane L Souvaine, and Csaba D Tóth. Disjoint compatible geometric matchings. *Discrete & Computational Geometry*, 49(1):89–131, 2013.

8 Andreas Razen. A lower bound for the transformation of compatible perfect matchings. *Proceedings of EuroCG*, pages 115–118, 2008.

Minimum Convex Partition of Degenerate Point Sets is NP-Hard*

Nicolas Grelier[1]

1 **Department of Computer Science, ETH Zürich**
nicolas.grelier@inf.ethz.ch

Abstract

Given a point set P and a natural integer k, are k closed convex polygons sufficient to partition the convex hull of P such that each polygon does not contain a point in P? What if the vertices of these polygons are constrained to be points of P? By allowing degenerate point sets, where three points may be on a line, we show that the first decision problem is NP-hard and the second NP-complete.

1 Introduction

The CG Challenge 2020 organised by Demaine, Fekete, Keldenich, Krupke and Mitchell [2], is about finding good solutions to the problem of *Minimum Convex Partition* (MCP). We give a definition equivalent to theirs, which fits better for the purpose of this paper.

▶ **Definition 1** (Minimum Convex Partition problem)**.** Given a set P of points in the plane and a natural number k, is it possible to find at most k closed convex polygons whose vertices are points of P, with the following properties:

- The union of the polygons is the convex hull of P,
- The interiors of the polygons are pairwise disjoint,
- No polygon contains a point of P in its interior.

The organisers of the CG Challenge 2020 mention that the complexity of this problem is unknown. Some partial results are known, under the additional assumption that no three points are on a line. For some more constrained point sets, Fevens, Meijer and Rappaport gave a polynomial time algorithm [3]. Keeping only the assumption that no three points are collinear, Knauer and Spillner have shown a $\frac{30}{11}$-approximation algorithm [6]. They also ask for the complexity of the Minimum Convex Partition problem. On a related note, Sakai and Urrutia have shown that for every set of n points, there exists a convex partition with at most $\frac{4}{3}n - 2$ polygons [8]. Although they do not mention it, it is straightforward to combine their result with the method of Knauer and Spillner to obtain a $\frac{8}{3}$-approximation algorithm. Concerning lower bounds, García-Lopez and Nicolás have given a construction for point sets for which any convex partition has at least $\frac{35}{32}n - \frac{3}{2}$ polygons [4]. An integer linear programming formulation of the problem, along with experimental results, has been recently introduced by Barboza, Souza and Rezende [1].

All those results, concerning algorithms and bounds, are shown for point sets in general position. However this is not assumed in the CG Challenge 2020. In this paper, we show that Minimum Convex Partition of degenerate point sets is NP-complete by a reduction from a modified version of planar 3-SAT. The complexity of the problem for point sets in general position is still open.

* Research supported by the Swiss National Science Foundation within the collaborative DACH project *Arrangements and Drawings* as SNSF Project 200021E-171681.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

We also show the NP-hardness of a similar problem, which we call *Minimum Convex Tiling* problem (MCT). The problem is exactly as in Definition 1, but the constraint about the vertices of the polygons is removed (i.e. they need not be points of P). This can make a difference as shown in Figure 1. Equivalently, the MCT problem corresponds to the MCP problem when Steiner points are allowed. A *Steiner point* is a point that does not belong to the point set given as input, and which can be used as a vertex of some polygons. Our proofs are very similar for the two problems. Due to lack of space, some parts of the NP-hardness proof of MCT, and how to adapt it for MCP, are postponed in the Appendix.

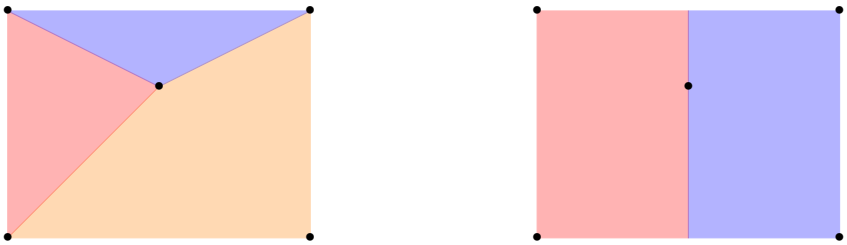


Figure 1 A minimum partition with three convex polygons, and a tiling with two.

Our proof builds upon gadgets introduced by Lingas [7]. He used them to prove NP-hardness of two decision problems: *Minimum Rectangular Partition for rectangles with point holes* and *Minimum Convex Partition for polygons with polygon holes*. In the second problem, Steiner points are allowed. However, as noted by Keil [5], one can easily adapt Lingas' proof not to use Steiner points. That is what we do in a second part to prove NP-hardness of the MCP problem. The two proofs of Lingas are similar, and consist in a reduction from the following variation of planar 3-SAT. The instances are a CNF formula F with set of variables X and set of clauses C, and a planar bipartite graph $G = (X \cup C, E)$, such that there is an edge between a variable $x \in X$ and a clause $c \in C$ if and only if x or $\bar{x}$ is a literal of c. Moreover, each clause contains either two literals or three, and if it contains three the clause must contain at least one positive and one negative literal. Lingas refers to this decision problem as the *Modified Planar* 3*-SAT* (MPLSAT). Lingas states that this problem is NP-complete, and we provide a proof of it in the Appendix. The main result of this paper is as follows:

▶ **Theorem 2.** *MPLSAT can be reduced in polynomial time to MCP, and to MCT.*

As it is easy to see that MCP is in NP, Theorem 2 implies that MCP is NP-complete. The question whether MCT is in NP is still open.

2 Construction of the point set

We do the reduction by constructing a point set in three steps. First we construct a non-simple polygon, in a very similar way as in Lingas' proof, with some more constraints. Secondly, we add some line segments to build a grid around the polygon, and finally we discretise all line segments into sets of evenly spaced collinear points. The idea of the first part is to mimic Lingas' proof. The second part makes the correctness proof easier, and the last part transforms the construction into our setting. The aim of the grid is to force the sets in a minimum convex tiling to be rectangular.

We use the gadgets introduced by Lingas, namely cranked wires and junctions [7]. A wire is shown in Figure 2. It consists of a loop delimited by two polygons, one inside the other.

In Lingas' construction, the two polygons are simple, and a wire is therefore a polygon with one hole. Moreover in his proof the dimensions of the cranks do not matter. In our case, the polygon inside is not simple, and each line segment has unit length. Each wire is bent several times with an angle of 90°, as shown in Figure 2, in order to close the loop.

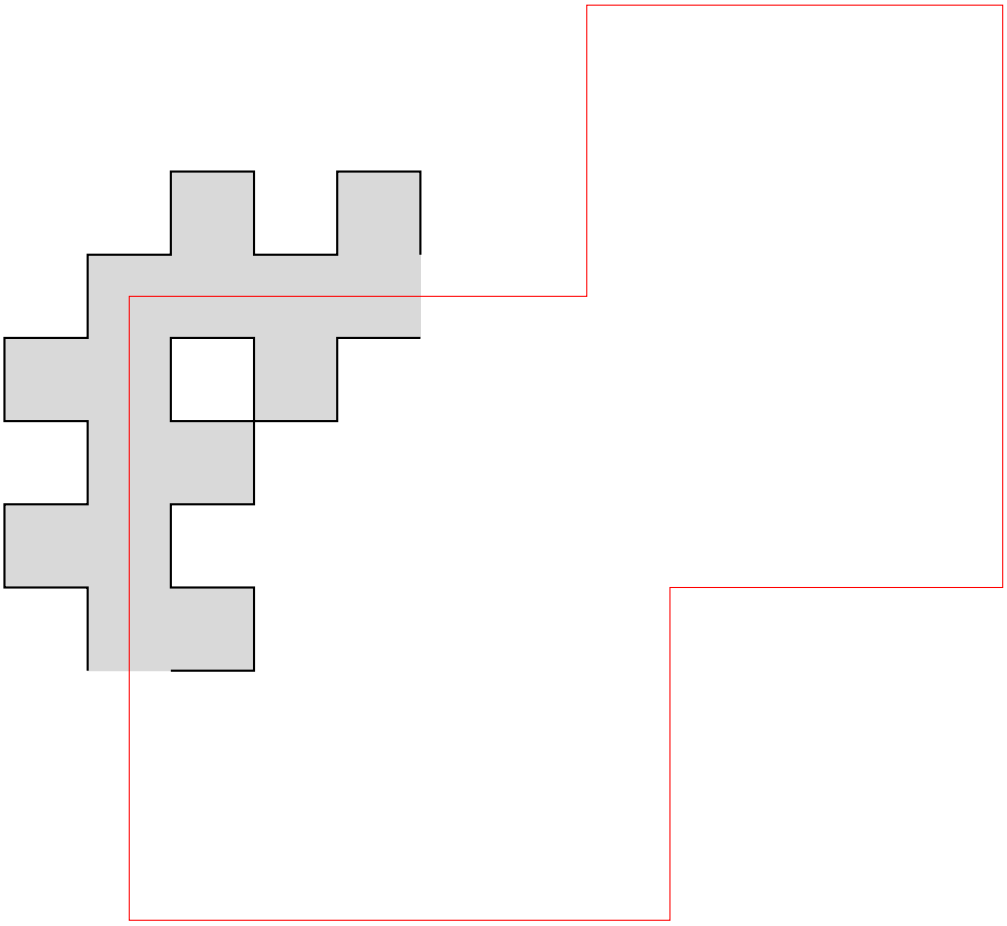


Figure 2 A cranked wire, edges are in black and its interior is in grey. The wire follows the whole red loop, but for sake of simplicity, only a section of the wire at a bend is drawn.

The wires are used to encode the values of the variables, with one wire for each variable. We are interested in two possible tilings of a wire, called vertical and horizontal, which are shown in Figure 3.

As in Lingas' proof, we interpret the vertical tiling as setting the variable to *true*, and the horizontal as *false*. Lingas proved the following:

▶ **Lemma 3** (Lingas [7]). *A minimum tiling with convex sets of a wire uses either vertical or horizontal rectangles but not both. Any other tiling requires at least one more convex set.*

The second tool is called a junction, and it serves to model a clause. Figure 4 depicts a junction corresponding to a clause of three literals. A junction has three arms, represented as dashed black line segments. A junction for a clause of two literals is obtained by blocking one of the arms of the junction. The blue line segments have length $1 + \varepsilon$, for a fixed ε arbitrarily small. Therefore, the red line segments are not aligned with the long black line segment to the left of the junction. A junction can be in four different orientations, which

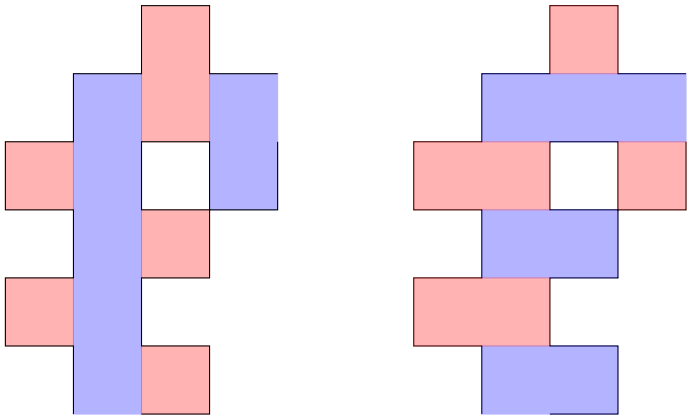


Figure 3 A section of a wire and its optimal tilings: vertical (left) and horizontal (right).

can be obtained successively by making rotations of 90°. Let us consider the orientation of the junction in Figure 4. One wire is connected from above, one from below, and one from the left. A wire can only be connected to a junction at one of its bends (see Figure 2). We then remove the line segment corresponding to the arm of the junction, as illustrated in Figure 4.

If the tiling of the wire connected from above is vertical, then one of the rectangles can be prolonged into the junction. The same holds for the wire connected from below. On the contrary, a rectangle can be prolonged from the wire connected from the left only if the tiling is horizontal. If a rectangle can be prolonged, we say that the wire *sends true*, otherwise it *sends false*. If a clause contains two positive literals x, y and one negative $\bar{z}$, the corresponding junction is as in Figure 4, or the 180° rotation of it. The wire corresponding to z is connected from the left or right, and the wires corresponding to x and y are connected from above and below, or vice versa. Therefore, the wire corresponding to x (respectively y) sends *true* if and only if x (respectively y) is set to *true*. On the contrary, the wire corresponding to z sends *true* if and only if z is set to *false*. If the clause has two negative literals, then the junction is horizontal, and the junction behaves likewise.

Lingas proved that when minimising the number of convex polygons in a tiling, for each junction at least one adjacent wire sends *true*. Before stating Lingas' lemma exactly, we need to explain the first step of the construction of the point set.

2.1 Construction of the polygon with holes

Let us consider one instance (F, G) of MPLSAT. Lingas states that the planarity of G implies that the junctions and the wires can be embedded as explained above, and so that they do not overlap [7]. Thus we obtain a polygon with holes, that we denote by Π. He adds without proof that the dimensions of Π are polynomially related to $|V|$, where V denotes the vertex set of G. We show in the appendix how to embed the polygon with holes into a grid Λ, such that each edge consists of line segments of Λ. Moreover Λ is of size $\Theta(|V|^2)$. We can now state Lingas' lemma:

▶ **Lemma 4** (Lingas [7]). *In a minimum tiling with convex sets of Π, a junction contains wholly at least three convex sets. The junction contains wholly exactly three if and only if at least one of the wires connected to the junction sends* true.

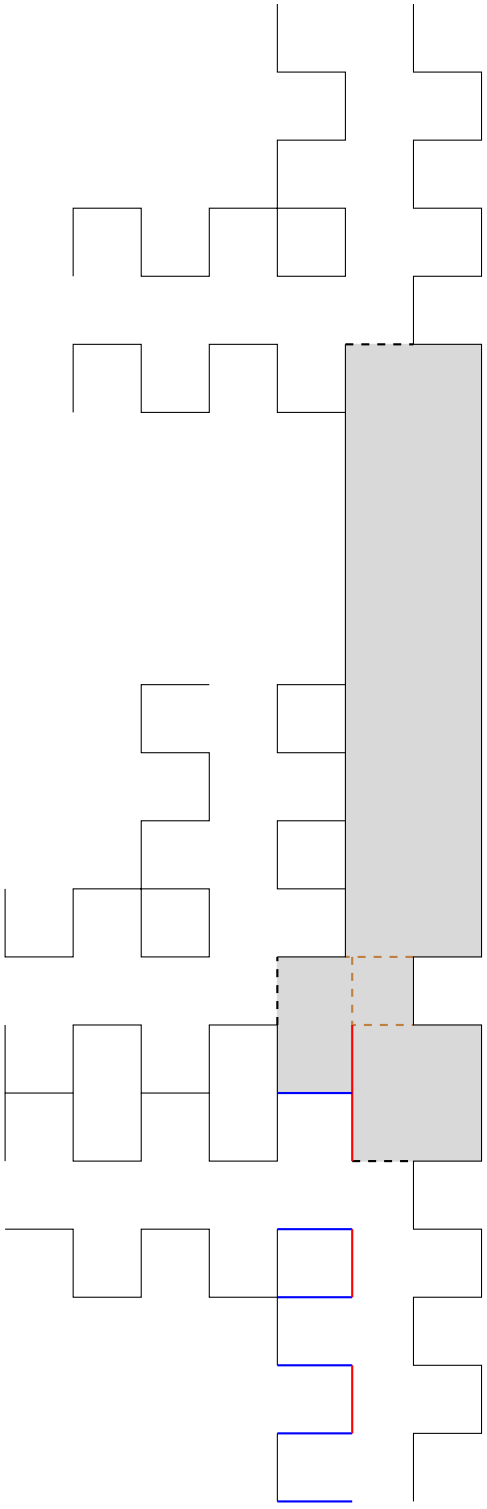


Figure 4 A junction for the MCT problem.

2.2 Discretisation of the line segments

To construct the point set, we first construct a collection of line segments. We then discretise this collection by replacing each line segment by a set of collinear points.

Let us consider our polygon with holes Π that lies in the grid Λ. The grid consists of points with integer coordinates, and line segments between points that are at distance 1. We consider the collection of line segments consisting of Π union each line segment of Λ whose interior is not contained in the interior of Π. Notice that therefore we have line segments outside Π, but also inside its holes. Moreover, the collection of line segments that we obtain, denoted by Φ, is a subgraph of the grid graph Λ.

Now we define K as twice the number of unit squares in Λ plus 1. Finally, we replace each line segment in Φ by K points evenly spaced. We denote this point set by P.

3 Proof of correctness

We have constructed P in order to have the following property:

▶ **Lemma 5.** *In a minimum convex tiling Σ of P, for each convex set $S \in \Sigma$, the interior of S does not intersect Φ.*

Let K' denote the number of unit squares in Φ, plus the minimum number of rectangles in a partition of the wires, plus three times the number of clauses. Using Lemmas 3 and 4 shown by Lingas coupled with Lemma 5, we immediately obtain the following theorem:

▶ **Theorem 6.** *The formula F is satisfiable if and only if there exists a convex tiling of P with K' polygons.*

Since P and K' can be computed in polynomial time, Theorem 6 implies Theorem 2 for the MCT problem. Due to lack of space, we postpone most of the proof of Lemma 5 to the Appendix. Nonetheless, we state and prove here the lemma giving the key idea of the proof. We use a packing argument, and claim that in a convex tiling Σ of P, if a convex set $S \in \Sigma$ has large area, then most of its area is contained in a unique cell of Φ. In the Appendix we show that in a minimum convex tiling, all convex sets have large area, and that each of them fills the cell that contains it. For a set S, let $\mathrm{A}(S)$ be the area of S.

▶ **Lemma 7.** *Let L and L' be two squares in Λ, and S be a convex polygon whose interior does not contain any point in P. If $\mathrm{A}(S \cap L) > 1/K$, and the boundary of S crosses a line segment of Φ between L and L', then $\mathrm{A}(S \cap L') \leq 1/K$.*

Proof. The proof is illustrated in Figure 5. By assumption, S goes between two points p and q at distance $1/K$. Let us consider the two line segments s and s' of the boundary of S that intersect the line ℓ spawned by p and q. Assume for contradiction that the lines spawned by s and s' do not intersect, or intersect on the side of ℓ where L lies. This implies that $S \cap L$ is contained in a parallelogram that has area $1/K$, as illustrated in Figure 6. Indeed such a parallelogram has base $1/K$ and height 1, therefore $\mathrm{A}(S \cap L) \leq 1/K$. This shows that the lines spawned by s and s' intersect on the side of ℓ where L' lies. Using the same arguments as above, this implies $\mathrm{A}(S \cap L') \leq 1/K$. ◀

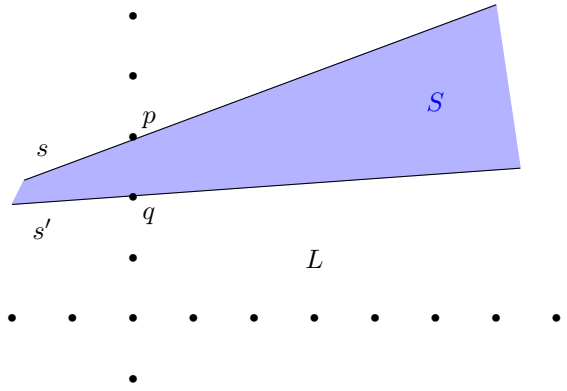


Figure 5 If $A(S \cap L) > 1/K$, the two lines spawned by s and s' intersect on the left side.

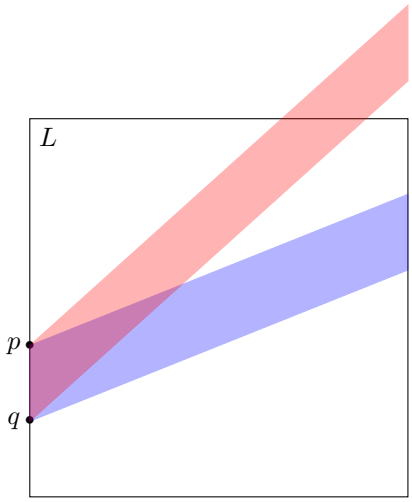


Figure 6 The area of the parallelograms is $1/K$.

4 Open problems

It is still open whether MCT is in NP. It may be that the coordinates of a vertex of a polygon require exponentially many bits to be written. We also do not know the complexity of MCP and MCT when it is assumed that no three points are collinear. A key property used for our proof can be summarised as follow: When a rectangle with large area is delimited by a lot of points, it is optimal to take this rectangle in the convex tiling or partition. But this cannot be achieved when no three points are on a line, as illustrated in Figure 7. In any convex partition the red edges are forced. Even in a convex tiling, one can observe that they are needed for the tiling to be minimum. This implies that three convex sets are necessary. But adding the convex set consisting of the points in convex position would add one non-necessary convex set. Adding more points to delimit the convex set cannot change the fact that taking this convex set in the tiling would not be optimal. The construction can easily be adapted for the MCP problem. Therefore it is not clear how one could force some convex sets to be in the partition or tiling.

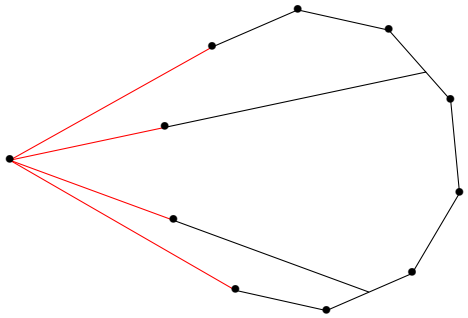


Figure 7 An optimal tiling splits the area delimited by points in convex position.

References

1 Allan S. Barboza, Cid C. de Souza, and Pedro J. de Rezende. Minimum convex partition of point sets. In *International Conference on Algorithms and Complexity*, pages 25–37. Springer, 2019. doi:/10.1007/978-3-030-17402-6_3.

2 Erik Demaine, Sándor Fekete, Phillip Keldenich, Dominik Krupke, and Joseph S. B. Mitchell. CG:SHOP 2020. https://cgshop.ibr.cs.tu-bs.de/competition/cg-shop-2020. Accessed: 12/02/2020.

3 Thomas Fevens, Henk Meijer, and David Rappaport. Minimum convex partition of a constrained point set. *Discrete Applied Mathematics*, 109(1-2):95–107, 2001. doi:10.1016/S0166-218X(00)00237-7.

4 Jesús García-López and Carlos M Nicolás. Planar point sets with large minimum convex decompositions. *Graphs and Combinatorics*, 29(5):1347–1353, 2013. doi:10.1007/s00373-012-1181-z.

5 J Mark Keil. Decomposing a polygon into simpler components. *SIAM Journal on Computing*, 14(4):799–817, 1985. doi:10.1137/0214056.

6 Christian Knauer and Andreas Spillner. Approximation algorithms for the minimum convex partition problem. In *Scandinavian Workshop on Algorithm Theory*, pages 232–241. Springer, 2006. doi:10.1007/11785293_23.

7 Andrzej Lingas. The power of non-rectilinear holes. In *International Colloquium on Automata, Languages, and Programming*, pages 369–383. Springer, 1982. doi:10.1007/BFb0012784.

8 Toshinori Sakai and Jorge Urrutia. Convex decompositions of point sets in the plane. *arXiv preprint arXiv:1909.06105*, 2019.

Computing the Fréchet distance of trees and graphs of bounded treewidth*

Maike Buchin[1], Amer Krivošija[2], and Alexander Neuhaus[3]

1 **Ruhr-Universität Bochum, Germany**
`maike.buchin@rub.de`
2 **TU Dortmund, Germany**
`amer.krivosija@tu-dortmund.de`
3 **TU Dortmund, Germany**
`alexander2.neuhaus@tu-dortmund.de`

Abstract

We give algorithms to compute the Fréchet distance of embedded trees and graphs with bounded treewidth. Our algorithms run in $\mathcal{O}(n^2)$ time for trees with fixed root and of bounded degree, and $\mathcal{O}(n^2\sqrt{n \log n})$ time for trees of arbitrary degree. For graphs of bounded treewidth we show one can compute the Fréchet distance in FPT time.

1 Introduction

The Fréchet distance, a distance measure for curves introduced by Fréchet in [8], is a popular measure for comparing polygonal curves. It is defined via homeomorphisms between the parameter spaces of the curves. Intuitively imagine a man walking his dog. The Fréchet distance between the paths of man and dog is the shortest length of a leash connecting them.

The Fréchet distance is well studied. Alt and Godau [4] gave a polynomial time algorithm to compute the Fréchet distance between two curves, sparking research in many applications, such as character recognition [12] and navigation on road maps [13]. One can also define the Fréchet distance between other objects like surfaces [3] or polygons [5]. Here we study the Fréchet distance of straight-line embedded graphs, i.e., every vertex of the graph is assigned to a point in the metric space and every edge between two vertices is a straight line segment between the respective points. First we observe that two graphs are homeomorphic in the topological sense, if they are isomorphic and do not contain degree 2 vertices [5]. That is, a graph homeomorphism induces a graph isomorphism on graphs without degree 2 vertices, and vice versa. Hence we assume that the graphs do not contain degree 2 vertices and define the Fréchet distance between embedded graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_1, E_1)$ as:

$$\delta_F(G_1, G_2) = \min_{\pi\colon G_1 \mapsto G_2} \max_{v \in V_1} \|v - \pi(v)\|$$

where $\pi\colon G_1 \to G_2$ is an isomorphism and $\|v - \pi(v)\|$ is the distance between the points corresponding to v and $\pi(v)$. If the graphs do have degree 2 vertices we need to contract every path consisting of degree 2 vertices (except the endpoints) as follows. We view these paths as embedded curves and exchange them with marked edges between their endpoints. We still have to take these curves into account when computing the Fréchet distance of the original graphs. To do so we use the embedded curves of marked edges when computing the Fréchet distance of the graphs.

* This work is based on the BSc thesis and student research project by the third author A. Neuhaus. A. Krivošija was supported by the German Science Foundation (DFG) Collaborative Research Center SFB 876 "Providing Information by Resource-Constrained Analysis", project A2. A more detailed version of this work is available at `https://arxiv.org/abs/2001.10502`.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Related work: To the best of our knowledge, the Fréchet distance of graphs (in this setting) has only been considered by Buchin et al. [5]. They studied the hardness of the Fréchet distance between surfaces and also discuss the Fréchet distance of graphs. In particular, they sketch how to decide in time $\mathcal{O}(n^2 \log n)$ whether there is an isomorphism between two embedded trees respecting a given distance δ. An isomorphism *respects* δ if the distance between every vertex and its image is at most δ. This decider can then be used by a binary search over all possible distances to compute the Fréchet distance. We expand upon this idea and show that one can compute the Fréchet distance between two trees even faster when computing the Fréchet distance between every subtree in a bottom up fashion.

As one can compute the Fréchet distance between two trees it seems natural to look at *tree-like* graphs. We say a graph is tree-like if it has bounded treewidth, see Section 3. The definition of the Fréchet distance given above requires the graphs to be isomorphic. Only recently Lokshtanov et al. [11] gave a canonization algorithm showing that graph isomorphism for graphs of bounded treewidth can be decided in fixed-parameter tractable-time. Their algorithm however only decides whether two graphs are isomorphic but does not provide any isomorphism as a witness. Grohe et al. [9] gave an algorithm computing all isomorphisms between two input graphs. We alter this algorithm to compute the Fréchet distance of two embedded graphs with bounded treewidth.

Results: In Section 2 we give an algorithm computing the Fréchet distance between two embedded trees. We show that this can be done in $\mathcal{O}(n^2\sqrt{d \log d})$ time, where n is the maximum number of vertices and d is the maximum degree of the trees. In Section 3 we show how one can compute the Fréchet distance of two graphs with n vertices and treewidth at most k in $2^{\mathcal{O}(k \log^c k)} n^{\mathcal{O}(1)} \cdot \log n$ time.

2 Algorithm for trees

Computing the Fréchet distance of two graphs requires the graphs to be isomorphic. It is well known that the isomorphism problem for two trees can be solved in polynomial time [1]. If not stated otherwise the trees have a designated root. If they do not, we use the fact that a tree isomorphism needs to match the centers of the trees.

As stated above we require homeomorphisms between the graphs to compute their Fréchet distance. Hence we contract all paths of degree 2 vertices of the two input trees as follows. We start by finding all paths of maximum length containing only degree 2 vertices in each graph with a DFS. For each path found, we connect the endpoints with an edge and store the whole path for the distance calculation as an embedded curve. The last step is to delete all degree 2 vertices and their adjacent edges. For a graph $G = (V, E)$ this can be done in time $\mathcal{O}(|V| + |E|)$ as we basically perform a DFS. Next we show that we can compute all Fréchet distances between edges in time $\mathcal{O}(n^2 \log n)$.

▶ **Lemma 2.1.** *Let $G = (V, E)$ and $G' = (V', E')$ be two trees with $|V| = |V'| = n$. One can contract all paths of degree 2 vertices and store the Fréchet distances between each pair of edges of the contracted graphs in an array of size $\mathcal{O}(n^2)$. This procedure takes $\mathcal{O}(n^2 \log n)$ time, or $\mathcal{O}(n^2)$ time if at least one graph has only paths of degree 2 vertices of constant length.*

Proof. Given two trees $G = (V, E)$ and $G' = (V', E')$ with $|V| = |V'| = n$. Notice that the paths of degree 2 vertices are pairwise disjoint, except possibly for their endpoints. Thus these paths define a partition of V and V' respectively. Also we can bound the number of such paths in each graph by $\frac{n}{2}$. Let $\ell_1, \ldots, \ell_{\frac{n}{2}}$ and $\ell'_1, \ldots, \ell'_{\frac{n}{2}}$ be the lengths of such paths

within G and G' respectively. If there are less than $\frac{n}{2}$ paths the corresponding lengths are 0. It holds that $\sum_{i=1}^{n/2} \ell_i \leq n$ and $\sum_{i=1}^{n/2} \ell'_i \leq n$.

We compute the distance between each pair of edges in the contracted graphs. The easy case is when both edges are non-contracted and we can compute the distance in $\mathcal{O}(1)$ time. If one edge is contracted and the other is not (or is contracted but has constant length), this takes time linear in the length of the contracted edge, which sums up to $\sum_{i=1}^{n/2}(c \cdot \ell'_i) \leq cn$ for a single edge, and hence quadratic time overall.

To compute the Fréchet distances between each pair of contracted edges (of non-constant length), i.e. paths of degree 2 vertices, in G and G' we use the algorithm of Alt and Godau [4] to compute the Fréchet distance of two paths in time $T(\ell_i, \ell'_j) = \mathcal{O}(\ell_i \ell'_j \log(\ell_i \ell'_j))$. Hence in total we need time $\sum_{i=1}^{n/2} \sum_{j=1}^{n/2} T(\ell_i, \ell'_j) \leq cn^2 \log n$. Note that this computation time is only necessary if both graphs have long degree 2 paths.

We store the distances in an array of size $O(|E| \cdot |E'|) = \mathcal{O}(n^2)$ ◀

The further steps are executed on the contracted graphs.

We give a brief overview of the algorithm, based on the ideas in [5]. The algorithm computes the Fréchet distance in a bottom up way, comparing two nodes of same height in every step. Let T and T' be the two input trees and $t \in T, t' \in T'$ two vertices of the same height and equal degree. Let $t_1, \ldots, t_i$ and $t'_1, \ldots, t'_i$ be the children of t and t'. Assume we already have computed the Fréchet distance between the subtrees rooted at the children. To compute the Fréchet distance between the trees rooted at t and t' we follow the ideas in [5] using ε-matchings. To find those matchings we create a new bipartite graph B. For every subtree of t and t' there is a vertex in B. We combine every vertex corresponding to a subtree of t with every vertex corresponding to a subtree of t' with a weighted edge. The weight of the edge is the Fréchet distance between the subtrees, if there are unmarked edges between the roots of the subtrees and their parents. If at least one of the edges is marked we assign the maximum of the Fréchet distance between the edges and the Fréchet distance of the trees. If the subtrees are not isomorphic we assign ∞ as the edge weight. Note if one of the edges was marked we use the corresponding path to compute the distance.

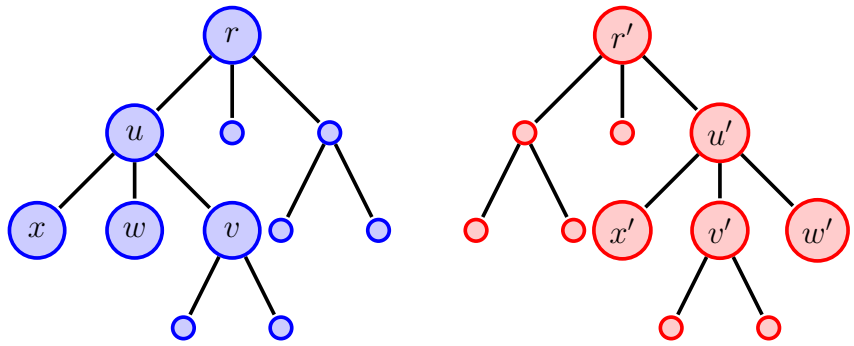
Now we find a *bottleneck matching* for this graph, which equals the ε-matching used in [5]. A bottleneck matching is a perfect matching such that the maximum weight of its edges is minimal for all perfect matchings possible. The Fréchet distance of the trees rooted at t and t' is either the maximum weight of the found bottleneck matching or the distance of t and t'. We store the correct value in a two-dimensional array. If we could not find a bottleneck matching the trees are not isomorphic and thus have no Fréchet distance. In this case we store ∞. If the subtrees are empty we store 0.

The algorithm iterates over both trees doing the above computation for each pair of nodes of same height. Afterwards the algorithm returns the value stored for the roots of the trees.

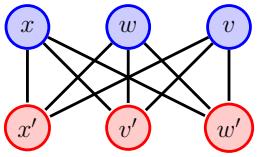
Figure 1 illustrates the computation of a bottleneck matching. Given the two trees in (a) as input. Consider the computation of the Fréchet distance of the trees rooted at u and u'. The graph B is shown in (b) and the distances are given in the table in (c). In (d) we see the computed bottleneck matching with maximum Fréchet distance of 1.

It remains to compute such a bottleneck matching. To do this store all edge weights in a sorted array and search for the smallest edge weight for which we can find a perfect matching. The matching corresponding to this weight is the desired bottleneck matching.

The running time of this algorithm is as follows. We use the algorithm of Alt et al. [2] to compute the perfect matchings. This means we can bound the running time of one bottleneck matching computation by $\mathcal{O}\left(d^2\sqrt{d \log d}\right)$ time, with d being the degree of the



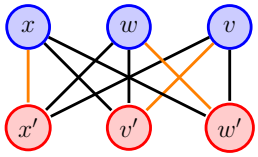
(a)



(b)

	v	w	x
v'	1	2	2
w'	2	1	2
x'	2	2	1

(c)



(d)

Figure 1 Two trees (a), the bipartite graph B (b), the weights (c) and bottleneck matching (d)

trees. A careful analysis of the number of bottleneck computations yields that the algorithm has running time $\mathcal{O}\left(n^2\sqrt{d\log d}\right)$. Thus we get the following result:

▶ **Theorem 2.2.** *The Fréchet distance of two embedded trees can be computed in time* $\mathcal{O}\left(n^2\sqrt{n\log n}\right)$. *If the trees have bounded degree the computation only takes* $\mathcal{O}(n^2)$ *time.*

Note that if we need more than constant time to compute the distance between two points the running time of this algorithm increases. Next we generalize this result to graphs that are not trees but are tree-like in their structure.

3 Algorithm for graphs with bounded treewidth

Here we consider a larger class of graphs. It is known that many hard problems for graphs can be solved easily on trees, such as 3-coloring problem and finding a vertex cover. Both these problems can be solved in linear time on a tree. One can ask if these problems are easy for tree-like graphs. A measurement for such a likeliness is the *treewidth*. To define the treewidth we first introduce *tree decompositions*. Intuitively a tree decomposition of a graph G represents the vertices and edges of G as subgraphs inside a tree. Formally:

▶ **Definition 3.1.** Let $G = (V, E)$ be a graph. Let T be a tree and β a function mapping all $t \in T$, the nodes, to subsets of V. We call (β, T) a tree decomposition of G if:

1. for each vertex $v \in V$ it holds that all subsets $\beta(t) \in T$, called bags, containing v induce a nonempty connected subtree of T, and
2. for each edge $(u, v) \in E$ there is one $t \in T$, such that the bag $\beta(t)$ contains u and v.

The width of such a tree decomposition is the size of its largest bag -1. The treewidth of G is the minimal width among all its tree decompositions.

Tree decompositions play an important role in the context of parameterized algorithms. An algorithm is parameterized, if its running time does not only depend on the input size n

but also on a parameter k. A problem is fixed parameter tractable, if there is a algorithm with running time $f(k) \cdot \text{poly}(n)$ for it. Many graph problems have algorithms with treewidth as parameter, see [6, 7]. Typically these algorithms use dynamic programming over a tree decomposition of the graph. Our algorithm is based on the algorithm of Grohe et al. [9]. We extended their algorithm such that we can compute all isomorphisms between two embedded graphs that respect a distance limit. Using this algorithm we conduct a binary search on every possible distance between nodes of G_1 and G_2. The Fréchet distance of the two graphs is the smallest distance found that is respected by an isomorphism.

We describe our algorithm. Let $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$ be two embedded graphs and δ a distance limit. We start by preprocessing the graphs the same way as in the case of trees, see Section 2. Next we compute an initial tree decomposition using the techniques of Leimer [10]. The resulting decomposition is known to be isomorphism invariant. We carefully refine these tree decompositions in a bottom up way using the techniques of [9]. In every step we maintain the invariant that the tree decompositions are isomorphism invariant, the size of the resulting bags is not too high, and we get crucial information about the structure of both graphs. We now use the resulting tree decomposition to compute the isomorphisms respecting δ in a bottom up way. Suppose we are looking at two nodes $t_1 \in T_1$ and $t_2 \in T_2$, and have already computed all isomorphisms respecting δ between the subtrees of t_1 and t_2 rooted at their respective children. To compute the isomorphisms between the graphs induced by the trees rooted at t_1 and t_2 Grohe et al. [9] used an abstraction called coset-hypergraph isomorphism. An instance of this abstraction consists of the two graphs, the isomorphisms computed between the subtrees one can extend to isomorphisms between the graphs, and colorings of the graphs to indicate which node sets can be mapped to each other.

We expand upon this approach by using the colorings to indicate which sets are isomorphic to each other, and respect a given distance limit. Especially the colorings can indicate if two edges correspond to curves that can be mapped to each other. Using these colorings we compute all isomorphisms between the graphs using the algorithm of [9]. The algorithm uses dynamic programming over the tree decompositions in a bottom up way. In every step it constructs all isomorphisms between the subgraphs induced by vertices that were already covered by the tree decompositions. During this the algorithm can find isomorphisms that do not respect the given distance limit. In this case we simply mark such an isomorphism, indicating that it does not witness the distance limit. After this computation we simply check if there is at least one isomorphism that is not marked. In this case we know that the input graphs are isomorphic under the given distance limit. If the dimension of the space where the graphs are embedded is constant, the running time of the algorithm by Grohe et al. [9] does not change. This means one can decide whether two embedded graphs with treewidth at most k are isomorphic under a given distance limit in time $2^{\mathcal{O}(k \log^c k)} n^{\mathcal{O}(1)}$, where n is the number of vertices of one graph, and c is a positive constant.

Furthermore we know that the Fréchet distance between the two graphs must be a distance between two vertices, with one belonging to the first and one to the second graph. Hence we store all n^2 distances in a sorted array and perform a binary search using the above algorithm to find the smallest distance under which the graphs are isomorphic. This yields:

▶ **Theorem 3.2.** *The Fréchet distance between two graphs with n nodes and treewidth at most k can be computed in time $2^{\mathcal{O}(k \log^c k)} n^{\mathcal{O}(1)} \cdot \log n$.*

4 Conclusion

We have shown how to compute the Fréchet distance of two rooted trees and of two graphs with bounded treewidth. It would be interesting to investigate the case of trees with high vertex degrees. For those trees the algorithm presented in Section 2 only gives a slight improvement over the algorithm sketched in [5]. The more general case remains interesting since a polynomial time algorithm for computing the Fréchet distance of two graphs could be used to decide whether the graphs are isomorphic. We do not expect a polynomial time algorithm for general graphs. But perhaps there are other parameters in which the Fréchet distance of two graphs is fixed parameter tractable and can be computed more efficiently.

References

1 A. V. Aho and J. E. Hopcroft. *The Design and Analysis of Computer Algorithms*. Addison-Wesley Longman Publishing, Boston, MA, USA, 1st edition, 1974.

2 H. Alt, N. Blum, K. Mehlhorn, and M. Paul. Computing a maximum cardinality matching in a bipartite graph in time $O(n^{1.5}\sqrt{m/\log n})$. *Inf. Process. Lett.*, 37(4):237–240, Feb. 1991.

3 H. Alt and M. Buchin. Can we compute the similarity between surfaces? *Discrete & Computational Geometry*, 43(1):78, Mar 2009.

4 H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *Int. J. Comput. Geometry Appl.*, 5:75–91, 03 1995.

5 K. Buchin, M. Buchin, and A. Schulz. Fréchet distance of surfaces: Some simple hard cases. In M. de Berg and U. Meyer, editors, *Algorithms – ESA 2010*, pages 63–74. Springer Berlin Heidelberg, 2010.

6 R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 2012.

7 J. Flum and M. Grohe. *Parameterized Complexity Theory (Texts in Theoretical Computer Science. An EATCS Series)*. Springer-Verlag, Berlin, Heidelberg, 2006.

8 M. M. Fréchet. Sur quelques points du calcul fonctionnel. *Rendiconti del Circolo Matematico di Palermo (1884-1940)*, 22(1):1–72, Dec 1906.

9 M. Grohe, D. Neuen, P. Schweitzer, and D. Wiebking. An improved isomorphism test for bounded-tree-width graphs. In I. Chatzigiannakis, C. Kaklamanis, D. Marx, and D. Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP,*, pages 67:1–67:14, 2018.

10 H.-G. Leimer. Optimal decomposition by clique separators. *Discrete Mathematics*, 113(1):99–123, 1993.

11 D. Lokshtanov, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. Fixed-parameter tractable canonization and isomorphism test for graphs of bounded treewidth. *SIAM Journal on Computing*, 46(1):161–189, 2017.

12 S. Sen, J. Chakraborty, S. Chatterjee, R. Mitra, R. Sarkar, and K. Roy. Online handwritten bangla character recognition using Fréchet distance and distance based features. In S. Sundaram and G. Harit, editors, *Document Analysis and Recognition*, pages 65–73, Singapore, 2019. Springer Singapore.

13 K. P. Sharma, R. C. Pooniaa, and S. Sunda. Map matching algorithm: curve simplification for Fréchet distance computing and precise navigation on road network using RTKLIB. *Cluster Computing*, 22(6):13351–13359, Nov 2019.

On the complexity of the middle curve problem*

Maike Buchin[1], Nicole Funk[2], and Amer Krivošija[3]

1 **Ruhr-Universität Bochum, Germany**
`maike.buchin@rub.de`
2 **Department of Computer Science, TU Dortmund, Germany**
`nicole.funk@tu-dortmund.de`
3 **Department of Computer Science, TU Dortmund, Germany**
`amer.krivosija@tu-dortmund.de`

Abstract

For a set of curves, Ahn et al. [1] introduced the notion of a *middle curve* and gave algorithms computing these with run time exponential in the number of curves. Here we study the computational complexity of this problem: we show that it is NP-complete and give approximation algorithms.

1 Introduction

Consider a group of birds migrating together. Several of these birds are GPS-tagged to analyze their behavior. The resulting data is a set of sequences of their positions. Such a sequence of data points can be interpreted as a polygonal curve. We want to represent the movement of the whole group, for instance to compare it to other groups or species. For this, we use a representative curve. Such a representative curve is also useful in other applications, such as the analysis of handwritten text or speech recognition.

There have been a few different approaches of defining such a representative curve. Buchin *et al.* [4] defined the median level of curves as only using parts input curves, where the median can change directions where two input curves cross paths. Har-Peled and Raichel [10] define a mean curve, which can be chosen freely and minimizes the distance to the input curves. They give an algorithm exponential in the number of curves for computing this.

Another approach is a version of the (k, ℓ)-center problem, which asks for a set of k center curves of complexity at most ℓ for which the distance of each input curve to its nearest center is minimized. In particular, the $(1, \ell)$-center problem asks for only one such center curve. The (k, ℓ)-center problem for curves was first introduced by Driemel *et al.* [8] and further analyzed by Buchin *et al.* [5] and Buchin *et al.* [6].

However, none of these representative curves use only actual data points of the GPS tracks. This could lead to the representative curves containing positions that the moving entities (e.g. birds) could not have visited. As the data points in the input curves are more reliable Ahn *et al.* [1] defined the *middle curve* to only use these points. For a more accurate representation of the original curves, Ahn *et al.* [1] define three variants of the middle curve. We use their definition of a middle curve in this paper.

Related work Ahn *et al.* [1] presented algorithms for all three variants of the middle curve problems, whose running time is exponential in the number of input curves. For several

* This work is based on the student research project by the second author N. Funk. A. Krivošija was supported by the German Science Foundation (DFG) Collaborative Research Center SFB 876 "Providing Information by Resource-Constrained Analysis", project A2. A full version of this work is available at `http://arxiv.org/abs/2001.10298`

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

representative curve problems it is known that they are NP-hard, such as (k, ℓ)-center [5, 8], minimum enclosing ball [5], (k, ℓ)-median [8], 1-median under Fréchet and dynamic time warping distance [6, 7]. Some problems are NP-hard even to approximate better than a constant factor, e.g. (k, ℓ)-center problem [5]. Similarly, Buchin *et al.* [3] showed, that assuming the Strong Exponential Time Hypothesis (SETH) the Frechet distance of k curves of complexity n each cannot be computed significantly faster than $O(n^k)$ time.

Our results We prove NP-completeness of the Middle Curve problem presented by Ahn *et al.* [1]. Next we define a parameterized version of the problem, and present a simple exact algorithm as well as an $(2 + \varepsilon)$-approximation algorithm for the parameterized problem.

2 Preliminaries

A polygonal curve P is given by a sequence of vertices $\langle p_1, \ldots, p_m \rangle$ with p_i in $\mathbb{R}^d$, $1 \leq i \leq m$, and for $1 \leq i < m$ the pair of vertices (p_i, p_{i+1}) is connected by the straight line segment $\overline{p_i p_{i+1}}$. We call the number of vertices m of the curve its *complexity*. Let the input consist of n polygonal curves $\mathcal{P} = \{P_1, \ldots, P_n\}$, each of complexity m.

Fréchet Distance We define the discrete Fréchet distance of two curves $P' = \langle p'_1, \ldots, p'_{m'} \rangle$ and $P'' = \langle p''_1, \ldots, p''_{m''} \rangle$ as follows: we call a *traversal* T of P' and P'' a sequence of pairs of indices (i, j) of vertices $(p'_i, p''_j) \in P' \times P''$ such that

i) the traversal T begins with $(1, 1)$ and ends with (m', m''), and
ii) the pair (i, j) of T can be followed only by one of $(i + 1, j)$, $(i, j + 1)$, or $(i + 1, j + 1)$.

We note that every traversal is monotone. Denote $\mathcal{T}$ the set of all traversals T of P' and P''. The discrete Fréchet distance between P' and P'' is defined as:

$$d_{dF}(P', P'') = \min_{T \in \mathcal{T}} \max_{(i,j) \in T} \|p_i - q_j\|_2.$$

We call the set of pairs of vertices $(p', p'') \in P' \times P''$ that realize $d_{dF}(P', P'')$ a *matching*, and say that these pairs of vertices are matched. A related similarity measure is the continuous Fréchet distance d_F, we refer to [2] and the full version for details. Both d_{dF} and d_F are metrics.

Middle Curve Given a set of n polygonal curves $\mathcal{P}$, a value $\delta \geq 0$, and a distance measure γ for polygonal curves. We use $\gamma = d_{dF}$ as in [1], for the continuous Fréchet distance d_F the definitions hold verbatim. A **middle curve** at distance δ to $\mathcal{P}$ is a curve $M = \langle m_1, \ldots, m_\ell \rangle$ with vertices $m_i \in \bigcup_{P_j \in \mathcal{P}} \bigcup_{p \in P_j} \{p\}$, $1 \leq i \leq \ell$, s.t. $\max\{d_{dF}(M, P_j) : P_j \in \mathcal{P}\} \leq \delta$ holds.

If the vertices of a middle curve M respect the order given by the curves of $\mathcal{P}$, then we call M an **ordered middle curve**. Formally, for all $1 \leq j \leq n$, if the vertex $m_i \in M$ is matched to $p_k \in P_j$ realizing $d_{dF}(M, P_j)$, then for the vertices $m_{i'} \in M$, $i < i'$, it holds that $m_{i'} \in \left(\bigcup_{P_x \in \mathcal{P} \setminus P_j} \bigcup_{p \in P_x} \{p\} \right) \cup (\bigcup \{p_{k'} : p_{k'} \in P_j, k' > k\})$. If the vertices of M are matched to themselves in their original curves $P \in \mathcal{P}$ in the matching realizing $d_{dF}(M, P) \leq \delta$, we have a **restricted middle curve**. Note that an ordered middle curve is a middle curve, and a restricted middle curve is ordered as well.

We define the decision problem corresponding to finding such a curve. Given a set of polygonal curves $\mathcal{P} = \{P_1, \ldots, P_n\}$ and a $\delta \geq 0$ as parameters. Unordered Middle Curve problem returns true iff there exists a middle curve M at distance δ to $\mathcal{P}$. The Ordered

Middle Curve and Restricted Middle Curve returns true iff there exists an ordered and a restricted middle curve respectively at distance δ to $\mathcal{P}$.

Ahn *et al.* [1] presented dynamic programming algorithms for each variant of the middle curve problem. The running times of these algorithms for $n \geq 2$ curves of complexity at most m are $O(m^n \log m)$ for the unordered case, $O(m^{2n})$ for the ordered case, and $O(m^n \log^n m)$ for the restricted middle curve case. All three cases have running time exponential in n, yielding the question if there is a lower bound for these problems. In the following section we prove that the Middle Curve problem is NP-complete.

3 NP-completeness

The technique for the proof that all variants of the Middle curve are NP-hard is based on the proof by Buchin *et al.* [5] and Buchin, Driemel, and Struijs [6] for the NP-hardness of the Minimum enclosing ball and 1-median problems for curves under Fréchet distance. Their proof is a reduction from the Shortest Common Supersequence (SCS), which is known to be NP-hard [11]. SCS problem gets as input a set $S = \{s_1, \ldots, s_n\}$ of n sequences over a binary alphabet $\Sigma = \{A, B\}$ and $t \in \mathbb{N}$. SCS returns true iff there exists a sequence s^* of length at most t, that is a supersequence of all sequences in S.

Our NP-hardness proof differs from the proof of [5, 6] in three aspects. First, the mapping of the characters of the sequence is extended by additional points. Second, in order to validate all three variants of our problem, the conditions of the restricted middle curve have to be fulfilled, i.e. each vertex has to be matched to itself. Third, our representative curve is limited to the vertices of the input curves. Due to the hierarchy of the middle curve problems we show the reductions from SCS to the Restricted Middle Curve, and from Unordered Middle Curve to SCS. We get the following theorem.

▶ **Theorem 3.1.** *Every variant of Middle Curve problem for the discrete and the continuous Fréchet distance is NP-hard.*

With this we can prove the NP-completeness of the Middle Curve decision problem. Given a Middle Curve instance $(\mathcal{P}, \delta)$ with $\mathcal{P}$ containing n curves of complexity m, we guess non-deterministically a middle curve M of complexity ℓ. We can decide whether the Fréchet distance between M and a curve $P \in \mathcal{P}$ is at most δ in $\mathcal{O}(m\ell)$ time using the algorithm by Alt and Godau [2] for the continuous, and by Eiter and Mannila [9] for the discrete Fréchet distance. We note that the algorithm by Alt and Godau [2] has to be modified a bit, as it uses a random access machine instead of a Turing machine, as this allows the computation of square roots in constant time. But comparing the distances is possible by comparing the squares of the square roots, thus this results in a non-deterministic $\mathcal{O}(nm\ell)$-time algorithm for the Middle Curve problem.

In order to decide the Ordered Middle Curve problem, it is necessary to compare the middle curve to the input curves, which is possible in $O(nm)$ time. For the restricted Restricted Middle Curve problem the matching corresponding to the Frechet distance $\leq \delta$ has to be known. This matching is a result of the decision algorithm by Alt and Godau [2]. Given this matching it can be checked in $\mathcal{O}(m + \ell)$ time if a vertex is matched to itself. This yields the following theorem.

▶ **Theorem 3.2.** *Every variant of the Middle Curve problem for the discrete or continuous Fréchet distance is NP-complete.*

If the SCS problem is parameterized by the number of input sequences n, it is known to be W[1]-hard [7]. In our reduction from SCS the number of input curves in the constructed

Middle Curve instance is $n+2$. Thus the shown reduction is also a parameterized reduction from SCS with the parameter n to the Middle Curve problem parameterized by the number of input curves, yielding the following theorem.

▶ **Theorem 3.3.** *The* Middle Curve *problem for the discrete and continuous Fréchet distance parameterized by the number of input curves n is W[1]-hard.*

4 Approximation algorithm

A different way of parameterizing the Middle Curve problem is to use the complexity of the middle curve. Given a set of polygonal curves $\mathcal{P}$, a $\delta \geq 0$, and a parameter $\ell \in \mathbb{N}$. We define the parameterized middle curve decision problems, that return true iff a middle curve of complexity $\leq \ell$ with corresponding conditions exists (for each of the three variants).

It is clear that there exists a simple brute force optimization algorithm for the Parameterized Middle Curve instance $(\mathcal{P}, \delta, \ell)$, that tests all ℓ-tuples of the vertices from the curves in $\mathcal{P}$ in $\mathcal{O}\big((mn)^{\ell} m\ell \log m\ell\big)$. This holds for all three versions of the problem.

We want to give an approximation algorithm for Parameterized Middle Curve optimization problem for the discrete Frechet distance. For this we use an approximation of the (k, ℓ)-center optimization problem on curves. The (k, ℓ)-center problem for curves was introduced by Driemel *et al.* [8]. Given a set $\mathcal{P} = \{P_1, \ldots, P_n\}$ of polygonal curves of complexity at most m, it looks for a set of curves $\mathcal{C} = \{C_1, \ldots, C_k\}$, each of complexity at most ℓ, that minimizes $\max_{P \in \mathcal{P}} \min_{i=1}^{k} \gamma(C_i, P)$ for a distance measure γ. The unordered Parameterized Middle Curve optimization problem is a $(1, \ell)$-center problem, where the curve C_1 is limited to vertices from the input curves and the distance measure γ is a variant of the Fréchet distance.

Given a set $\mathcal{P}$ of n curves of complexity m in $\mathbb{R}^d$, let C be the $(1, \ell)$-center curve returned by some α-approximation algorithm for the discrete Fréchet distance. Let $\delta = \max_{P \in \mathcal{P}} d_{dF}(C, P)$. We construct d-dimensional balls centered at vertices of the curve C with radius δ. It holds that $d_{dF}(C, P) \leq \delta$, $\forall P \in \mathcal{P}$, thus in each ball centered at the vertices of C there has to be a vertex of each curve from $\mathcal{P}$. We choose at random one vertex from each of the ℓ balls, and connect them with line segments in the order of the vertices along C. We denote the curve we got with M, and claim that it is a good approximation of an unordered parameterized middle curve. See Figure 1 for an illustration of the algorithm.

Let C^* be an optimal $(1, \ell)$-center curve (for the discrete Fréchet distance) for the given input set $\mathcal{P}$. Let $\delta^* = \max_{P \in \mathcal{P}} d_{dF}(C^*, P)$. It holds that $\delta \leq \alpha\delta^*$. For each $P \in \mathcal{P}$ and each vertex of P, there is a vertex in M, that is at distance at most 2δ (diameter of the ball both of them lie in). Thus there is a traversal of P and M with pairwise distance of the vertices at most 2δ, implying $d_{dF}(M, P) \leq 2\delta$. We have $d_{dF}(M, P) \leq 2\delta \leq 2\alpha\delta^*$.

Let the optimal parameterized middle curve with complexity ℓ be M^*. By definition it holds that $\delta^* = \max_{P \in \mathcal{P}} d_{dF}(C^*, P) \leq \max_{P \in \mathcal{P}} d_{dF}(M^*, P)$. Thus

$$d_{dF}(M, P) \leq 2\alpha \max_{P \in \mathcal{P}} d_{dF}(M^*, P)\,,$$

and M is a 2α-approximation to the optimal parameterized middle curve. This implies:

▶ **Lemma 4.1.** *Given a set of n curves $\mathcal{P}$ each with complexity at most m, a $\delta > 0$ and an α-approximation algorithm for (k, ℓ)-center with running time T, we can compute a 2α-approximation of the* Parameterized Middle Curve *optimization problem for discrete Fréchet distance in $O(\ell mn + T)$ time.*

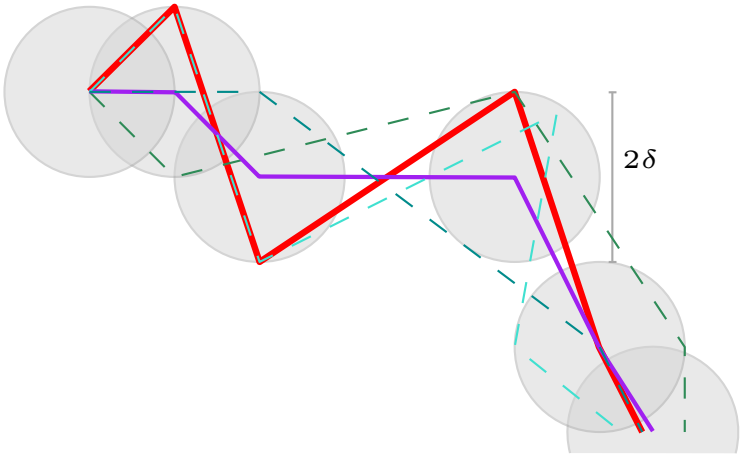


Figure 1 Illustration of the approximation algorithm. The input curves are dashed and in shades of green, while the $(1, \ell)$-center approximation with distance δ is the full purple curve. The constructed middle curve is the red fat curve.

Plugging the $(1+\varepsilon)$-approximation algorithm of Buchin *et al.* [6] for (k, ℓ)-center for discrete Fréchet distance into Lemma 4.1, we get

▶ **Theorem 4.2.** *Given a set of n curves $\mathcal{P}$ each of complexity at most m, and a $\delta > 0$, we can compute a $(2+\varepsilon)$-approximation of the* Parameterized Middle Curve *optimization problem for discrete Fréchet distance in $\mathcal{O}\big(((c\ell)^\ell + \log(\ell+n))\ell mn\big)$ time, with $c = (\frac{4\sqrt{d}}{\varepsilon}+1)^d$.*

5 Conclusion

We showed that the Middle Curve problem is NP-complete and gave a $(2+\varepsilon)$-approximation for the Parameterized Middle Curve problem, parameterized in the complexity of the middle curve. It would be interesting to gain further insight into the complexity of the parameterized problem. Fixing the parameter in the brute-force algorithm gives an XP-algorithm, however it remains open whether Parameterized Middle Curve is in FPT.

References

1 H.-K. Ahn, H. Alt, M. Buchin, E. Oh, L. Scharf, and C. Wenk. A middle curve based on discrete Fréchet distance. In E. Kranakis, G. Navarro, and E. Chávez, editors, *LATIN 2016: Theoretical Informatics - 12th Latin American Symposium*, pages 14–26, 2016.

2 H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 05:75–91, 1995.

3 K. Buchin, M. Buchin, M. Konzack, W. Mulzer, and A. Schulz. Fine-grained analysis of problems on curves. In *Proceedings of the 32nd European Workshop on Computational Geometry*, 2016.

4 K. Buchin, M. Buchin, M. van Kreveld, M. Löffler, R. I. Silveira, C. Wenk, and L. Wiratma. Median trajectories. *Algorithmica*, 66(3):595–614, 2013.

5 K. Buchin, A. Driemel, J. Gudmundsson, M. Horton, I. Kostitsyna, M. Löffler, and M. Struijs. Approximating (k, ℓ)-center clustering for curves. In T. M. Chan, editor, *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 2922–2938, 2019.

6 K. Buchin, A. Driemel, and M. Struijs. On the hardness of computing an average curve. *CoRR*, abs/1902.08053, 2019.

7 L. Bulteau, F. Hüffner, C. Komusiewicz, and R. Niedermeier. Multivariate algorithmics for NP-hard string problems. *Bulletin of the EATCS*, 114, 2014.

8 A. Driemel, A. Krivošija, and C. Sohler. Clustering time series under the Fréchet distance. In R. Krauthgamer, editor, *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 766–785, 2016.

9 T. Eiter and H. Mannila. Computing discrete Fréchet distance. Technical Report CD-TR 94/64, Christian Doppler Laboratory, 1994.

10 S. Har-Peled and B. Raichel. The Fréchet distance revisited and extended. *ACM Transactions on Algorithms*, 10(1):3:1–3:22, 2014.

11 K. Pietrzak. On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems. *Journal of Computer and System Sciences*, 67(4):757–771, 2003.

t-spanners for Transmission Graphs Using the Path-Greedy Algorithm

Stav Ashur[1] and Paz Carmi[1]

1 Ben-Gurion University of the Negev

Abstract

Let $D = \{d_1, \ldots, d_n\}$ be a set of n disks in the plane, and let $C = \{c_1, \ldots, c_n\}$ be their centers. The *transmission graph* $G = (C, E)$ has the centers of D as its vertex set C, and a directed edge (c_i, c_j) if and only if c_j lies in the disks associated with c_i.

A *t-spanner* G' for G is a sparse subgraph of G such that for any two vertices p, q connected by a directed path in G, there is a directed path from p to q in G' of length at most t times the length of the path from p to q in G.

In this paper, we consider the problem of computing a t-spanner with a linear number of edges and bounded in-degree for transmission graphs. We show that the well-known *Path-Greedy* algorithm produces such a t-spanner for transmission graphs, thus, providing a much simpler method than ones that are currently in use.

In addition, we show that the weight of the resulting t-spanner is $O(\log n \cdot wt(MST(D))(1 + \Psi))$, where Ψ is the ratio between the largest and smallest disk radii, and $wt(MST(D))$ is the weight of the MST built over the centers of the disks. To the best of our knowledge, this is the first upper bound on the weight of a t-spanner for transmission graphs.

1 Introduction

Given a directed graph G, let $\delta_G(p, q)$ be the shortest directed path from p to q in G, and let $|\delta_G(p, q)|$ denote its length. A t-spanner for a weighted directed graph $G = (V, E, w)$ is a sparse subgraph $G' \subseteq G$ such that every two vertices $p, q \in V$, connected by a directed path from p to q of weight $|\delta_G(p, q)|$, are connected in G' by a directed path of weight at most $t \cdot |\delta_G(p, q)|$, i.e. $|\delta_{G'}(p, q)| \leq t \cdot |\delta_G(p, q)|$. Algorithms for the construction of t-spanners for geometric graphs have been widely studied, and various results exist for different types of graphs, see [11] for a comprehensive survey.

Transmission graphs

Transmission graphs are a common model for communication networks composed of devices with different transmission ranges. The set of vertices $D = \{d_1, ..., d_n\}$, where every node is tuple $d_i = (c_i, r_i) \in \mathbb{R}^2 \times \mathbb{R}$, represents devices with wireless capabilities that are given as a pair consisting of their location in $\mathbb{R}^2$ (c_i) and their transmission range (r_i). These disks in $\mathbb{R}^2$ induce an intuitive directed graph by connecting two vertices $p = (c_i, r_i)$ and $q = (c_j, r_j)$ if q lies within the transmission range of p, or formally, $||c_j - c_i|| \leq r_i$. See Figure 1 for an example.

Peleg and Roditty [13] presented a method to construct a t-spanner for transmission graphs in metric spaces with constant doubling dimension using $O(\frac{n}{\epsilon^d} \log \Psi)$ edges where $\Psi = \frac{radius_{max}}{radius_{min}}$ is the ratio between the maximum and minimum radii. They later proved [12] that in this setting, it is not possible to guarantee a spanner whose size is independent of Ψ. In later papers, Kaplan et al. [9,10] showed a t-spanner for the Euclidean metric setting with $O(n)$ edges and a lower construction time of $\big(O(n(\log n + \log \Psi)$ or $n \log^5 n$ instead of $n \log n\big)$, thus reducing the running time and removing the dependence on the ratio Ψ.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

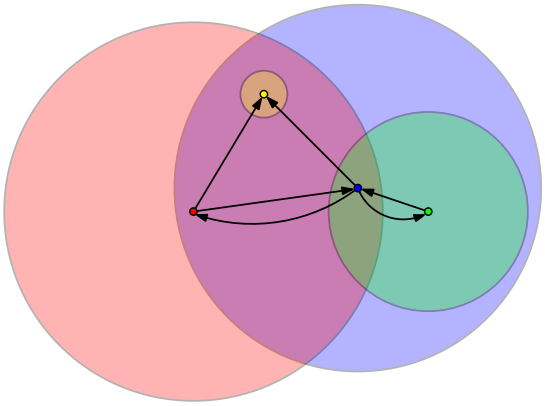


Figure 1 An example of a transmission graph.

The Path-Greedy Algorithm

The simple well-known greedy algorithm for constructing t-spanner was given by Althöfer et al. [1]. They proved that the algorithm can be used to achieve a t-spanner for arbitrary weighted graphs. For euclidean graphs, they prove that the algorithm guaranties a linear number of edges and a bounded degree, both depending on t. A weight bound of $O(wt(MST(P)))$, where P is a set of points in $\mathbb{R}^d, d \leq 3$ was given by Das et al. [4], and generalized for any dimension by Das and Narasimhan [5]. In their paper, Althöfer et al. also mentioned that the algorithm was independently proposed by Bern.

The algorithm itself is very simple, and it is a natural generalization of Kruskal's algorithm for finding an MST of a given graph, see Algorithm 1.

Algorithm 1: Path-Greedy

Input: A graph $G = (V, E)$, $1 < t \in \mathbb{R}^+$
Result: A t-spanner $G' = (V, E')$ for G
$E \leftarrow$ Sort all edges of G in non-decreasing order
$E' = \emptyset$
for $(u, v) \in E$ *(in sorted order)* **do**
  if $\delta_{G'}(u, v) > t \cdot \delta_G(u, v)$ **then**
    Add (u, v) to E'
return G'

Surprisingly, this simple algorithm gives both theoretical and experimental results that are better than many more complicated state-of-the art algorithms. Farshi and Gudmundsson [6] experimented with implementations of several well known algorithms and showed that the Path-Greedy algorithm out-preformed other algorithms even when the theoretical bounds were similar. As it can be seen from Tables II-V in [6], the Path-Greedy algorithm achieved significantly better results than other algorithms including *θ-Graph*, *WSPD* based spanners, *sink-spanner*, by building smaller and lighter spanners with a lower maxumium degree regardless of the distribution or number of input points.

From a theoretical point of view, Filtser and Solomon [7] have narrowed the gap between the experimental results showing the superiority of the Path-Greedy algorithm and the known upper bounds, by showing that the Path-Greedy is nearly optimal in many cases.

The simplicity and efficiency of the Path-Greedy algorithm and the naïve implementation with runtime $O(n^3 \log n)$ encouraged researchers to find faster algorithms that mimic or approximate it. An $O(n \log^2 n)$ time algorithm approxmating Path-Greedy was given by Das and Narasimhan [5], and was later improved to an $O(n \log n)$ time algorithm by Gudmundsson et al. [8], while Bar-On and Carmi [2] and Bose et al. [3] showed constructions of the Path-Greedy itself in $O(n^2 \log n)$.

Contribution

In this paper, and specifically in section 2, we provide a simple alternative to the fairly involved algorithms that were previously described, by proving in subsection 2.1 that the Path-Greedy algorithm is also applicable in the case of transmission graphs and provides a t-spanner with $O(\frac{n}{(t-1)^{d-1}})$ edges and in-degree $O(\frac{1}{(t-1)^{d-1}})$ for every real $t > 1$. We then prove in subsection 2.2 that the weight of the resulted spanner can be bounded by a function of the *radius-ratio*. For simplicity, we conduct our analysis in $\mathbb{R}^2$, however, all of the results extend naturally to $\mathbb{R}^d$ for $d > 2$ with the appropriate bounds of the d-dimensional Path-Greedy algorithm.

2 Path-Greedy Analysis

2.1 In-Degree Bound

We begin by showing that the in-degree of the vertices in the t-spanner created by the Path-Greedy algorithm on transmission graphs is bounded by a constant which is a function of the *stretch-factor* t, thus essentially proving the bound on the size of the t-spanner. In order to do so, we prove that if p is the sink of a directed edge $e = (q, p)$, then the angle between e and every other edge e' directed towards p is bigger than a constant depending on t.

▶ Lemma 1. Let $TG = (D, E)$ be a transmission graph, with $D = \{d_1, ..., d_n\}$ where $d_i = (c_i, r_i)$, $c_i \in \mathbb{R}^2$, is the center and $r_i \in \mathbb{R}$ is the radius of the disk, and $(d_i, d_j) \in E$ is a directed edge in the graph if $||c_i, c_j|| \leq r_i$. And let $G = (D, E')$ be the result of using the *Path-Greedy* algorithm on the input graph TG with $1 < t \in \mathbb{R}$. Then G is a t-spanner of TG, and $|E'| = O(\frac{n}{t-1})$.

Proof. It is rather simple to discern that G is a t-spanner of TG due to the exhaustive nature of the algorithm, and so we are left with proving the bound on the size of $|E'|$.

We provide a bound on the in-degree of any vertex in G by showing that any two edges with a point p as their destination, form an angle bigger than a certain constant depending on t. The t-spanning property of the resulted graph follows directly from the algorithm.

Let $e = (q, p), f = (r, p)$ be two edges in the set of spanner edges E', and let $\theta = \measuredangle qpr$. We assume w.l.o.g that $|qp| \geq |rp|$, and that $\overline{qp}$ is horizontal (see Figure 2), and also assume that $\theta \leq \frac{\pi}{4}$ since otherwise we are done. Let r' be the projection of r on $\overline{qp}$. Such a projection is possible and represents all possible cases due to our assumptions.

We now make the following observations:

- $|r'p| = |rp| \cdot \cos\theta$
- $|r'r| = |rp| \cdot \sin\theta$
- $|qr| < |r'r| + |qr'|$ (triangle inequality)

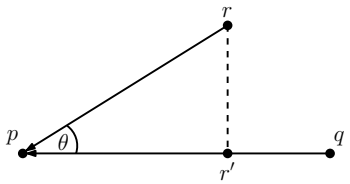


Figure 2 W.l.o.g $\overline{qp}$ is horizontal and at least as long as $\overline{rp}$

This gives us:

$$|qr| < |r'r| + |qr'| = |r'r| + (|qp| - |r'p|) = |rp| \cdot \sin\theta + |qp| - |rp| \cdot \cos\theta$$
$$= |qp| - |rp|(\cos\theta - \sin\theta),$$

which leads directly to:

$$\left(\frac{1}{\cos\theta - \sin\theta}\right)|qr| + |rp| < \left(\frac{1}{\cos\theta - \sin\theta}\right)|qp|.$$

Since we assume $\theta \leq \frac{\pi}{4}$, we get that $\overline{rq}$ is not the longest edge in Δqpr, and since we assume $\overline{rp} \leq \overline{qp}$ we get that $\overline{qr} \leq \overline{qp}$ as well, meaning that r is inside the disk centered at q. So, if $\left(\frac{1}{\cos\theta - \sin\theta}\right) \leq t$ we get that since the algorithm considered both $\overline{rp}$ and $\overline{qr}$ before $\overline{qp}$, the edge $\overline{qp}$ should not have been added to E' since at that point $\delta_G(q,r) \leq t \cdot |qr|$, which means that $\delta_G(q,r) + |rp| \leq t \cdot |qp|$, a contradiction to the choice of $\overline{qp}$ and $\overline{rp}$.

Thus, we get that the in-degree of any vertex $d \in D$ is at most $\frac{2\pi}{\theta}$. When t is big enough, it is clear that this degree is bounded by a constant, as the inequality $\frac{1}{\cos\theta - \sin\theta} \leq t$ is true for larger values of θ. But, as $t \to 1$, we get that $\theta \to 0$, meaning that the in-degree might be unbounded. We approximate $\frac{1}{\cos\theta - \sin\theta}$ using the Mclaurin series, and get that $\frac{1}{\cos\theta - \sin\theta} \approx 1 + \theta + O(\theta^2)$. It is now possible to see that as $t \to 1$ and after ommiting the negligible $O(\theta^2)$, we get that $\theta \leq t - 1$. So the in-degree of any vertex $d \in D$ is $O(\frac{1}{t-1})$, meaning that it is bounded by a constant depending on t, as reuiered.

◀

2.2 Weight Bound

In this section, we show a bound on the weight of *t-spanners* resulted by the Path-Greedy on transmission graphs. The bound is a function of the stretch factor t and a parameter called the *radius ratio*, which signifies the ratio between the largest and smallest radii amongst the given disks. More formally: let $G = (D, E')$ be the t-spanner computed by the Path-Greedy algorithm for the set D. Let $r_{max} = max\{r_i\}_{i=1}^n$, $r_{min} = min\{r_i\}_{i=1}^n$ and $\Psi = \frac{r_{max}}{r_{min}}$. Ψ is called the radius-ratio.

We show an upper bound on the total weight of the edges of G. That is, we show that $\sum_{e \in E'} |e|$ is

$$O((1 + \frac{1}{w}) \cdot \log n \cdot wt(MST(D))),$$

where $wt(MST(D))$ is the weight of the MST of the disk centers, and w is a constant that depends on the stretch factor t and the radius ratio.

A set of directed edges E satisfy the w-gap property if for any two directed edges (p,q) and (r,s) in E, we have that $|pr| > min(|pq|, |rs|)$. I.e., the sources of any two edges are relatively far apart with respect to the length of the shorter of the two edges. In this section,

we slightly change this definition and consider the distances between sinks instead of the distances between sources. Notice that the two definitions are equivalent by changing the direction of the edges.

In Lemma 2, we show that any two edges in E that form an angle of size at most θ, satisfy the w-gap property. That is, if $e, f \in E$ are 2 edges contained in the lines l_e and l_f respectively, and the angle between the two rays emenating from $l_e \cap l_f$ and that contain e and f is at most θ, then e and f satisfy the w-gap property, where w is a constant depending on θ, which in turn depends on t. Except for the additional constraint on w to be also smaller than $\frac{1}{\Psi}$, this lemma is similar to Lemma 15.1.1. in [11]. It is well known (Theorem 6.1.2 [11]) that for a set E of directed edges that satisfies the *w-gap property* we have that the total weight of E is less than $(1 + \frac{2}{w}) \cdot \log |P| \cdot wt(MST(P))$, where P is the set of the end-points of E.

▶ Lemma 2. Let $G = (D, E')$ be the t-spanner computed by the *Path-Greedy* algorithm for the set D. Let θ and w be two real numbers such that $0 < \theta < \frac{\pi}{4}$, $0 < w < \frac{\cos\theta - sin\theta}{2}$, $w < \frac{1}{\Psi}$, and $t \geq \frac{1}{\cos\theta - sin\theta - 2w}$. Let (q, p), (s, r) be two distinct directed edges in E, such that $\measuredangle(\overline{qp}, \overline{sr}) \leq \theta$. Then, (q, p) and (s, r) satisfy the *w-gap property.*

Proof. Assume w.l.o.g., that the *Path-Greedy* algorithm considers the edge (s, r) before it considers the edge (q, p), thus $|sr| \leq |qp|$. Assume towards contradiction that $|pr| \leq w|sr|$ (see figure 3 for an illustration). By *Lemma 6.4.1* from [11] we get:

1. $|qs| < |qp|$
2. $|rp| < |qp|$
3. $t|qs| + |sr| + t|rp| \leq t|qp|$.

By our choice of w, we have $w \leq \frac{1}{\Psi} = \frac{r_{min}}{r_{max}}$, and by our assumption we get, $|pr| \leq w|sr|$, thus $|pr| \leq \frac{r_{min}}{r_{max}}|sr| \leq r_{min}$. Therefore, we have that p is in the disk corresponding to r (since $|pr| \leq r_{min}$). Moreover, we have that s is in the disk corresponding to q, since $|qs| < |qp|$.

When the *Path-Greedy* algorithm considered the edge (q, p), the spanner already contained a t-spanning path from v to z, if z is in the disk corresponding to v and $|vz| < |pq|$. Therefore, when the *Path-Greedy* algorithm considered the edge (q, p) the spanner already contained a directed path from q to s of length at most $t|qs|$ and the edge (s, r) and a directed path from r to p of length at most $t|rp|$. This contradicts the fact that edge (q, p) has been added to the spanner, since by the lemma we have that $t|qs| + |sr| + t|rp| \leq t|qp|$. Thus, we conclude that $|pr| > w|sr|$. ◀

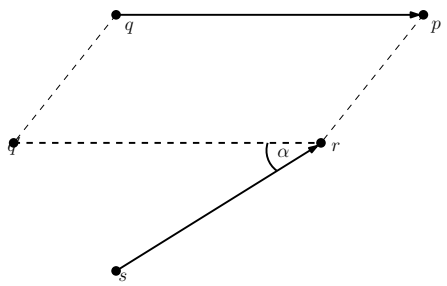


Figure 3 An illustration of the settings described in the proof above.

References

1 Ingo Althöfer, Gautam Das, David P. Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9:81–100, 1993. URL: https://doi.org/10.1007/BF02189308, doi:10.1007/BF02189308.

2 Gali Bar-On and Paz Carmi. \delta -greedy t-spanner. In *Algorithms and Data Structures - 15th International Symposium, WADS 2017, St. John's, NL, Canada, July 31 - August 2, 2017, Proceedings*, pages 85–96, 2017. URL: https://doi.org/10.1007/978-3-319-62127-2_8, doi:10.1007/978-3-319-62127-2_8.

3 Prosenjit Bose, Paz Carmi, Mohammad Farshi, Anil Maheshwari, and Michiel H. M. Smid. Computing the greedy spanner in near-quadratic time. *Algorithmica*, 58(3):711–729, 2010. URL: https://doi.org/10.1007/s00453-009-9293-4, doi:10.1007/s00453-009-9293-4.

4 Gautam Das, Paul J. Heffernan, and Giri Narasimhan. Optimally sparse spanners in 3-dimensional euclidean space. In *Proceedings of the Ninth Annual Symposium on Computational GeometrySan Diego, CA, USA, May 19-21, 1993*, pages 53–62, 1993. URL: https://doi.org/10.1145/160985.160998, doi:10.1145/160985.160998.

5 Gautam Das and Giri Narasimhan. A fast algorithm for constructing sparse euclidean spanners. *Int. J. Comput. Geometry Appl.*, 7(4):297–315, 1997. URL: https://doi.org/10.1142/S0218195997000193, doi:10.1142/S0218195997000193.

6 Mohammad Farshi and Joachim Gudmundsson. Experimental study of geometric t-spanners. *ACM Journal of Experimental Algorithmics*, 14, 2009. URL: https://doi.org/10.1145/1498698.1564499, doi:10.1145/1498698.1564499.

7 Arnold Filtser and Shay Solomon. The greedy spanner is existentially optimal. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing, PODC 2016, Chicago, IL, USA, July 25-28, 2016*, pages 9–17, 2016. URL: https://doi.org/10.1145/2933057.2933114, doi:10.1145/2933057.2933114.

8 Joachim Gudmundsson, Christos Levcopoulos, and Giri Narasimhan. Fast greedy algorithms for constructing sparse geometric spanners. *SIAM J. Comput.*, 31(5):1479–1500, 2002. URL: https://doi.org/10.1137/S0097539700382947, doi:10.1137/S0097539700382947.

9 Haim Kaplan, Wolfgang Mulzer, Liam Roditty, and Paul Seiferth. Spanners and reachability oracles for directed transmission graphs. In *31st International Symposium on Computational Geometry, SoCG 2015, June 22-25, 2015, Eindhoven, The Netherlands*, pages 156–170, 2015. URL: https://doi.org/10.4230/LIPIcs.SOCG.2015.156, doi:10.4230/LIPIcs.SOCG.2015.156.

10 Haim Kaplan, Wolfgang Mulzer, Liam Roditty, and Paul Seiferth. Spanners for directed transmission graphs. *SIAM J. Comput.*, 47(4):1585–1609, 2018. URL: https://doi.org/10.1137/16M1059692, doi:10.1137/16M1059692.

11 Giri Narasimhan and Michiel H. M. Smid. *Geometric spanner networks*. Cambridge University Press, 2007.

12 David Peleg and Liam Roditty. Relaxed spanners for directed disk graphs. *CoRR*, abs/0912.2815, 2009. URL: http://arxiv.org/abs/0912.2815, arXiv:0912.2815.

13 David Peleg and Liam Roditty. Localized spanner construction for ad hoc networks with variable transmission range. *TOSN*, 7(3):25:1–25:14, 2010. URL: https://doi.org/10.1145/1807048.1807054, doi:10.1145/1807048.1807054.

Diverse Voronoi Partitions of 1D Colored Points*

Marc van Kreveld[1], Bettina Speckmann[2], and Jérôme Urhausen[1]

1 **Dep. of Information and Computing Sciences, Utrecht University**
{m.j.vankreveld|j.e.urhausen}@uu.nl
2 **Dep. of Mathematics and Computer Science, TU Eindhoven**
b.speckmann@tue.nl

Abstract

We introduce the diverse Voronoi partition problem: for a given set of colored points and a number k, determine a set of k point sites so that the Voronoi cells of these sites contain as many colors as possible. We show that the problem is already NP-complete for points colored in four colors on a line, and give polynomial-time algorithms for a few special cases.

1 Introduction

Inspired by recent research on algorithmic fairness and similar concepts [1, 8], we study a problem involving diversity and representation. Imagine that a set of objects is represented by points in a space, and different classes are represented by colors. How can we represent all of the colored points by a smaller set of points, each of which represents many colors? We call the representing points *sites*. To formulate such problems, we need to specify when a site represents a colored point. The most obvious choice is by distance: a colored point will always be represented by the nearest site. A site is *diverse* if it represents many colors, in particular, the diversity score of a site is the number of colors it represents. There are different options to combine the diversity scores of sites into a global diversity score. We choose the sum measure. This leads to the following problem:

DIVERSE VORONOI PARTITION (DVP)
Input: Set P containing n points of h different colors and a number $k \in \mathbb{N}$.
Question: Determine a point set $S = \{s_1, \ldots, s_k\}$ that maximizes $\sum_{i=1}^{k} c_i$, where, in the Voronoi Diagram of S, c_i is the number of colors present in the cell of s_i.

For $i \in \{1, \ldots, h\}$, we define $P_i \subseteq P$ as the set of points of color i. One way to view the goal is to find a set of sites S, which represents each set P_i. The number of colors c_i in the cell of s_i is called the *score* of that cell, and $\sum c_i$ is the *score* of the Voronoi partition. The sites in Figure 1 for example have a score of 9.

A lot of related research exists in computational geometry. Firstly, various problems on colored points have been studied, see for example Kaneko and Kano [9]. In some cases the problems concern partitioning. For example, Dumitrescu and Pach [6] study partitioning multi-colored point sets into uni-colored subsets by convex cells, Majumder et al. [10] consider the same but partition with lines, and Bespamyatnikh et al. [4] consider partitioning a red-blue point set into convex cells so that each cell has the same number of red points and the same number of blue points (extensions were given in [2, 3, 5]).

Secondly, our problem is a type of clustering problem reminiscent of k-means clustering, where a representation of multiple points by a single point is also used, following a nearest

* Research on the topic of this paper was initiated at the 4th Workshop on Applied Geometric Algorithms (AGA 2018) in Langbroek, The Netherlands, supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 639.023.208. The first and third authors are supported by the NWO TOP grant no. 612.001.651.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

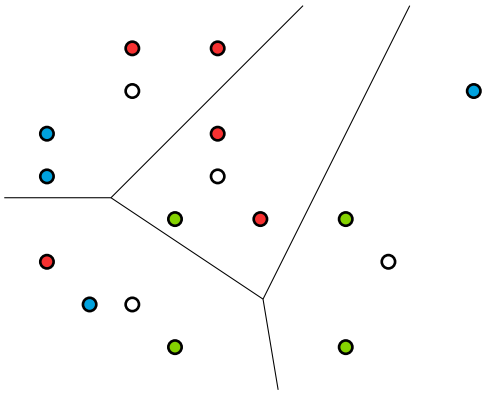


Figure 1 The sites (the white disks) induce a Voronoi diagram that determines which points a site represents. The bottom-left cell has a score of 3; the other three cells have a score of 2 each.

neighbor rule. While k-means clustering aims to minimize the sum of squared distances to the nearest representative, our version has colored points and aims to maximize the number of colors close to each representative.

Thirdly, our problem bears some resemblance to multi-criteria facility location [7] where multiple facilities are placed.

We restrict ourselves to points and sites on a line. We prove that even then, and with only four colors, the problem is NP-hard. In contrast, dropping the condition that the k cells be Voronoi intervals of points on the real line makes the problem solvable in polynomial time by dynamic programming, for any number of colors. We consider special cases where optimally diverse Voronoi partitions can be found in polynomial time. One such case is where a discrete set of m candidate locations is given on which the k sites of S should lie. A second case is where we have exactly n/h points per color and we want to know if an optimal solution with $k = n/h$ sites in S exists of score n; we call this a *perfect partition*.

2 Preliminaries

In this section we explore the difference between a Voronoi partition and any partition of a real line into intervals.

Assume a set P of n colored points is given. Assume that we roughly know where the interval ends should be to maximize the score, when placing k points. These interval ends, or *boundaries*, can be specified by intervals themselves, called *b-intervals*.

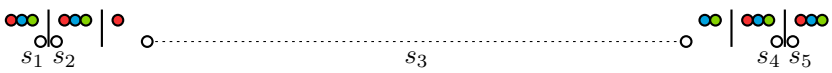


Figure 2 15 colored points that admit a perfect partition but not a perfect Voronoi partition.

Figure 2 shows a situation with 15 colored points, five in each of three colors. It is easy to see that a partition exists into five intervals such that the score is 15, the maximum possible. However, to find a Voronoi partition, we need to place five sites such that the boundaries are at the desired places, which are the b-intervals. A careful inspection shows that we cannot place five sites to realize a score of 15. The middle site must be sufficiently far to the left to ensure that the second boundary is correctly placed, and also sufficiently far to the right to

ensure that the third boundary is correctly placed. We cannot move the sites s_1, s_2, s_4, s_5 enough to realize this.

When we need to place sites $s_1, \ldots, s_k$ so that the Voronoi cell boundaries lie exactly in given b-intervals, we have a set of constraints to satisfy. Let b_i be the midpoint of s_i and s_{i+1}, for $1 \leq i < k$. Then $s_1 \leq b_1 \leq s_2 \leq \cdots b_{i-k} \leq s_k$, and $b_i = (s_i + s_{i+1})/2$.

To ensure that the Voronoi cell boundaries b_i lie inside their respective b-intervals, we use another $2k-2$ linear inequalities. Altogether, we have set up a system of $5k-5$ linear inequalities whose solution—if it exists—gives a Voronoi partition.

A specific case where this approach works is the following. Assume we have n points, equally many in each of the h colors. The problem is to place $k = n/h$ sites to answer the question whether the total score of n can be realized. We solve this problem as follows. We first check if a partition exists, which is only the case when the points come in n/h groups of points with h different colors. It is clear where the boundaries should be, and we can set up the system of linear inequalities in $O(n)$ time. Then we solve the linear program in time polynomial in $k = n/h$. If it is infeasible, there is no Voronoi partition that is perfect.

3 Algorithm for Discrete Site Locations

Here we assume that sites can only be placed at a finite set M of prespecified positions. We show that with dynamic programming, we can develop a polynomial-time algorithm.

The dynamic program works its way from left to right, using the fact that in a placement of the first i sites, we use an optimal placement of the first $i-1$ sites. However, the Voronoi boundaries between sites are determined by the last two sites, so our recurrence for an optimal solution has parameters for the last two sites. Furthermore, since we do not know the score for the i-th site, since its right boundary has not been fixed, we define maximum total score for the first $i-1$ sites for locations of the $(i-1)$-th and i-th sites.

Consider the function $f : \{2, \ldots, k\} \times M \times M \to \mathbb{N}$. For $i \in \{2, \ldots, k\}$ and $u, v \in M$, with $u < v$, let

$$f(i, u, v) = \max_{s_1, \ldots, s_{i-2} \in M} \Big(\sum_{j=1}^{i-1} c_j \mid s_1 < \cdots < s_i,\ s_{i-1} = u,\ s_i = v \Big)$$

be the best possible score of the first $i-1$ cells while fixing $s_{i-1} = u$ and $s_i = v$.

We further define the function $g : (M \cup \{-\infty\}) \times M \times (M \cup \{+\infty\}) \to \mathbb{N}$, where $g(a, b, c)$ is the number of colors present in the cell corresponding to the site b, where a and c are its left and right neighboring sites.

For $a, b \in M$, we have $f(2, a, b) = g(-\infty, a, b)$. Additionally we have the recursive definition:

$$\forall i \in \{3, \ldots, k\}, \forall b, c \in M, \text{ with } b < c,\ f(i, b, c) = \max_{a \in M, a < b} \big(f(i-1, a, b) + g(a, b, c) \big) . \quad (1)$$

In order to find the best sites overall we determine $\max_{a,b \in M, a<b}(f(k, a, b) + g(a, b, +\infty))$, which includes the number of colors in the k-th cell in the second term. The dynamic program uses a table of size $O(km^2)$, where $m = |M|$, and filling an entry requires optimizing over m choices. Since the function g can be evaluated trivially in $O(n)$ time, we immediately get an $O(km^3 n)$ time algorithm. The positions of the sites can then be obtained by backtracking as usual for dynamic programming algorithms.

We can improve the running time by preprocessing, to be able to evaluate g faster. Note that there are only $O(m^3)$ different values for g, whereas the function is evaluated $O(km^3)$

times. We will show how to precompute all values of g, so that a lookup during the main algorithm evaluates g in only $O(1)$ time:

1. For each $p \in P$, make a sorted list of the leftmost points right of it, one per color.
2. For each $b \in M$, make a sorted list of the boundaries $(b+c)/2$ with $c \in M$, $c > b$ in sorted order. This list has at most $m-1$ entries (the options for c).
3. For each $a, b \in M$, $a < b$, store the rightmost point $p(a,b) \in P$ left of $(a+b)/2$.

This information can be computed easily in $O(m^2+nh)$ time. Then, for each $a, b \in M$, we access $p(a,b)$ to get access to the sorted list of points with unique colors, stored with $p(a,b)$. We simultaneously scan the sorted list of colored points and the sorted list of boundaries $(b+c)/2$ to fill in all values $g(a,b,\cdot)$ in $O(m+h)$ time.

The running time of the whole algorithm becomes $O(km^3 + m^2h + nh)$.

4 Deciding DVP is NP-complete

We prove that the decision version of DVP is NP-complete. The decision version, referred to as D-DVP, receives an extra parameter z and asks if a diversity score of at least z can be realized with a Voronoi partition using k points.

We start by proving containment in NP. For a given instance of D-DVP, there are only exponentially many partitions into subsets, each defined by $k-1$ b-intervals, and we can test them all non-deterministically in polynomial time by linear programming to decide if they allow Voronoi partitions of k sites (see Section 2).

In order to prove hardness we reduce from Subset Sum.

> Subset Sum
> **Input:** A set $A = \{a_1, \ldots, a_r\}$ of integers and an integer b.
> **Question:** Is there a subset $A' \subseteq A$ such that $\sum_{a_j \in A'} a_j = b$?

Before starting with the reduction, we define a few terms. A point $p \in P$ is *represented* by a site $s \in S$ if s is the site closest to p. We say that for each color $i \in \{1, \ldots, h\}$, a point $p \in P_i$ is *scored* if each other point p' of the same color that is represented by the same site is to the right of p. That is, for each site only the leftmost point of each color that it represents is scored. It follows that finding the optimal S is equal to maximizing the number of scored points. A point is *unscored* if it is not scored. Our reduction uses only four colors, so we will define point sets P_1, P_2, P_3, P_4 from an instance of Subset Sum.

Let $0 < \delta \ll 1$ be a small real and let $0 < \varepsilon \ll \delta$ be an even smaller real. We can take $\delta = 1/n^2$ and $\varepsilon = 1/n^4$. Later we can multiply each coordinate by n^4 and thus obtain a set of integer positions with polynomial size. Let $a = \sum_{i=1}^{r} a_i$ be the sum of all input values.

We describe P from left to right. First, there is a starting gadget H of six points. Then we have a gadget D^j for each a_j, consisting of eight points (the gadgets can be in any order). Next, we have a subset sum gadget E of two points to represent b, and finally we have an ending gadget G of six points. $P = H \cup D^1 \cup \cdots \cup D^r \cup E \cup G$. Figure 3 shows an example for $A = \{1, 2\}$, so $P = H \cup D^1 \cup D^2 \cup E \cup G$.

To start the construction we define a set H of six points in two colors. We set $H_1 = \{-2\delta, -\delta, 0\} \subset P_1$ and $H_2 = \{-2\delta - \varepsilon, -\delta - \varepsilon, -\varepsilon\} \subset P_2$. The set H thus forms three groups of two points of different colors. We can only score all points in H with three sites s_{-2}, s_{-1}, s_0 if we have $-\delta < s_0 < 2\delta - 2\varepsilon$. So, in order to score all six points with three sites, the rightmost of those sites, s_0, needs to be close to zero.

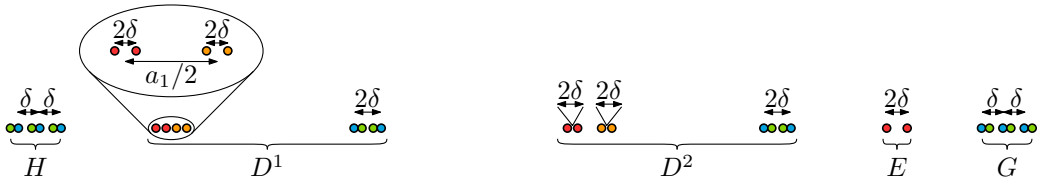


Figure 3 The reduction from Subset Sum to D-DVP illustrated. We have $a_1 = 1$, $a_2 = 2$ and $b = 2$. The points of P_1, P_2, P_3, and P_4 are blue, green, red, and yellow, respectively. Any two touching points are considered to be at ε distance. Note that δ is not drawn to scale, for clarity.

For each $a_j \in A$ we create a set D^j of eight points that will somehow encode whether a_j is chosen in the subset A' or not. Let $D^j = D^j_1 \cup D^j_2 \cup D^j_3 \cup D^j_4$, with $D^j_i \subset P_i$ (the points in D^j_i have color i). Let $D^j_1 = \{(4j-1)a-\delta, (4j-1)a+\delta\}$, $D^j_2 = \{(4j-1)a-\delta+\varepsilon, (4j-1)a+\delta-\varepsilon\}$, $D^j_3 = \{(4j-3)a-\delta, (4j-3)a+\delta\}$ and $D^j_4 = \{(4j-3)a+a_j/2-\delta, (4j-3)a+a_j/2+\delta\}$. The distances between D^j_3 and D^j_4 are roughly $a_j/2$.

We define a set $E \subset P$ that encodes that we want the subset sum to be b. We define $E \subset P_3$ with $E = \{4ra + (a-b)/2 - \delta, 4ra + (a-b)/2 + \delta\}$.

Finally we define a set G of six points to end the construction. It is similar to H. It can only be scored fully by three sites if the leftmost of the sites is close to $(4r+1)a$. We set $G_1 = \{(4r+1)a, (4r+1)a+\delta, (4r+1)a+2\delta\} \subset P_1$ and $G_2 = \{(4r+1)a+\varepsilon, (4r+1)a+\delta+\varepsilon, (4r+1)a+2\delta+\varepsilon\} \subset P_2$.

The instance of D-DVP has $n = 8r+14$ points and asks to place $k = 2r+6$ sites to realize a score of $z = 7r+14$, which can be achieved if and only if the corresponding Subset Sum instance has a solution. Intuitively, we want to use the sites to create boundaries that separate the first three pairs of points, the last three pairs of points, either D^j_3 or D^j_4, and also E. Separating D^j_3 corresponds to not choosing a_j in a subset and separating D^j_4 corresponds to choosing a_j. If we choose the correct a_j, the boundary between the last site chosen for D^r and the first site chosen for G will magically separate the points in E, due to its careful placement. Then, only one point of each D^j is not scored. An example can be seen in Figure 4.

The proof of correctness essentially shows that there are no other options: the Subset Sum instance has a solution if and only if the DVP instance can score $7r+14$. For details we refer to the full version.

References

1 M. Abbasi, S. A. Friedler, C. Scheidegger, and S. Venkatasubramanian. Fairness in representation: quantifying stereotyping as a representational harm. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 801–809. SIAM, 2019.

2 S. Bereg, P. Bose, and D. Kirkpatrick. Equitable subdivisions within polygonal regions. *Computational Geometry*, 34(1):20–27, 2006.

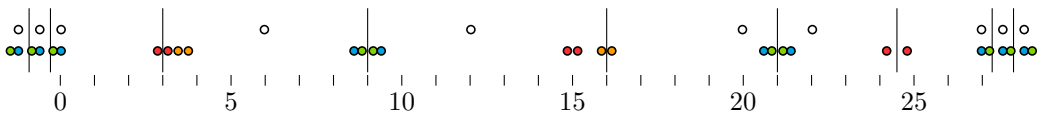


Figure 4 An induced D-DVP instance is illustrated. Here we have $a_1 = 1$, $a_2 = 2$ and $b = 2$. The sites and boundaries for a score of $7r+14$ are indicated.

3 S. Bereg, F. Hurtado, M. Kano, M. Korman, D. Lara, C. Seara, R. I. Silveira, J. Urrutia, and K. Verbeek. Balanced partitions of 3-colored geometric sets in the plane. *Discrete Applied Mathematics*, 181:21–32, 2015.

4 S. Bespamyatnikh, D. Kirkpatrick, and J. Snoeyink. Generalizing ham sandwich cuts to equitable subdivisions. *Discrete & Computational Geometry*, 24(4):605–622, 2000.

5 F. Chierichetti, R. Kumar, S. Lattanzi, and S. Vassilvitskii. Fair clustering through fairlets. In *Advances in Neural Information Processing Systems*, pages 5029–5037, 2017.

6 A. Dumitrescu and J. Pach. Partitioning colored point sets into monochromatic parts. *International Journal of Computational Geometry & Applications*, 12(05):401–412, 2002.

7 R. Z. Farahani, M. SteadieSeifi, and N. Asgari. Multiple criteria facility location problems: A survey. *Applied Mathematical Modelling*, 34(7):1689–1709, 2010.

8 V. Gupta, P. Nokhiz, C. D. Roy, and S. Venkatasubramanian. Equalizing recourse across groups. *arXiv preprint arXiv:1909.03166*, 2019.

9 A. Kaneko and M. Kano. Discrete geometry on red and blue points in the plane, a survey. In *Discrete and Computational Geometry, The Goodman-Pollack Festschrift*, pages 551–570. Springer, 2003.

10 S. Majumder, S. C. Nandy, and B. B. Bhattacharya. Separating multi-color points on a plane with fewest axis-parallel lines. *Fundamenta Informaticae*, 99(3):315–324, 2010.

Smoothed Analysis of Resource Augmentation

Jeff Erickson[1], **Ivor van der Hoog**[2], **and Tillmann Miltzow**[2]

1 **University of Illinois**
`jeffe@illinois.edu`
2 **Utrecht University**
`i.d.vanderhoog@uu.nl`
`t.mitzow@gmail.com`

Abstract

The predominant approach to find decent solutions for hard optimization problems is to compute an approximation. An alternative approach is resource augmentation (a form of problem relaxation), where you consider an optimal solution subject to slightly weaker problem constraints. This alternative approach has considerably less traction in theoretical computer science than approximation algorithms have. We study optimization problems with natural resource augmentations and show that the bit-precision of their optimal solution can be bounded using smoothed analysis of their augmentation. Our results imply that for realistic problem constraints, the optimal solution to a augmented version of a problem yields an optimal solution for the original problem. We hope our results help solidify the traction that resource augmentation has in theoretical computer science.

1 Introduction

This paper is an extended abstract from [14]. The RAM is a mathematical model of a computer which emulates how a computer can manipulate data. Within computational geometry, algorithms are often analyzed within the real RAM [15, 18, 23] where values with infinite precision can be stored and compared in constant space and time. By allowing these infinite precision computations, it becomes possible to verify geometric primitives in constant time, which simplifies the analysis of geometric algorithms. Mairson and Stolfi [19] point out that "without this assumption it is virtually impossible to prove the correctness of any geometric algorithms." The downside of the real RAM is that it neglects the bit-precision of the underlying algorithms. If an algorithm can be correctly executed with a limited bit-precision then the algorithm is called *robust*. Many classical examples in computational geometry are inherently nonrobust [23].

Often inputs which require excessive bit-precision are contrived and do not resemble *realistic* inputs. A natural way to theoretically model this is smoothed analysis, which interpolates *smoothly* between worst case analysis and average case analysis [28]. Practical inputs are constructed inherently with small amount of noise and random perturbation. This perturbation helps to show performance guarantees in terms of the input size and and the magnitude of the perturbation. By now smoothed analysis is well-established, for instance Spielman and Teng received the Gödel

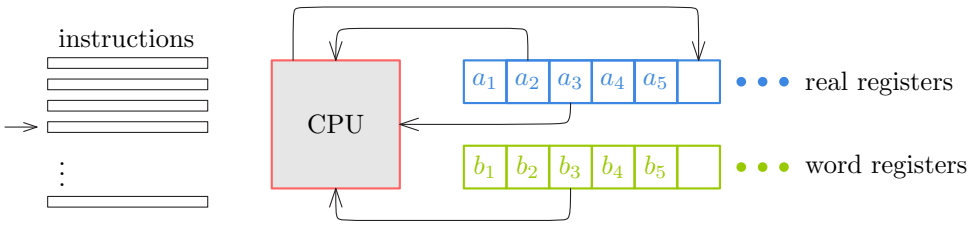


Figure 1. The dominant model in computational geometry is the real RAM. It consists of a central processing unit, which can operate on real and word registers in constant time, following a set of instructions.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

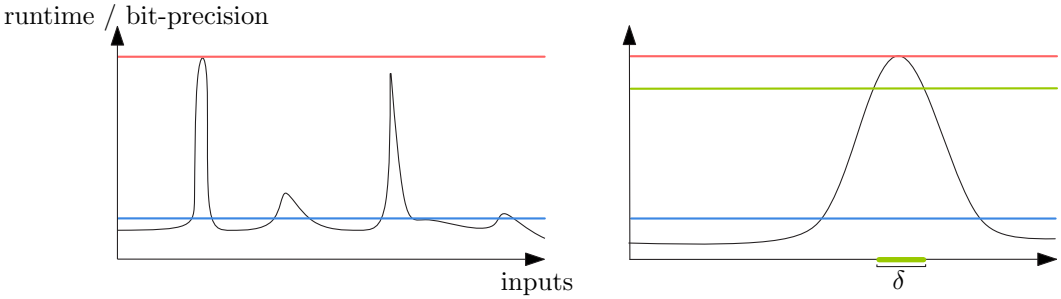


Figure 2. The x-axis symbolizes all inputs. The red line indicates the worse case running time or required bit-precision. The blue line indicates the average however, typical instances are not always average. Smoothed analysis considers the average of inputs near some worst instance (shown in green).

Prize for it. However, within computational geometry its application is limited to smoothed analysis of the bit-precision of the art gallery problem [10] and order type realisability [29], and smoothed analysis of the runtime of k-means clustering [3, 20], Euclidean TSP [12, 21], and partitioning algorithms for Euclidean functionals [5].

In this paper, we introduce a framework applicable to a wide class of real RAM optimization problems and show that under smoothed analysis of their resource augmentation, the optimal solution to these problems can be computed with logarithmic bit-precision. This is an extended abstract of Section 3 of [14].

Smoothed analysis. In *smoothed analysis*, the performance of an algorithm is studied for worst case input which is randomly perturbed by a magnitude of δ. Intuitively, smoothed analysis interpolates between average case and worst case analysis (Figure 2). The smaller δ, the closer we are to true worst case input. Correspondingly larger δ is closer to the average case analysis.

Formally, for smoothed analysis we fix some $\delta \in [0,1]$, which describes the *magnitude of perturbation*. In this paper, we consider an array $I = (a, b) \in \mathbb{R}^n \times \mathbb{Z}^m$ of n real numbers and m integers as the input of the optimization problem (for an extensive overview of the real RAM model that takes both real and integer input refer to [14], A.1). We assume that each real number is perturbed independently and that the integers stay as they are. We denote by $(\Omega_\delta, \mu_\delta)$ the probability space where each $x \in \Omega_\delta$ defines for each instance I a new 'perturbed' instance $I_x = (a + x, b)$. We denote by $\mathscr{C}(I_x)$ the cost of instance I_x (note that traditionally, smoothed analysis is applied to algorithms where the cost of an instance is the runtime required by that algorithm on the instance. In this paper, the cost is the required number of bits to represent the optimal solution of the instance). The smoothed expected cost of instance I equals:

$$\mathscr{C}_\delta(I) = \mathop{\mathbb{E}}_{x \in \Omega_\delta} \mathscr{C}(I_x) = \int_{\Omega_\delta} \mathscr{C}(I_x)\mu_\delta(x)\,\mathrm{d}x.$$

If we denote by Γ_n the set of all instances of size n, then the smoothed complexity equals:

$$\mathscr{C}_{\text{smooth}}(n, \delta) = \max_{I \in \Gamma_n} \mathop{\mathbb{E}}_{x \in \Omega_\delta} [\mathscr{C}(I_x)].$$

Intuitively smoothed analysis shows that not only do the majority of instances behave nicely, but actually in every neighborhood (bounded by the maximal perturbation δ) the majority of instances behave nicely. The smoothed complexity is measured in terms of n and δ. If the expected complexity is small in terms of $1/\delta$ then we have a theoretical verification of the hypothesis that worst case examples are well-spread. Following [8, 28] we perceive an algorithm to have polynomial cost in practice, if the expected cost of the algorithm is polynomial in n and in $1/\delta$.

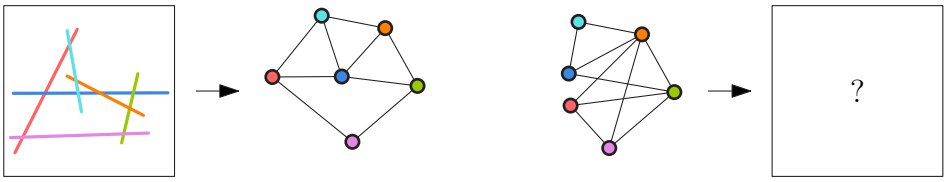


Figure 3. Left: given a set of segments S, they define a segment intersection graph G_S.
Right: given a graph G, is there a set of segments S' such that $G_{S'} = G$?

Spielman and Teng explain smoothed analysis by applying it to the simplex algorithm, which was known for a particularly good performance in practice that was seemingly impossible to verify theoretically [16]. Since the introduction of smoothed analysis, it has been applied to numerous problems. Most relevant for us is the recent smoothed analysis of the art gallery problem [10] and of order types [29]. Both papers deal with the required bit-precision needed in computations under slight perturbations. In the worst case, both problems need an exponential bit-precision, as both problems are complete for the existential theory of the reals.

The Existential Theory of the Reals. The required precision of an algorithm plays an important role if we want to show that a problem lies in the class NP. It is often easy to describe a potential witness to an NP-hard problem, but the bit-precision of the witness is unknown. A concrete example is the recognition of segment intersection graphs (Figure 3): given a graph, can we represent it as the intersection graph of segments? The canonical witness is the set of segments, but the required bit-precision is unclear. Matoušek [22] comments on this as follows:

Serious people seriously conjectured that the number of digits can be polynomially bounded—but it cannot.

Indeed, there are examples which require an exponential number of bits in any numerical representation. This *exponential bit-precision phenomenon* occurs not only for segment intersection graphs, but also for many other natural algorithmic problems [1,2,4,6,7,9,11,13,24–27]. It turns out that all of those algorithmic problems do not accidentally require exponential bit-precision, but are closely linked, as they are all complete for a certain complexity class called $\exists\mathbb{R}$. Thus either all of those problems belong to NP, or none of them do. Using our results on smoothed analysis, we show that for many $\exists\mathbb{R}$-hard optimization problems the exponential bit-precision phenomenon only occurs for near-degenerate input.

The complexity class $\exists\mathbb{R}$ can be defined as the set of decision problems that are polynomial-time equivalent to deciding if a formula of the *Existential Theory of the Reals* (ETR) is true or not. An ETR formula has the form:

$$\Psi = \exists x_1, \ldots, x_n \quad \Phi(x_1, \ldots, x_n),$$

where Φ is a well-formed sentence over the alphabet

$$\Sigma = \{0, 1, x_1, \ldots, +, \cdot, =, \leq, <, \wedge, \vee, \neg\}.$$

More specifically, Φ is quantifier-free and $x_1, \ldots, x_n$ are all variables of Φ. We say Ψ is true if and only if there are real numbers $x_1, \ldots, x_n \in \mathbb{R}$ such that $\Phi(x_1, \ldots, x_n)$ is true.

2 Results of Smoothed Analysis of Resource Augmentation

Under the resource augmentation of an algorithmic problem, you try to find an optimal solution to a problem formulation with weaker problem constraints. Resource augmentation does not compromise on optimality: the aim is to find an optimal solution to the newly augmented problem.

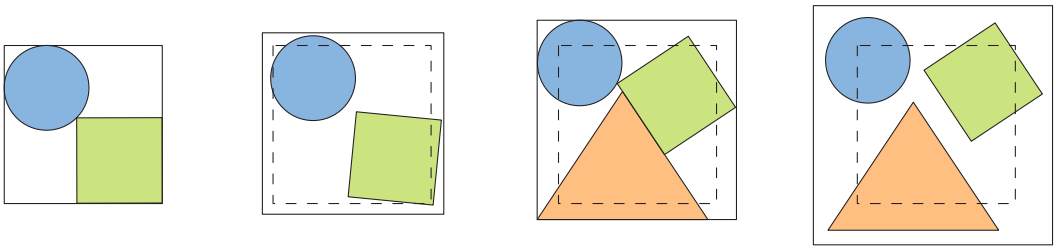


Figure 4. We augment the container from left to right. This extra space can lead to a better solution. If the optimal solution *value* does not change, the extra space allows for a solution with low bit-precision.

Using smoothed analysis, we argue that studying slight augmentations of algorithmic problems is justifiable for practical applications of the algorithm as we show that the problem conditions that make the problem hard are brittle.

An example of resource augmentation exists for the geometric packing problem (Figure 4) where an algorithm needs to pack a set of convex objects into a unit-size square container. To pack the optimal number of objects into this container is $\exists\mathbb{R}$-complete [2] and therefore a word RAM algorithm cannot hope to correctly find an optimal solution with limited time or memory. A resource augmentation algorithm looks to find a way to pack as many objects into a container C' which is larger by a factor $(1+\alpha)$ (α being the augmentation parameter). We apply smoothed analysis to resource augmentation problems, where we study these problems under a slight perturbation of such an augmentation parameter. We prove in [14] that the resource augmentation problems that we study have an optimal solution with expected logarithmic bit-precision.

► **Theorem 2.1.** *Let P be a resource augmentation problem that is monotonous, moderate and smoothable. Under perturbations of the augmentation of magnitude δ, the problem P has an optimal solution with an expected bit-precision of $O(\log(n/\delta))$.*

In the proof of this theorem (see full version) we argue about the solution space of the problem P and we define three natural properties of this solution space. The *monotonous property* demands that as we augment P more and more, the solution space only gains more candidate solutions. The *moderate property* demands that as we continuously augment the problem, we do not encounter more than a polynomial number of new optimums. In many hard optimization problems, the optimum is a value between 1 and n and the moderate property is then immediately implied. The *smoothable property* is the least intuitive of the three, it demands if you augment a problem P by ε, then it contains a solution which is optimal for the *original* problem and has a bit-precision of $O(\log(n/\varepsilon))$. It might appear as though the third property immediately implies the theorem, yet recall that we look for an optimal solution for the newly augmented problem. The other two properties, together with common bounds in probability theory, bound the expected bit-precision of an optimal solution to the perturbed problem.

Implications of Theorem 2.1. To illustrate the applicability of our findings, we give 3 corollaries:

The art gallery problem has been shown to be $\exists\mathbb{R}$-complete [1] which (assuming $\exists\mathbb{R} \neq NP$) prohibits a compact representation for all art gallery solutions. Yet our corollary states that under realistic conditions, the solution to the art gallery problem can be represented using logarithmic bit-precision. This result was already shown in [10], however with Theorem 2.1 this result can be re-proven by showing that the art gallery problem, with a resource augmentation of edge inflation is in fact monotonous, moderate and smoothable.

Recently Kostitsyna et al. showed that an optimal solution to the minimum-link path in a

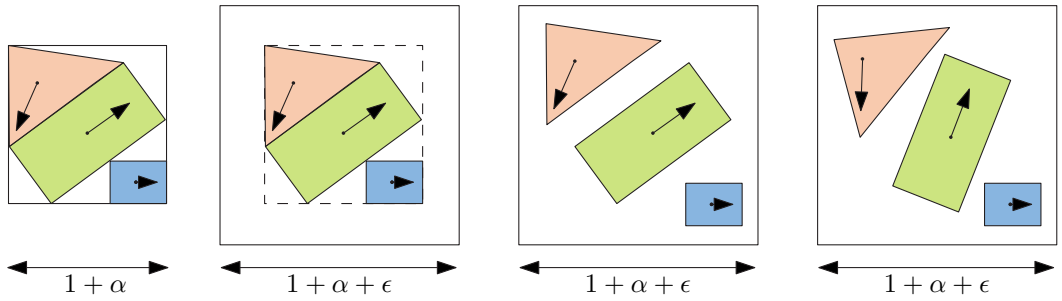


Figure 5. We increase a container of size $(1+\alpha)$ to size $(1+\alpha+\varepsilon)$. This extra space allows us to take the original solution, and space each object by a distance of $O(\varepsilon/n)$, which in turn allows us to find a more favourable rotation and / or translation for the object.

simple polygon has linear bit-precision in the worst case [17]. Just as the art gallery problem, this problem can be augmented by inflating the edges of the simple polygon. With a similar analysis, it then swiftly follows that the problem of computing the minimum-link path in a simple polygon is monotonous, moderate and smoothable.

The proof for $\exists\mathbb{R}$-completeness of the packing problems is in preparation [2] and just as for the art gallery problem this implies that the optimal solution to a packing problem cannot always be compactly represented. The packing problem has a natural resource augmentation, where one simply increases the size of the container. In the full version we show that the packing problem with container augmentation is monotonous, moderate and smoothable in the following way: if a container of size 1 can fit a collection I of items then a container of size $(1+\alpha)$ can also fit I and possibly more, thereby the monotonous property is trivial. If the input is a set of n elements that need to be packed in a container, then an optimal solution can pack at most k elements with $k \in [n]$. Therefore as we increase the container size continuously, there can be at most $O(n)$ new optimal solutions which implies the moderate property. The monotonous property is the hardest to show (Figure 5). In a solution to the packing problem, every object is rotated and translated and especially describing the rotation of an object is hard if you have to use limited bit-precision. In the full version we consider an optimal solution to a given container size, and show that if that container size increases by a value ε, then all the convex objects in the container can freely move and rotate a distance of $O(\varepsilon/n)$. This in turn, allows us to describe the translation and rotation of each object with a bit-precision of $O(\log(n/\varepsilon))$. Note that computing an embedding of an object with such a translation and rotation, might require more bit-precision.

▶ **Corollary 2.2.** *Under perturbations of the augmentation of magnitude δ, the following problems have an optimal solution with an expected bit-precision of $O(\log(n/\delta))$.*

- *the art gallery problem under perturbation of edge inflation [10].*
- *packing polygonal objects into a square container under perturbation of the container width.*
- *computing the minimum-link path in a simple polygon under perturbation of edge inflation.*

Limitations. We hope that Corollary 2.2 provides a convincing argument that our framework applies to a wide set of algorithmic problems that have a natural resource augmentation. Yet, our result is not wihout limiations: given an algorithmic problem, it is not clear *a priori* whether there is a way to augment resources such that it is both mathematically sound, satisfying, as well as practically plausible. For example, if we search for the smallest square container that fits a given set of items, the number of changes in the optimum is unbounded thus the moderate property does not hold.

Acknowledgments. The third author acknowledges the generous support of the NWO Veni grant EAGER.The second author acknowledges the support of the NWO grant 614.001.504.

References

1 Mikkel Abrahamsen, Anna Adamaszek, and Tillmann Miltzow. The art gallery problem is $\exists\mathbb{R}$-complete. In *STOC*, pages 65–73, 2018.

2 Mikkel Abrahamsen, Tillmann Miltzow, and Nadja Seiferth. A framework for $\exists\mathbb{R}$-completeness of two-dimensional packing problems. *in Preparation*, 2020.

3 David Arthur, Bodo Manthey, and Heiko Röglin. k-means has polynomial smoothed complexity. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 405–414. IEEE, 2009.

4 Daniel Bienstock. Some provably hard crossing number problems. *Discrete & Computational Geometry*, 6(3):443–459, 1991.

5 Markus Bläser, Bodo Manthey, and Raghavendra Rao. Smoothed analysis of partitioning algorithms for euclidean functionals. *Algorithmica*, 66(2):397–418, 2013.

6 Jean Cardinal, Stefan Felsner, Tillmann Miltzow, Casey Tompkins, and Birgit Vogtenhuber. Intersection graphs of rays and grounded segments. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 153–166. Springer, 2017.

7 Jean Cardinal and Udo Hoffmann. Recognition and complexity of point visibility graphs. *Discrete & Computational Geometry*, 57(1):164–178, 2017.

8 Daniel Dadush and Sophie Huiberts. A friendly smoothed analysis of the simplex method. In *STOC*, pages 390–403. ACM, 2018.

9 Michael G. Dobbins, Linda Kleist, Tillmann Miltzow, and Paweł Rzążewski. $\forall\exists\mathbb{R}$-completeness and area-universality. *ArXiv 1712.05142*, 2017.

10 Michael Gene Dobbins, Andreas Holmsen, and Tillmann Miltzow. Smoothed analysis of the art gallery problem. *CoRR*, abs/1811.01177, 2018. arXiv:1811.01177.

11 Michael Gene Dobbins, Andreas Holmsen, and Tillmann Miltzow. A universality theorem for nested polytopes. *arXiv*, 1908.02213, 2019.

12 Matthias Englert, Heiko Röglin, and Berthold Vöcking. Worst case and probabilistic analysis of the 2-opt algorithm for the tsp. In *SODA*, pages 1295–1304, 2007.

13 Jeff Erickson. Optimal curve straightening is $\exists\mathbb{R}$-complete. *arXiv:1908.09400*, 2019.

14 Jeff Erickson, Ivor van der Hoog, and Tillmann Miltzow. A framework for robust realistic geometric computations. *CoRR*, abs/1912.02278, 2019. arXiv:1912.02278.

15 Steven Fortune and Christopher Van Wyk. Efficient exact arithmetic for computational geometry. In *Proceedings of the ninth annual symposium on Computational geometry*, pages 163–172. ACM, 1993.

16 Victor Klee and George Minty. How good is the simplex algorithm. Technical report, Washington Univ. Seattle Dept. of Mathematics, 1970.

17 Irina Kostitsyna, Maarten Löffler, Valentin Polishchuk, and Frank Staals. On the complexity of minimum-link path problems. *Journal of Computational Geometry*, 8(2):80–108, 2017.

18 Chen Li, Sylvain Pion, and Chee-Keng Yap. Recent progress in exact geometric computation. *The Journal of Logic and Algebraic Programming*, 64(1):85–111, 2005.

19 Harry Mairson and Jorge Stolfi. Reporting and counting intersections between two sets of line segments. In *Theoretical Foundations of Computer Graphics and CAD*, pages 307–325. Springer, 1988.

20 Bodo Manthey and Heiko Röglin. Improved smoothed analysis of the k-means method. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pages 461–470. Society for Industrial and Applied Mathematics, 2009.

21 Bodo Manthey and Rianne Veenstra. Smoothed analysis of the 2-opt heuristic for the tsp: Polynomial bounds for gaussian noise. In *International Symposium on Algorithms and Computation*, pages 579–589. Springer, 2013.

22 Jiří Matoušek. Intersection graphs of segments and $\exists\mathbb{R}$. 2014. ArXiv 1406.2636.

23 David Salesin, Jorge Stolfi, and Leonidas Guibas. Epsilon geometry: building robust algorithms from imprecise computations. In *Proceedings of the fifth annual symposium on Computational geometry*, pages 208–217. ACM, 1989.

24 Marcus Schaefer. Complexity of some geometric and topological problems. In *Proceedings of the 17th International Symposium on Graph Drawing (GD 2009)*, Lecture Notes in Computer Science (LNCS), pages 334–344. Springer, 2009.

25 Marcus Schaefer. Realizability of graphs and linkages. In János Pach, editor, *Thirty Essays on Geometric Graph Theory*, chapter 23, pages 461–482. Springer-Verlag New York, 2013.

26 Marcus Schaefer and Daniel Štefankovič. Fixed points, Nash equilibria, and the existential theory of the reals. *Theory of Computing Systems*, 60(2):172–193, 2017.

27 Yaroslav Shitov. A universality theorem for nonnegative matrix factorizations. 2016, journal=ArXiv 1606.09068.

28 Daniel Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM (JACM)*, 51(3):385–463, 2004.

29 Ivor van der Hoog, Tillmann Miltzow, and Martijn van Schaik. Smoothed analysis of order types. *arXiv:1907.04645*, 2019.

The Multivariate Schwartz-Zippel Lemma

M. Levent Doğan[1], Alperen A. Ergür[2], Jake D. Mundo[3], and Elias Tsigaridas[4]

1 **Technische Universität Berlin, Institut für Mathematik, Strasse des 17. Juni 136, 10623, Berlin, Germany**
dogan@math.tu-berlin.de

2 **Carnegie Mellon University, School of Computer Science, 5000 Forbes Avenue, Pittsburgh, PA, 15213, USA**
aergur@cs.cmu.edu

3 **Inria Paris and Institut de Mathématiques de Jussieu-Paris Rive Gauche, Sorbonne Université and Paris Université, France**
elias.tsigaridas@inria.fr

4 **Swarthmore College, Department of Mathematics & Statistics, 500 College Avenue, Swarthmore, PA, 19081, USA**
jakedmundo@gmail.com

Abstract

Motivated by applications in combinatorial geometry, we consider the following question: Let $\lambda = (\lambda_1, \ldots, \lambda_m)$ be an m-partition of n, let $S_i \subseteq \mathbb{C}^{\lambda_i}$ be finite sets, and let $S \coloneqq S_1 \times S_2 \times \cdots \times S_m \subseteq \mathbb{C}^n$ be the multi-grid defined by S_i. If p is a degree d polynomial with n variables, how many zeros can p have on S?

We show that, except for a special family of polynomials –that we call λ-reducible– a natural generalization of the Schwartz-Zippel-DeMillo-Lipton Lemma holds. Moreover, we mention a symbolic algorithm to detect λ-reducibility for a special case. Along the way, we also present a multivariate generalization of Combinatorial Nullstellensatz, which might be of independent interest.

We refer the reader to the extended version of work [2] for further details, the missing proofs, and the presentation of the symbolic algorithm [2].

1 Introduction

Counting the number of zeros of a polynomial on a finite grid of points has been a subject of extensive research in combinatorics and theoretical computer science, see, for example, [6], [5]. The Schwartz-Zippel-DeMillo-Lipton Lemma is a well-known result in this line of research [7, 12, 3].

▶ **Theorem 1.1** (The Schwartz-Zippel-DeMillo-Lipton Lemma). *Let $\mathbb{F}$ be a field, let $S \subseteq \mathbb{F}$ be a finite set, and let $p \in \mathbb{F}[x_1, \ldots, x_n]$ be a polynomial of degree d. Suppose $|S| > d$ and let $S^n \coloneqq S \times S \times \cdots \times S$. Then we have*

$$|Z(p) \cap S^n| \leq d|S|^{n-1}$$

where $Z(p) = \{v \in \mathbb{F}^n \mid p(v) = 0\}$ is the zero set of p.

Alon [1] presents a similar statement for polynomials and grids. The result is known as Combinatorial Nullstellensatz:

▶ **Theorem 1.2** (Combinatorial Nullstellensatz). *Let $p \in \mathbb{F}[x_1, \ldots, x_n]$ with $\deg(p) = \sum_{i=1}^n d_i$ for some positive integers d_i, and assume that the coefficient of $\prod_{i=1}^n x_i^{d_i}$ in p is non-zero.*

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Let $S_i \subseteq \mathbb{F}$ be finite sets with $|S_i| > d_i$ and let $S \subseteq \mathbb{F}^n$ be defined by $S \coloneqq S_1 \times S_2 \times \cdots \times S_n$. Then there exists $v \in S$ such that

$$p(v) \neq 0.$$

We generalize the Schwartz-Zippel-DeMillo-Lipton Lemma and the Combinatorial Nullstellensatz to *multi-grids*.

▶ **Definition 1.3** (Algebraic Degree of a Finite Set)**.** Let $\mathbb{F}$ be a field, and let $S \subseteq \mathbb{F}^n$ be a finite set of points. Let $I(S) \subseteq \mathbb{F}[x_1, \ldots, x_n]$ be the ideal of polynomials vanishing on S. We define the algebraic degree, $\deg(S)$, of S to be

$$\deg(S) \coloneqq \min_{0 \neq p \in I(S)} \deg(p).$$

For the univariate case we have $S \subseteq \mathbb{F}$ and so $\deg(S) = |S|$. However, for $n \geq 2$, one can have arbitrarily large sets of degree one. For example, in $\mathbb{F}^n$ we can consider arbitrarily many points sampled from a hyperplane. The only general relation between the size and the degree of a set $S \subseteq \mathbb{F}^n$ seems to be the following inequality that we can prove using basic linear algebra:

$$|S| \geq \binom{\deg(S) - 1 + n}{n}.$$

Notation We call a sequence $\lambda = (\lambda_1, \ldots, \lambda_m)$ a partition of n into m parts if $n = \lambda_1 + \lambda_2 + \cdots + \lambda_m$. In this case, we write $\lambda \underset{m}{\vdash} n$. Given a partition $\lambda \underset{m}{\vdash} n$, we introduce the notation $\overline{x_1} = (x_1, x_2, \ldots, x_{\lambda_1}), \overline{x_2} = (x_{\lambda_1+1}, \ldots, x_{\lambda_1+\lambda_2})$, and so on. Given a polynomial $p \in \mathbb{F}[x_1, x_2, \ldots, x_n]$, we denote by $\deg_i(p)$, the degree of p with respect to the variables in $\overline{x_i}$. Given finite sets $S_1 \subseteq \mathbb{F}^{\lambda_1}, S_2 \subseteq \mathbb{F}^{\lambda_2}$ etc. we call the product

$$S_1 \times S_2 \times \cdots \times S_m \subseteq \mathbb{F}^n$$

the multi-grid defined by $S_1, S_2, \ldots, S_m$.

Now we can give our first result that forbids polynomials to vanish on multi-grids defined by finite sets of large degree:

▶ **Theorem 1.4.** *Let $\mathbb{F}$ be a field, $\lambda \underset{m}{\vdash} n$ be a partition of n into m parts, and let $p \in \mathbb{F}[x_1, \ldots, x_n]$ be a polynomial with $\deg(p) = \sum_{i=1}^{m} d_i$. Furthermore, suppose that there exists a non-zero term x^α in p with $\deg_i(x^\alpha) = d_i$ for all $i \in [m]$. Let $S_i \subseteq \mathbb{F}^{\lambda_i}$ be finite sets, and let the multi-grid $S \subseteq \mathbb{F}^n$ be defined by $S \coloneqq S_1 \times S_2 \times \cdots \times S_m$. If $\deg(S_i) > d_i$ for all $i \in [m]$, then there exists $v \in S$ such that*

$$p(v) \neq 0.$$

In the case that $\lambda = (1, 1, \ldots, 1) \vdash n$, we obtain Alon's Combinatorial Nullstellensatz. In this sense, the above theorem is a generalization of Combinatorial Nullstellensatz. However, for the applications in incidence geometry, we want to obtain quantitative statements. In other words, we want to give bounds in terms of $|S_i|$. The next observation shows that it is not always possible to obtain such bounds:

▶ **Observation 1.5.** *Let $g_1 \in \mathbb{C}[x_1, x_2] \setminus \mathbb{C}$ and $g_2 \in \mathbb{C}[x_3, x_4] \setminus \mathbb{C}$. For any $h_1, h_2 \in \mathbb{C}[x_1, x_2, x_3, x_4]$ and $p = g_1 h_1 + g_2 h_2$, the zero set $Z(p)$ of p contains $Z(g_1) \times Z(g_2)$ which is a positive dimensional variety. Thus, we can have arbitrarily large finite sets $S_1 \subseteq Z(g_1)$ and $S_2 \subseteq Z(g_2)$ such that*

$$S_1 \times S_2 \subseteq Z(p).$$

As the above observation shows, in order to have a quantitative statement on $|Z(p) \cap S|$, one has to assume certain compatibility conditions between p and S:

▶ **Definition 1.6** (λ-irreducibility)**.** Let $\lambda \underset{m}{\vdash} n$ be a partition of n into m parts, and let $V \subseteq \mathbb{C}^n$ be an algebraic set. We call V λ-reducible if there exist positive dimensional varieties $V_i \subseteq \mathbb{C}^{\lambda_i}$ for $i = 1, \ldots, m$ such that

$$V_1 \times V_2 \times \cdots \times V_m \subseteq V.$$

We call V λ-irreducible otherwise. If V is a hypersurface defined by a polynomial p, then we say p is λ-reducible (resp. λ-irreducible).

Mojarrad et al. [4] study the same problem for the special case of $\lambda = (2, 2)$. Their observation is that $(2, 2)$-reducible polynomials have a particularly concrete form. Namely, a polynomial $p \in \mathbb{C}[x_1, x_2, x_3, x_4]$ is λ-reducible if and only if there exist polynomials $g_1 \in \mathbb{C}[x_1, x_2] \setminus \mathbb{C}, g_2 \in \mathbb{C}[x_3, x_4] \setminus \mathbb{C}$ and $h_1, h_2 \in \mathbb{C}[x_1, x_2, x_3, x_4]$ such that

$$p = g_1 h_1 + g_2 h_2.$$

Mojarrad et al. ask for an algorithm to check whether a given polynomial $p \in \mathbb{C}[x_1, x_2, x_3, x_4]$ is $(2, 2)$-reducible. In the last section, we mention an algorithm which detects λ-reducibility for partitions of the form $\lambda = (2, 2, \ldots, 2)$. The full details of this algorithm can be found in [2]. For now, we turn our attention back to polynomials and multi-grids.

Now, using the concept of λ-reducibility, we can give a bound on the cardinality of multi-grids on which a λ-irreducible polynomial can vanish:

▶ **Theorem 1.7.** *Let $\lambda \underset{m}{\vdash} n$ be a partition of n into m parts, and let $p \in \mathbb{C}[x_1, \ldots, x_n]$ be a λ-irreducible polynomial such that $\deg_i(p) = d_i$. Let $S_i \subseteq \mathbb{C}^{\lambda_i}$ be finite sets, and set $S \coloneqq S_1 \times S_2 \times \cdots \times S_m$. If $|S_i| > d_i^{\lambda_i}$, then there exists $v \in S$ such that*

$$p(v) \neq 0.$$

We state our main theorem.

▶ **Theorem 1.8.** *Let $\lambda \underset{m}{\vdash} n$, let $S_i \subseteq \mathbb{C}^{\lambda_i}, i = 1, \ldots, m$ be finite sets, and let $S \coloneqq S_1 \times S_2 \times \cdots \times S_m$ be the multi-grid defined by S_i. Then for a λ-irreducible polynomial p of degree $d \geq 2$, and for every $\varepsilon > 0$ we have*

$$|Z(p) \cap S| = O_{n,\varepsilon}\left(d^5 \prod_{i=1}^{m} |S_i|^{1 - \frac{1}{\lambda_i + 1} + \varepsilon} + d^{2n^4} \sum_{i=1}^{m} \prod_{j \neq i} |S_j| \right)$$

where $O_{n,\varepsilon}$ notation only hides constants depending on n and ε.

2 Applications

As our first application, we recover the complex version of Szemerédi-Trotter Theorem [10] on the number of incidences between points and lines in real plane. To our knowledge, this version is first proven by Tóth except for the ε in the exponent [11].

▶ **Corollary 2.1.** *Let P be a set of points and L be a set of lines in the complex plane $\mathbb{C}^2$, and let $\mathcal{I}(P, L)$ denote the set of point-line incidences. Then, for all $\varepsilon > 0$, we have*

$$|\mathcal{I}(P, L)| = O(|P|^{\frac{2}{3} + \varepsilon} |L|^{\frac{2}{3} + \varepsilon} + |P| + |L|).$$

Proof. Let $p = x_1 + x_2x_3 + x_4$. It is straightforward to show that p is $(2,2)$-irreducible. Moreover, for a point $v = (z_1, z_2) \in \mathbb{C}^2$ and a line $l : x + by + c = 0$, we have $p \in l$ if and only if $p(z_1, z_2, b, c) = 0$. Theorem 1.8 yields the result. ◀

As a second application, we consider the following problem: Given a set P of n points in the complex plane $\mathbb{C}^2$, can we bound the number of pairs $((v_1, v_2), (w_1, w_2)) \in P \times P$ such that $(v_1 - w_1)^2 + (v_2 - w_2)^2 = 1$? In the real plane, the problem is known as the unit distance problem and a subquadratic upper bound is given by Spencer, Szemerédi and Trotter [9]. In the complex case, Solymosi and Tao reproduced the same bound except for the ε in the exponent. [8]. We obtain the same bound using Theorem 1.8.

▶ **Corollary 2.2.** *Let $P \subseteq \mathbb{C}^2$ be a finite set of points in the complex plane. Set $S = \{((v_1, v_2), (w_1, w_2)) \in P \times P \mid (v_1 - w_1)^2 + (v_2 - w_2)^2 = 1\}$. Then, for all $\varepsilon > 0$, we have*

$$|S| = O(|P|^{4/3+\varepsilon}).$$

Proof. Let $p = (x_1-y_1)^2+(x_2-y_2)^2-1 \in \mathbb{C}[x_1, x_2, y_1, y_2]$. We claim that no 3×3 multi-grid is contained in $Z(p)$. Given three distinct points $u = (u_1, u_2), v = (v_1, v_2), w = (w_1, w_2) \in \mathbb{C}^2$, the system

$$\begin{aligned} p(u_1, u_2, y_1, y_2) &= 0 \\ p(v_1, v_2, y_1, y_2) &= 0 \\ p(w_1, w_2, y_1, y_2) &= 0 \end{aligned}$$

has at most one solution: If u, v, w are on an affine (complex) line, a direct computation shows that the above system has no solution. If they are not on an affine (complex) line then taking differences between pairs of polynomials in the above system, we see that

$$\begin{bmatrix} y_1 & y_2 \end{bmatrix} \cdot \begin{bmatrix} v_1 - u_1 & w_1 - u_1 & w_1 - v_1 \\ v_2 - u_2 & w_2 - u_2 & w_2 - v_2 \end{bmatrix} = 0$$

and as u, v, w are affinely independent, we obtain $y = 0$. We deduce that p is $(2,2)$-irreducible and applying Theorem 1.8 to $\varepsilon/2$ yields the result. ◀

▶ **Theorem 2.3** (The Sparse Hypersurface-Point Incidence Theorem)**.** *Let $A = \{a_1, a_2, \ldots, a_k\}$ be a set of lattice points in $\mathbb{Z}^n_{\geq 0}$ with $\sum_{j=1}^n a_{ij} \leq d$ for all $1 \leq i \leq k$. We say a polynomial f is supported in A if*

$$f = \sum_{j=1}^{k} c_j x^{a_j}$$

where $c_j \in \mathbb{C}$ and $x^{a_j} = x_1^{a_{j1}} x_2^{a_{j2}} \ldots x_n^{a_{jn}}$. Let P be a set of points in $\mathbb{C}^n$, L be a set of polynomials supported in A, and let $\mathcal{I}(P, L)$ denote the set of incidences between P and L. We assume for any sets $U_1 \subseteq P$ with $|U_1| > d^n$ and $U_2 \subseteq L$ with $|U_2| > d^k$, the product $U_1 \times U_2$ is not included in $\mathcal{I}(P, L)$. Then, for all $\varepsilon > 0$, we have

$$|\mathcal{I}(P, L)| = O_{n,k,\varepsilon}(d^3|P|^{1-\frac{1}{n+1}+\varepsilon}|L|^{1-\frac{1}{k+1}+\varepsilon} + d^{(n+k)^4}(|P| + |L|)).$$

3 Algorithms

In [2], Section 3, we give an algorithm for the following problem:

Problem: Consider the partition $\lambda = (n, n, \ldots, n)$ of $n(m+1)$ into $m+1$ parts. Given a polynomial $p \in \mathbb{Q}[\overline{x_1}, \overline{x_2}, \ldots, \overline{x_{m+1}}]$, are there polynomials $g_i \in \mathbb{Q}[\overline{x_i}] \setminus \mathbb{Q}$ and $h_i \in \mathbb{Q}[\overline{x_1}, \overline{x_2}, \ldots, \overline{x_{m+1}}]$ such that

$$p = \sum_{i=1}^{m+1} g_i h_i \quad ? \tag{1}$$

Equivalently, are there hypersurfaces $\mathcal{V}_i \subseteq \mathbb{C}^n$ such that

$$\mathcal{V}_1 \times \mathcal{V}_2 \times \cdots \times \mathcal{V}_{m+1} \subseteq V(p) \subseteq \mathbb{C}^{n(m+1)},$$

where $\mathcal{V}_i = V(g_i)$ are the zero sets of the polynomials g_i for $i \in [m+1]$?

In the case that $\lambda = (2, 2, \ldots, 2)$, we can check λ-reducibility using the previous algorithm: If p is $(2, 2, \ldots, 2)$-reducible, then it contains a product

$$\mathcal{V}_1 \times \mathcal{V}_2 \times \cdots \times \mathcal{V}_m \subseteq Z(p)$$

where each $\mathcal{V}_i$ is an algebraic curve in $\mathbb{C}^2$. As $\mathcal{V}_i$ are hypersurfaces, they can be written as $\mathcal{V}_i = Z(g_i)$ for some polynomials $g_i \in \mathbb{C}[x_{2i}, x_{2i+1}] \setminus \mathbb{C}$ and thus p is contained in the ideal generated by $g_1, \ldots, g_m$. Conversely, if p is contained in the ideal generated by $(g_1, \ldots, g_m)$, then p contains the product $\mathcal{V}_1 \times \mathcal{V}_2 \times \cdots \times \mathcal{V}_m$ in its zero set. We deduce that a polynomial $p \in \mathbb{C}[x_1, x_2, \ldots, x_{2n}]$ is $(2, 2, \ldots, 2)$-reducible if and only if it is of the form

$$p(x_1, \ldots, x_{2n}) = \sum_{i=1}^{n} g_i(x_{2i}, x_{2i+1}) h_i(x_1, \ldots, x_{2n}),$$

for some $g_i \in \mathbb{C}[x_{2i}, x_{2i+1}] \setminus \mathbb{C}$ and $h_i \in \mathbb{C}[x_1, \ldots, x_{2n}]$. We can detect this property using our algorithm.

We leave the existence of an algorithm that detects λ-reducibility for general λ as an open problem.

Acknowledgements

ET is partially supported by ANR JCJC GALOP (ANR-17-CE40-0009), a public grant as part of the Investissement d'avenir project reference ANR-11-LABX-0056-LMH LabEx LMH (PGMO ALMA), and the PHC GRAPE.

References

1 N. M. Alon. Combinatorial Nullstellensatz. *Combinatorics Probability and Computing*, 8(1-2):7–29, 1 1999. doi:10.1017/S0963548398003411.

2 M. L. Doğan, A. A. Ergür, J. D. Mundo, and E. Tsigaridas. The multivariate schwartz-zippel lemma, 2019. arXiv:1910.01095.

3 R. J. Lipton. The curious history of the Schwartz-Zippel lemma. https://rjlipton.wordpress.com/2009/11/30/the-curious-history-of-the-schwartz-zippel-lemma/.

4 H. N. Mojarrad, T. Pham, C. Valculescu, and F. de Zeeuw. Schwartz-Zippel bounds for two-dimensional products. *Discrete Analysis*, 2018. URL: http://dx.doi.org/10.19086/da.2750, doi:10.19086/da.2750.

5 Orit E. Raz, Micha Sharir, and József Solymosi. Polynomials vanishing on grids: The Elekes-Rónyai problem revisited, 2014. arXiv:1401.7419.

6 N. Saxena. Progress on Polynomial Identity Testing - II, 2014. arXiv:1401.0976.

7 J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, October 1980. URL: https://doi.org/10.1145/322217.322225, doi:10.1145/322217.322225.

8 József Solymosi and Terence Tao. An incidence theorem in higher dimensions. *Discrete & Computational Geometry*, 48(2):255–280, 2012. URL: https://doi.org/10.1007/s00454-012-9420-x, doi:10.1007/s00454-012-9420-x.

9 J. Spencer, E. Szemerédi, and W.T. Trotter. *Unit distances in the Euclidean plane*, pages 294–304. Academic Press, 1984.

10 E. Szemerédi and W. T. Trotter. Extremal problems in discrete geometry. *Combinatorica*, 3(3):381–392, 1983. URL: https://doi.org/10.1007/BF02579194, doi:10.1007/BF02579194.

11 C. D. Tóth. The Szemerédi-Trotter theorem in the complex plane. *Combinatorica*, 35(1), Feb 2015. URL: http://dx.doi.org/10.1007/s00493-014-2686-2, doi:10.1007/s00493-014-2686-2.

12 R. Zippel. Probabilistic algorithms for sparse polynomials. In *Symbolic and Algebraic Computation*, pages 216–226, Berlin, Heidelberg, 1979. Springer Berlin Heidelberg.

Orthogonal Schematization with Minimum Homotopy Area

Bram Custers[1], Jeff Erickson[2], Irina Kostitsyna[1], Wouter Meulemans[1], Bettina Speckmann[1], and Kevin Verbeek[1]

1 **TU Eindhoven, the Netherlands**
[b.a.custers|i.kostitsyna|w.meulemans|b.speckmann|k.a.b.verbeek]@tue.nl
2 **University of Illinois at Urbana-Champaign, United States of America**
jeffe@illinois.edu

1 Introduction

Visualizing data in its geographic context is useful for exploration, analysis and communication. Thematic maps show data as layers on top of a base cartographic map. However, in many cases high accuracy of spatial locations is restrictive and only of secondary concern for visualization: the spatial dimension may be distorted to further emphasize the data itself. This process of simplifying beyond the needs of a target scale is called cartographic *schematization* [12], and is often accompanied by stylistic restrictions on the used geometric primitives. The prototypical example is the London Underground map with its iconic octilinear style.

We typically distinguish schematization types based on the object to be schematized and the geometry restrictions. Objects are typically either graphs (e.g. transit networks) or polygons (e.g. territorial outlines such as countries). Geometry restrictions are often formulated via a set of admissible orientations [2, 11, 10], with orthogonal, hexilinear and octilinear being the most common, using angles that are a multiple of 90, 60 or 45 degrees. Alternatives include curves, popularized by Roberts [13], such as Bézier curves [6, 15] or circular arcs [7, 14, 15, 16]. Here we focus on orthogonal schematization of polygons.

When schematizing polygons, there are two main quality criteria beyond using few geometric primitives [2]: **(1)** the result should be a simple polygon, if the input is simple; and **(2)** the result should resemble the input polygon, as the schematic often functions as a base map for thematic data. Requiring simplicity of the output often causes simplification and schematization problems to be NP-hard [1, 8, 9]. Moreover, the similarity measure used to capture resemblance has a large impact on the visual quality of the schematic. This schematic may also exhibit undesirable behavior when optimizing for a measure, even in the most restricted orthogonal case [2]. Both factors have led to various heuristics being developed to schematize polygons either enforcing simplicity [2] or permitting self-intersections [4].

Problem definition. Given a simple orthogonal polygon P with n vertices and an integer $k < n$, compute an orthogonal polygon S^* with at most k vertices such that $\sigma(P, S^*)$ is minimized. Here, σ denotes the minimal homotopy area [3] between the two polygons; see Section 2 for definitions. We may require S^* to be a simple polygon or allow self-intersections.

Contributions. In Section 3 we illustrate the differences in homotopy area between the simple and nonsimple case, and lower bound the number of intersections in a nonsimple optimal solution. In Section 4 we present a dynamic program to compute in $O(n^5k)$ time the optimal solution which may self-intersect. Details and full proofs will be available in the full version of this paper.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

2 Preliminaries

Ortho-polygons. We consider only orthogonal polygons and polylines, which we abbreviate to ortho-polygons and ortho-polylines. We use $P = \langle v_1, \ldots, v_n \rangle$ to denote a simple ortho-polygon with n edges. Edge e_i connects vertices v_i and v_{i+1}, where $v_{n+1} = v_1$. A schematization of P is an ortho-polygon S with at most k edges. The complexity of an ortho-polygon or ortho-polyline is its number of edges.

Homotopy area. Homotopy area [3], measures the similarity between curves. To define this measure, interpret polygons P and S as continuous functions mapping the unit interval $[0, 1]$ to $\mathbb{R}^2$ for an arbitrary fixed starting point on the polygon. A *homotopy* $H : [0, 1] \times [0, 1] \to \mathbb{R}^2$ between polygons P and S is defined as a continuous deformation from P to S over a time $t \in [0, 1]$ such that $H(a, t = 0) = P(a)$ and $H(b, t = 1) = S(b)$. Here, a and b are the parameters for the parameterization of P and S.

The homotopy area of H is defined as the total area that is swept by the deformation, with multiplicity. The minimal homotopy, H^*, between two curves is a homotopy with the smallest homotopy area; the minimal homotopy area, $\sigma(P, S)$, is then the area swept by H^*.

As shown by Chambers and Wang [3], the minimal homotopy between two *simple* curves can be decomposed into smaller *subhomotopies* that deform the curve in a consistent direction relative to the deforming curve. These subhomotopies are delimited by *anchorpoints*, which are a subset of the intersections between P and S. These anchorpoints are stationary in the minimal homotopy and must therefore occur in the same order along both curves (Observation 3.1 in [3]). The subhomotopies are minimal homotopies for the curves between the delimiting anchorpoints. Moreover, any minimal homotopy without (or between) anchorpoints is *sense preserving* (Lemma 3.2 in [3]): intuitively, during the morph between the two curves, a point either consistently moves locally to the left (or stands still) or consistently to the right (or stands still). A careful inspection of the proofs in [3] reveals that these results also hold for the case of self-intersecting curves, as the arguments are fully local.

The homotopy area of any subhomotopy is given by the areas of the cells of the arrangement induced by the affected subcurves, multiplied by their *winding number* (Lemma 4.3 in [3]). The winding number of a cell is intuitively defined as the (possibly negative) number of counterclockwise rotations one makes when standing at a point in the cell and following the boundary of the curve.

Formally, homotopy area requires the given curves to be at least C^1 continuous, for the derivatives to be well defined. This is not the case for polygons, but we may imagine each corner to be an infinitesimally small smooth curve instead; see also [5].

3 Comparing simple and nonsimple schematization

Simplicity. Is the optimal schematization S always simple if the input polygon P is simple? Unfortunately, this is not the case. Let P be an ortho-polygon with a ratio $\delta \ll 1$ between the longest and shortest edge length (see Figure 1). Let S and S' be the optimal simple and nonsimple schematization of P. In the worst case, $\sigma(P, S)/\sigma(P, S') = \Omega(1/\delta)$.

Self-intersections. The optimal nonsimple schematization S' can have $\Omega(k^2)$ self-intersections, asymptotically matching the trivial upper bound (see Figure 2). As length d can be arbitrarily large with respect to area A, S' smooths out the top linesFor κ repetitions of the top lines and κ repetitions of the comb-like structure below, we obtain $\Theta(\kappa^2)$ intersections. With $k = \Theta(\kappa)$ and $n = \Theta(\kappa)$, the lower bound follows.

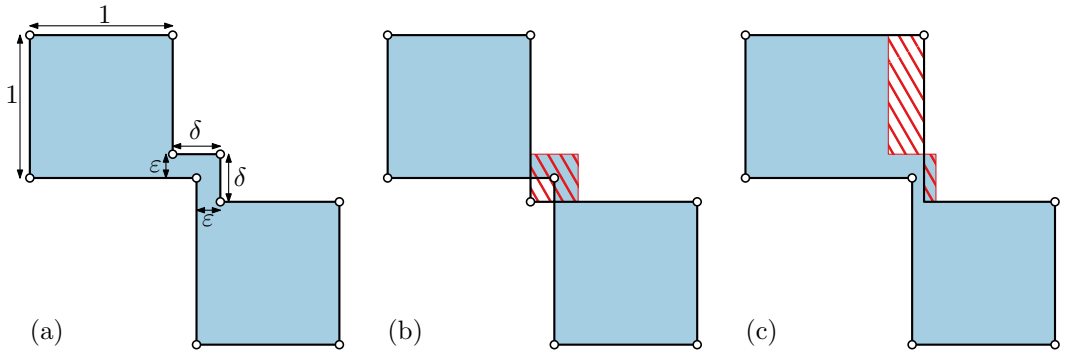


Figure 1 (a) Ortho-polygon with 10 edges. (b) Optimal nonsimple schematization S' with 8 edges self-intersects and has homotopy area δ^2. (c) Optimal simple schematization S has homotopy area greater than $(\delta - \varepsilon) \cdot (1 - \varepsilon) = \Omega(\delta)$ for small ε.

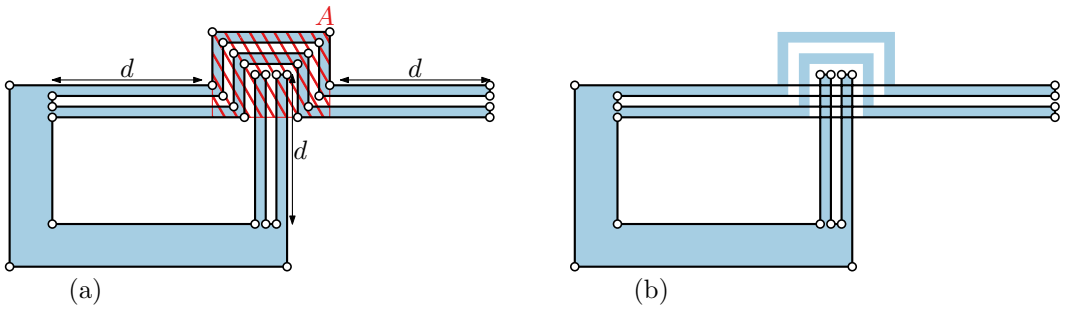


Figure 2 (a) Ortho-polygon with $n = 34$ and $\kappa = 4$. Length d is larger than area A; drawing shows d to be smaller for clarity of illustration. (b) Optimal nonsimple schematization S' for $k = 18$, with $\kappa^2 = \Omega(k^2)$ intersections.

4 Dynamic program for nonsimple S^*

We now compute an optimal schematization S^* for a given value of k that is allowed to self-intersect. Leveraging Chambers and Wang [3], we first prove that optimal solutions have a canonical form. Then, we discuss a dynamic program that establishes the following result.

▶ **Theorem 4.1.** *Given a simple ortho-polygon P with n vertices and an integer $k < n$, we can compute the schematization S^* with minimal homotopy area in $O(n^5 k)$ time, if S^* is allowed to self-intersect.*

4.1 Canonical form

▶ **Lemma 4.2.** *Each edge of S^* has at least one anchorpoint.*

Proof sketch. If an edge e has no anchorpoints, then it must be part of a single subhomotopy of the minimal homotopy. By sense preservation (Lemma 3.1 of [3]), moving e in the direction of sense preservation decreases the subhomotopy area. ◀

Hence, in an optimal solution, subhomotopies span at most two edges in S^* and the matching subcurve is simple. Thus, self-intersections of S^* can occur only between edges separated by anchorpoints. An *anchorsegment* is a nonempty subsegment of an edge e of S^* that coincides with an edge e' of P, such that all points on this anchorsegment are anchorpoints; the direction of e and e' must match, if this segment is more than a single point.

▶ **Lemma 4.3.** *Each edge of S^* has an anchorsegment of non-zero length.*

Proof. For a contradiction, assume we have an edge e in S^* that does not overlap an edge of P with the same direction. Let H^* denote the minimal homotopy between S^* and P. By Lemma 4.2, we know that e has one or more anchorpoints and thus we may consider the two or more subhomotopies $H_1, \ldots, H_m$ that involve part of e. Each H_i is between two simple curves and thus sense preserving (Lemma 3.2 of [3]).

Without loss of generality, assume e is horizontal. As the subcurves are simple, each subhomotopy area is the multiplication of area and winding number, summed over all faces in the arrangement (Lemma 4.1 and Lemma 4.3 of [3]). Consider moving e up or down: this move may change the homotopy area for each H_i, but cannot cause local intersections in the subcurves. Hence, the total change Δh in the area of all minimal subhomotopies is the change in face area with winding-number multiplicity. Either Δh is zero in both directions, or Δh is positive in one direction and negative in the other.

In the latter case, we find a contradiction with the optimality of S^*, so assume $\Delta h = 0$. We may freely move the edge up or down, until one of the considered arrangements changes; see Figure 3(a,b). At this moment e must overlap some edge e' of P, also considered in one of the original subhomotopies. Let S' denote the new schematization, with minimal homotopy H'. Now, either (a part of) this overlap is stationary in H' and thus an anchorsegment, or the entire edge still moves in H'. The former case implies an overlap in more than a single point: otherwise, Δh does not change. That is, the arrangement may have a face split, but these have the same winding numbers. In the latter case, we can continue shifting our edge e as Δh is zero (or positive in the same direction, contradicting optimality). See Figure 3(c) for an example.

Note that we cannot make an edge of S^* disappear during this motion. ◀

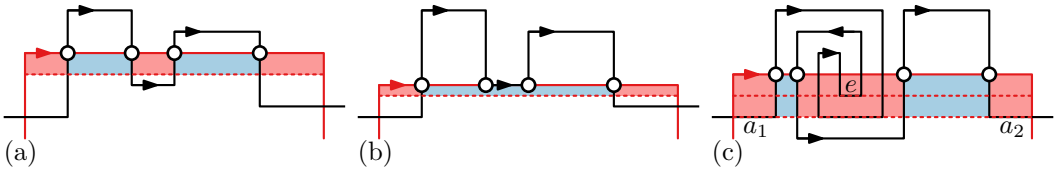


Figure 3 Moving an edge in the schematization. The black line shows part of the input polygon; the red line is part of the schematization. Dots indicate anchorpoints. (a) Moving the horizontal edge down to the dashed line decreases (red) and increases (blue) homotopy area for subhomotopies. The areas are exactly balanced: $\Delta h = 0$. (b) The edge coincides with an edge of the polygon, causing the arrangement of the middle subhomotopy to change: its cell collapsed. Moving the edge further down increases homotopy. (c) Moving the edge down causes the edge to overlap with input edge e. This edge moves in the underlying homotopy and thus can be ignored. Moving the edge further down results in anchor segments a_1 and a_2.

Lemma 4.3 implies a canonical form for S^*, in which each edge is anchored to an edge of P through its anchorsegment. Consequently, every vertex of S^* lies on the grid G induced by the edges of P. The *forward halfline* of an edge e_i is the halfline originating from v_i, overlapping e_i. Similarly, the *backward halfline* of an edge e_i originates from v_{i+1}, overlapping e_i. An edge e of S^* must start on the backward halfline of its anchored edge in P and end on the forward halfline, and the direction of e matches the direction of its anchored edge.

4.2 Dynamic Programming

First, we pick a midpoint of an edge of G that is on an edge of P to cut P into an ortho-polyline P'. In the DP, we select only vertices of G as intermediate vertices. Thus, an optimal schematization of P' with $k+1$ edges is a schematization of P with k edges. Testing all $O(n^2)$ midpoints yields the optimal schematization S^*, as any anchorsegment must contain such a midpoint. In the remainder P is an ortho-polyline, to be schematized with k edges.

Partial solutions. A *partial solution* is an ortho-polyline S where each edge has an anchorsegment. The first anchorsegment starts at v_1 on e_1. The last anchorsegment anchors to some edge e_i, but is not yet complete. Instead, the last vertex of S is a gridpoint $\gamma_{i,j}$ of G on the backward halfline of e_j and forward halfline of an e_i, $i < j$. If $\gamma_{i,j}$ lies on e_j, then the anchorsegment must contain $\gamma_{i,j}$; otherwise, the start of e_j must eventually be part of the anchorsegment. Between two anchorsegments of S we can compute the minimal subhomotopy area, even if the last is not yet complete. That is, consider two subsequent anchorsegments of S, anchored to edges e_i and e_j of P with $i < j$. The corresponding subhomotopy $\sigma_{i,j}$ can be computed purely from this information. This follows from one of four cases (see Figure 4) depending on the intersection $\gamma_{i,j}$. If the halflines do not intersect, we call such a pair incompatible and use $\sigma_{i,j} = \infty$. Subhomotopy areas $\sigma_{1,i}$ and $\sigma_{j,n+1}$ are independent of the choice of starting point v_1, though the corresponding γ-values do. As anchorsegments are directed and ordered, we need to consider only pairs of compatible edges ($\gamma_{i,j}$ exists).

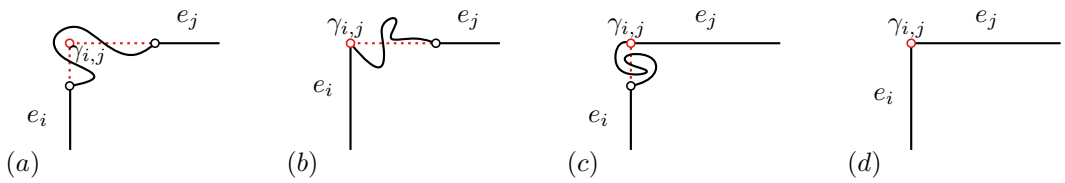


Figure 4 Four cases for compatible edges $e_{i'}$ and e_i to compute $\sigma_{i',i}$. The gridpoint $\gamma_{i',i}$ (red marker) can be outside both edges (a), on one of both edges (b,c), or on both edges (d).

However, we must be careful not to reverse an edge of S. Suppose S ends with edges anchored at $e_{i''}$, e_i and e_j, such that the first two edges are compatible, as well as the last two. If $\gamma_{i,j}$ is after $\gamma_{i'',i}$ in the direction of e_i, then the edge of S anchored on e_i is directed along e_i. However, if this is not the case, this edge of S is reversed and does not follow the canonical form. Hence, we do not need to consider these cases.

To decide whether we can extend partial solution S ending at e_i thus depends on where the last vertex of S is located with respect to e_i. We thus pair indices i with gridpoints g'. We call a pair (i, g') a *compatible predecessor* for (j, g) if $i < j$, $\gamma_{i,j}$ occurs after g' on the forward halfline of e_i, and $\gamma_{i,j}$ does not occur before g on the backward halfline of e_j; see Figure 5 for examples. Observe that the latter two conditions are independent of each other.

Dynamic program. We characterize a subproblem of our DP as $D[j, g, l]$: the minimal homotopy area for the optimal partial solution S with at most l edges, such that g lies on the backward halfline of e_j and the last vertex of S is not before g on this backward halfline. Figure 6 shows an example for such a partial solution. The main question is how to compute $D[j, g, l]$ based on "smaller" instances $D[i, g', l']$. We are effectively choosing anchorsegments one at a time. As these occur in order, smaller instances have $1 \le i < j$ and $l' = l - 1$.

We should consider compatible predecessors for (j, g) described by $D[i, g', l-1]$. If g'' is closer to $\gamma_{i,j}$ than g', g'' is less restrictive for the partial solution and hence $D[i, g'', l-1] \le$

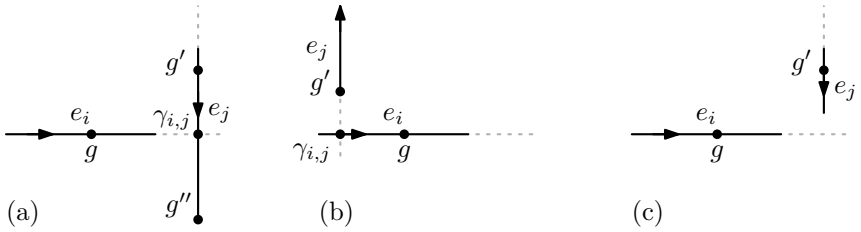


Figure 5 Examples of compatibility of predecessors. Black dots denote gridpoints, arrows give the directions of the edges. (a) (i, g) is a compatible predecessor of (j, g''), but not of (j, g'), since $\gamma_{i,j}$ occurs before g' in the backward direction of e_j. (b) (i, g) is not a compatible predecessor of (j, g') since $\gamma_{i,j}$ occurs after g on e_i in the backward direction. (c) (i, g) is not a compatible predecessor of any gridpoint on e_j, since $\gamma_{i,j}$ does not exist.

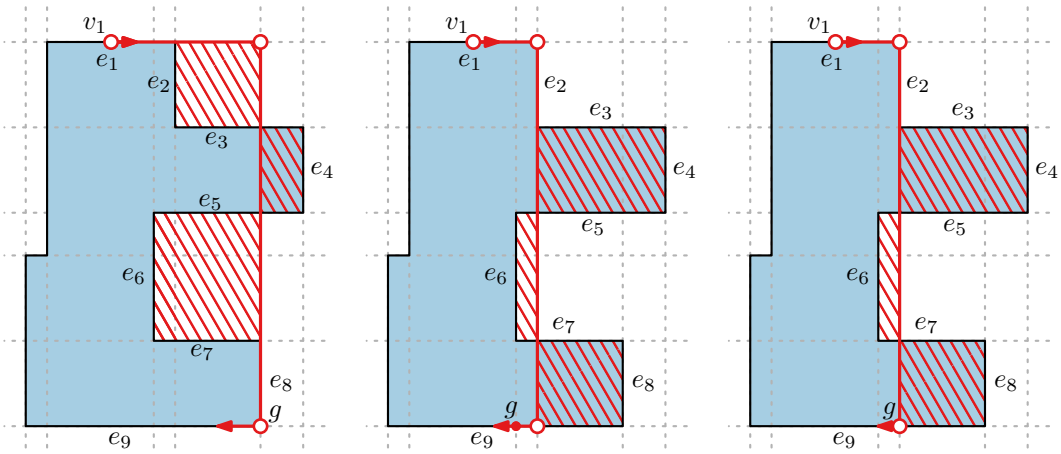


Figure 6 Partial solutions for input P (blue) with v_1 at the top. Shown in red are $D[9, g, 3]$ for various gridpoints g. Homotopy areas are given by the red, hatched area.

$D[i, g', l-1]$. Thus, the appropriate neighbor of $\gamma_{i,j}$ in G suffices if $\gamma_{i,j}$ lies on e_i, or the endpoint of e_i otherwise; we denote this least restrictive neighbor by $\lambda_{i,j}$.

We obtain the following dynamic program. If $j < l$, the schematization should have more edges than the input so far, thus we set $D[j, g, l]$ to ∞. If $l = j = 1$, we have not created a second anchorsegment and thus $D[j, g, l] = 0$. Otherwise, we set $D[j, g, l]$ to $\min_{1 \le i < j} \sigma_{i,j} + D[i, \lambda_{i,j}, l-1]$, testing all least restrictive compatible predecessors.

Running time. We first compute all $\sigma_{i,j}$ values in $O(n^4 \log n)$ total time using [3]: each pair of subcurves has $O(n)$ intersections. For the DP, computing $\lambda_{i,j}$ takes $O(1)$ time using G and we can lookup $\sigma_{i,j}$: computing all $O(n^2 k)$ cells takes $O(n^3 k)$ time. Running the DP for all $O(n^2)$ starting points thus takes $O(n^5 k)$ time.

References

1 Quirijn Bouts, Irina Kostitsyna, Marc van Kreveld, Wouter Meulemans, Willem Sonke, and Kevin Verbeek. Mapping Polygons to the Grid with Small Hausdorff and Fréchet Distance. In *Proceedings of the 24th Annual European Symposium on Algorithms*, volume 57 of *LIPIcs*, pages 22:1–22:16, 2016.

2 Kevin Buchin, Wouter Meulemans, André Van Renssen, and Bettina Speckmann. Area-preserving simplification and schematization of polygonal subdivisions. *ACM Transactions on Spatial Algorithms and Systems*, 2(1), April 2016.

3 Erin Wolf Chambers and Yusu Wang. Measuring similarity between curves on 2-manifolds via homotopy area. In *Proceedings of the 29th Annual Symposium on Computational Geometry*, pages 425–434. ACM, 2013.

4 Serafino Cicerone and Matteo Cermignani. Fast and simple approach for polygon schematization. In *Proceedings of the International Conference on Computational Science and Its Applications*, LNCS 7333, pages 267–279. Springer, 2012.

5 Brittany Fasy, Selcuk Karakoc, and Carola Wenk. On minimum area homotopies of normal curves in the plane. *arXiv preprint arXiv:1707.02251*, 2017.

6 Martin Fink, Herman Haverkort, Martin Nöllenburg, Maxwell Roberts, Julian Schuhmann, and Alexander Wolff. Drawing metro maps using Bézier curves. In *Proceedings of the International Symposium on Graph Drawing*, LNCS 7704, pages 463–474. Springer, 2012.

7 Martin Fink, Magnus Lechner, and Alexander Wolff. Concentric metro maps. In *Schematic Mapping Workshop*, 2014.

8 Leonidas Guibas, John Hershberger, Joseph Mitchell, and Jack Scott Snoeyink. Approximating polygons and subdivisions with minimum-link paths. *International Journal of Computational Geometry & Applications*, 3(04):383–415, 1993.

9 Maarten Löffler and Wouter Meulemans. Discretized approaches to schematization. In *Proceedings of the 29th Canadian Conference on Computational Geometry*, 2017.

10 Gabriele Neyer. Line simplification with restricted orientations. In *Proceedings of the 6th International Workshop on Algorithms and Data Structures*, page 13–24. Springer-Verlag, 1999.

11 Martin Nöllenburg and Alexander Wolff. Drawing and labeling high-quality metro maps by mixed-integer programming. *IEEE Transactions on Visualization and Computer Graphics*, 17(5):626–641, 2011.

12 Andreas Reimer. *Cartographic modelling for automated map generation.* PhD thesis, Technische Universiteit Eindhoven, 2015.

13 Maxwell Roberts. *Underground maps unravelled: Explorations in information design.* Self published, 2012.

14 Thomas van Dijk, Arthur van Goethem, Jan-Henrik Haunert, Wouter Meulemans, and Bettina Speckmann. Map schematization with circular arcs. In *Proceedings of the International Conference on Geographic Information Science*, LNCS 8728, pages 1–17. Springer, 2014.

15 Arthur van Goethem, Wouter Meulemans, Andreas Reimer, Herman Haverkort, and Bettina Speckmann. Topologically safe curved schematization. *The Cartographic Journal*, 50(3):276–285, 2013.

16 Arthur van Goethem, Wouter Meulemans, Bettina Speckmann, and Jo Wood. Exploring curved schematization of territorial outlines. *IEEE Transactions on Visualization and Computer Graphics*, 21(8):889–902, 2015.

Improved space bounds for Fréchet distance queries

Maike Buchin[1], Ivor van der Hoog[2], Tim Ophelders[3], Rodrigo I. Silveira[4], Lena Schlipf[5], and Frank Staals[2]

1 **Ruhr University Bochum**
`maike.buchin@rub.de`
2 **Utrecht University**
`[i.d.vanderhoog,f.staals]@uu.nl`
3 **Michigan State University**
`ophelder@egr.msu.edu`
4 **Universitat Politècnica de Catalunya**
`rodrigo.silveira@upc.edu`
5 **Universität Tübingen**
`schlipf@informatik.uni-tuebingen.de`

Abstract

We revisit a data structure from de Berg, Mehrabi and Ophelders that can store a polygonal curve P, such that for any directed horizontal query segment pq one can compute the Fréchet distance between P and pq in polylogarithmic time. We extend their analysis of the geometric constructions that can realize the Fréchet distance between P and pq and prove that in fact, their data structure only requires $O(n^{3/2})$ space, as opposed to the $O(n^2)$ space originally believed.

1 Introduction

Comparing the shapes of polygonal curves is an important task that arises in many contexts such as GIS applications [2, 4], protein classification [8], curve simplification [3], curve clustering [1] and even speech recognition [9]. Within computational geometry, there are two well studied distance measures for polygonal curves: the Hausdorff and the Fréchet distance. In this paper, we consider the problem of preprocessing a polygonal curve P of n edges in the plane, such that given a query segment pq traversed from p to q, the Fréchet distance between pq and P can be computed in sublinear time. The curve may self-intersect and pq may intersect P. For this version, the proofs have been omitted.

We give an overview of recent results that preprocess a polygonal chain P in order to compute the Fréchet distance between P and a query segment pq. Driemel and Har-Peled [6] studied how to process a polygonal chain P, such that given a query segment pq one can compute a $(1+\epsilon)$-approximation of the Fréchet distance between P and pq in $O(\epsilon^{-2} \log n \log\log n)$ time. Gudmundsson, Mirzanezhad, Mohades and Wenk [7] consider the Fréchet distance between polygonal curves where each curve contains only edges which are long when compared to the Fréchet distance between the two curves. A corollary of their result is that they can preprocess a curve P such that given a query segment pq one can compute the exact Fréchet distance between P and pq in $O(\log^2 n)$ time, provided that the length of pq and each edge of P is relatively large compared to this distance. Recently [5], de Berg, Mehrabi and Ophelders presented a paper in which they preprocess a curve P in $O(n \log^2 n)$ time, using $O(n^2)$ space, such that for any *horizontal* query segment pq one can compute the Fréchet distance between P and pq in $O(\log^2 n)$ time. In this paper we extend these results, by showing, via a more involved analysis, that the data structure by de Berg, Mehrabi and Ophelders requires only $O(n^{3/2})$ space.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

1.1 Preliminaries.

The *directed Hausdorff distance* is a distance measure between any two point sets. Let A and C be two point sets, we define the *directed Hausdorff distance* from A to C as $D_{\overrightarrow{H}}(A, C) := \sup_{a \in A} \inf_{c \in C} ||a - c||$. The Fréchet distance is a distance measure between any two curves, which is commonly explained with the following "leash" analogy: consider two curves in the plane P and Q where a person walks along curve P and a dog walks along curve Q, and neither of them is allowed to walk backwards. Then what is the minimum length that a leash between the person and the dog needs to have? Formally, we denote by $\alpha : [0, 1] \to P$ a (non-strict) monotone traversal of P in the time interval $[0, 1]$ and we denote by $\beta : [0, 1] \to Q$ an identical traversal of Q. The Fréchet distance between P and Q is the infimum over all choices of α and β, of the maximal distance realized during the traversal:

$$D_F(P, Q) = \inf_{\substack{\alpha:[0,1]\to P \\ \beta:[0,1]\to Q}} \left\{ \max_{t \in [0,1]} ||\alpha(t) - \beta(t)|| \right\}$$

De Berg, Mehrabi and Ophelders consider the scenario where P is a polygonal curve $P = (v_0, v_1, \ldots, v_n)$, where each vertex v_i is a point in the plane, and Q is a horizontal segment pq in the plane, at height y with p left of q. Their data structure uses the notion of *backward pairs*: any ordered pair of vertices (v_i, v_j) with $i \leq j$ in P form a *backward pair* if v_j lies further to the left than v_i. Note that a vertex of P can be be part of many backward pairs, even if its outgoing edge is pointed along the directed edge from p to q (Figure 1). There are $O(n^2)$ backward pairs in total. We denote the set of backward pairs by $\mathcal{B}(P)$. De Berg, Mehrabi and Ophelders note that a backward pair v_i, v_j has the following effect on the Fréchet distance. For the point on the query segment minimizing the distance to the farthest of v_i and v_j, that distance is a lower bound on the Fréchet distance. This point on the query segment lies either on the bisector B_{v_i,v_j} between v_i and v_j, or it is the point on the query segment closest to v_i or v_j. We define the distance function $F(y, v_i, v_j) := ||v_i - \ell_y \cap B_{v_i,v_j}||$ from v_i to the closest point that lies both on the bisector of v_i and v_j, and on the horizontal line ℓ_y at height y. De Berg, Mehrabi and Ophelders observe the following:

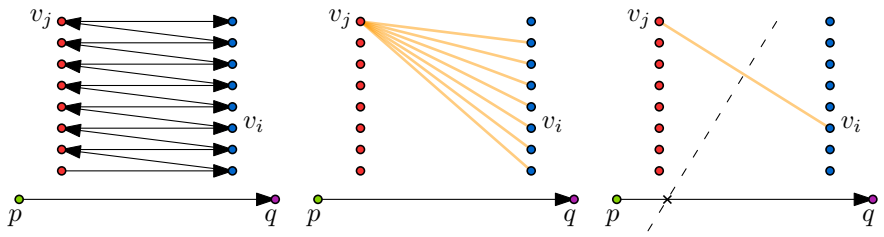


Figure 1 (Left) a polygonal curve that zigzags and a query segment from left to right. (Middle) The red vertex v_j forms a backward pair with all but one blue vertex. (Right) For a fixed backward pair (v_i, v_j), we consider the point of intersection between their bisector and pq (cross) and we are interested in the distance between that point and either v_i or v_j.

▶ Observation 1 (From [5]). For all $(v_i, v_j) \in \mathcal{B}(P)$, for any y, if the intersection between ℓ_y and B_{v_i,v_j} lies in the rectangle spanned by v_i and v_j, then $F(y, v_i, v_j)$ is a hyperbolic segment with absolute slope smaller than 1. Otherwise, it is a line with slope 1 or -1 (Fig 2).

They prove that for any backward pair (v_i, v_j), the value $F(y, v_i, v_j)$ is a lower bound for the Fréchet distance between pq and P if pq has height y. Note that the Fréchet distance is also lower-bounded by the distance between (1) p and the start of P, (2) q and the end of P

and (3) by the directed Hausdorff distance from P to pq. Specifically, de Berg, Mehrabi and Ophelders prove that the Fréchet distance is the maximum of any of these lower bounds:

$$D_F(P,pq) = \max\left\{||v_0 - p||, \quad ||v_n - q||, \quad d_{\overrightarrow{H}}(P,pq), \quad \max_{(v_i,v_j)\in\mathcal{B}(P)} F(y,v_i,v_j)\right\}$$

In this paper, we perform a deeper analysis on the data structure of de Berg, Mehrabi and Ophelders that computes these four terms, and give better bounds on its space complexity.

2 A data structure for horizontal segments

For any polygonal curve $P = (v_0, v_1, \ldots, v_n)$ and for any segment pq the distance $||v_0 - p||$ and $||v_n - q||$ can be computed in constant time. De Berg, Mehrabi and Ophelders present a linear space data structure that can compute the directed Hausdorff distance from P to pq in $O(\log^2 n)$ time. To compute the remaining component of the lower bound on the Fréchet distance they provide a data structure with $O(\log^2 n)$ query time whose space is linear in the number of backward pairs. They obtained this as follows: they consider the function $F(y, v_i, v_j)$ for every backward pair $(v_i, v_j) \in \mathcal{B}(P)$ and compute the upper envelope of all these functions. They argue that the upper envelope is linear in the number of backward pairs, which gives a quadratic upper bound on the space of the data structure.

We extend their analysis with the following observation: consider a vertex $v_i \in P$, the set of vertices $V_i := \{v' \mid (v_i, v') \in \mathcal{B}(P)\}$ and the upper envelope of all $\{F(y, v', v_i) \mid v' \in V_i\}$ (Figure 3). We define $L_i(y) := \min_{v_j \in V_i}(\ell_y \cap B_{v_i j, v_i})_x$ as the *left chain* of V_i and the function $F_i^L(y) := ||\ell_y \cap L_i(y) - v_i||$ as the distance from a point on L_i at height y to v_i. Note that the points $(L_i(y), y)$, that for simplicity we will denote L_i, correspond to the "left envelope" in the arrangement of bisectors. We use the term *chain* to avoid confusion with the upper envelope of the distances functions F_i^L, which we denote by $F^L(y) := \max_i\{F_i^L(y)\}$. Analogously, we define the right chain R_i, its corresponding distance function F_i^R, and the upper envelope $F^R(y) = \max_i F_i^R(y)$. It then follows that $F(y) = \max\{F^L(y), F^R(y)\}$ and thus:

▶ Observation 2. If the complexity of $F^L(y)$ and $F^R(y)$ are both upper bounded by $O(n^{3/2})$ then the complexity of $F(y)$ is upper bound by $O(n^{3/2})$.

In the remainder of this section, we bound the complexity of $F^L(y)$, the proof for $F^R(y)$ is analogous. We denote by V_i^* the subset of V_i, such that $v' \in V_i^*$ if and only if a piece

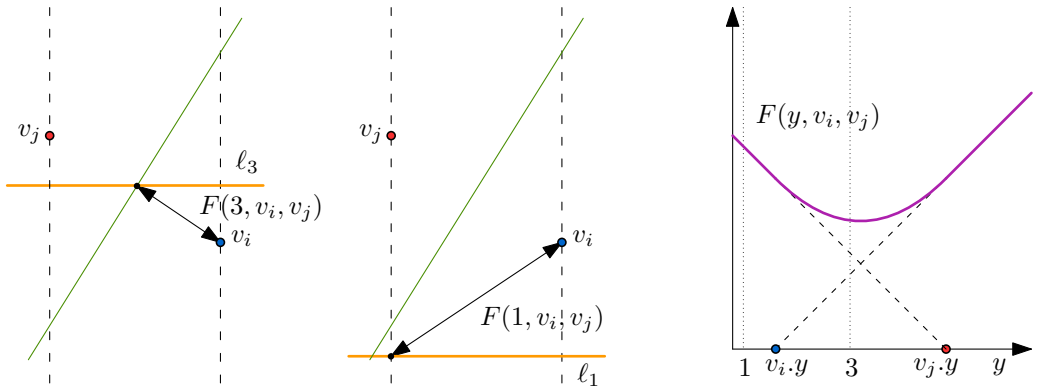


Figure 2 (Left) The backward pair (v_i, v_j), a line at height 3 and the distance $F(3, v_i, v_j)$. (Middle) A line at height 1 and the distance $F(1, v_i, v_j)$. (Right) The function $F(y, v_i, v_j)$.

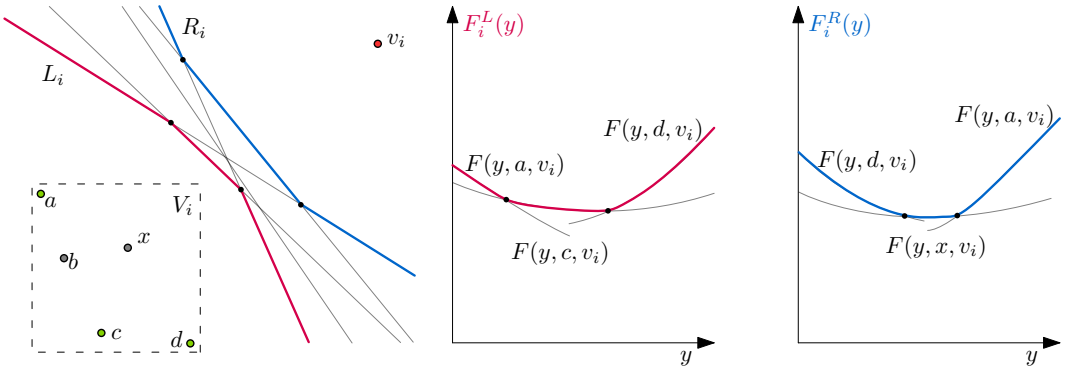


Figure 3 (Left) A point v_i and the set of vertices V_i that form a backward pair with v_i, together with the left and the right chains; vertices of V_i^* in green. (Middle) The envelope $F_i^L(y)$ is a piecewise curve with three pieces. (Right) The envelope $F_i^R(y)$ is also a piecewise curve with three pieces.

of the bisector B_{v',v_i} appears on L_i, and the distance function F_i^L is maximal there, i.e., $F_i^L(y) = F^L(y)$. The proofs of the following observations are deferred to the appendix.

▶ **Lemma 2.1.** *For any vertex v_i, the vertices V_i^* lie in convex position and the left chain L_i is a convex chain where the clockwise ordering of the bisectors on L_i is identical to the clockwise ordering of the vertices of V_i^*.*

▶ **Lemma 2.2.** *Let $v \in V_i^* \cap V_j^*$ be a point that forms a backward pair with both v_i and v_j. The bisectors B_{v,v_i} and B_{v,v_j} intersect B_{v_i,v_j} at a single point.*

▶ **Lemma 2.3.** *For any v_i, v_j, the chains L_i and L_j intersect B_{v_i,v_j} in a common point c. Moreover, the bisectors that intersect in c correspond to the same point $v \in V_i^* \cap V_j^*$.*

▶ **Corollary 2.4.** *For any v_i, v_j, for any horizontal line ℓ_y of height y, the line ℓ_y intersects L_i and L_j on the same side of the bisector B_{v_i,v_j}.*

▶ **Lemma 2.5.** *(Illustrated by Figure 4) For any two vertices v_i, v_j consider their Voronoi diagram. If the set $V_i^* \cap V_j^*$ contains at least four elements, then there is an edge (which is not a halfline) on L_i that is entirely contained in the Voronoi cell of v_i, or there is an edge (which is not a halfline) on L_j that is entirely contained in the Voronoi cell of v_j.*

Recall that the upper envelope $F^L(y)$ begins and ends with halflines of slope -1/1. All other bisectors generate at most one hyperbolic segment on $F^L(y)$. In the following lemma, we consider all pairs $v_i, v_j \in P$ that do not participate in these two halflines.

▶ **Lemma 2.6.** *For any v_i, v_j that do not participate in a halfline of $F^L(y)$, the set $V_i^* \cap V_j^*$ contains at most three vertices.*

Proof. Suppose for the sake of contradiction that $V_i^* \cap V_j^*$ contains four vertices (a, b, c, d) in counter-clockwise order, with v_i below the bisector B_{v_i,v_j}. By Lemma 2.5, we can assume without loss of generality that B_{a,v_i} and B_{a,v_j} are entirely contained in the Voronoi cell of v_i. Because of the assumption of the lemma, and because $a \in V_i^*$, the bisector B_{a,v_i} generates a hyperbolic segment on $F^L(y)$. Next we prove that this hyperbolic segment cannot appear on the upper envelope of $\{F_i^L(y), F_j^L(y)\}$ which contradicts the assumption that a is in V_i^*.

We denote the point of intersection between ℓ_y and B_{a,v_i} by S_i. Consider the point of intersection between ℓ_y and L_j and denote it by S_j. Observe that S_j must also lie in the

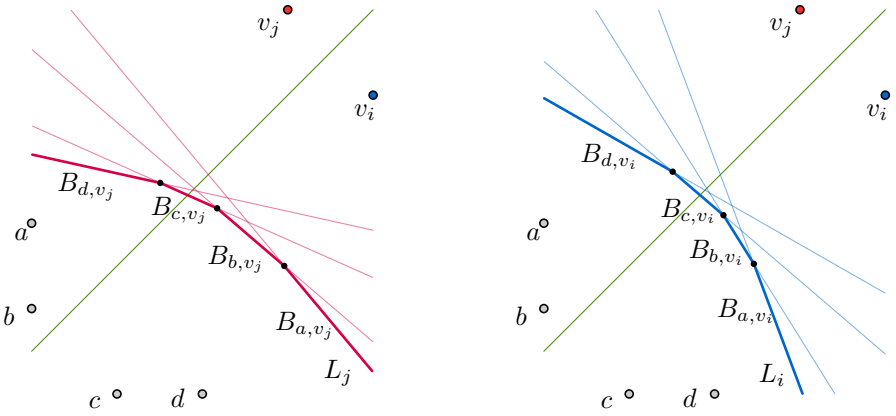


Figure 4 Two vertices v_i and v_j and their Voronoi diagram. We drew four points $a, b, c, d \in V_i^* \cap V_j^*$. Left we see the bisectors between these points and V_j^*, and their left chain L_j. Right we see the bisectors between these points and V_i^*, and their left chain L_i.

Voronoi cell of v_i (Corollary 2.4). We split the proof in three cases depending on S_i. We prove that the first case cannot exists. The remaining cases are illustrated in Figure 5.

Case S_j lies left of S_i. Following Observation 1 and the assumption that v_i and v_j are both the rightmost points of the backward pairs, S_i must lie left of v_i and S_j must lie left of v_j. Thus the distance $\|v_i - S_i\|$ is smaller than the distance $\|v_i - S_j\|$. But since S_j lies in the Voronoi cell of v_i, the distance $\|v_i - S_j\|$ is smaller than the distance $\|v_j - S_j\|$. This implies that the distance $\|v_i - S_i\|$ is smaller than the distance $\|v_j - S_j\|$ which contradicts the assumption that for this y-coordinate $F(y, v_i, a)$ lies on $F_i^L(y)$.

Case S_j lies right of S_i and on the bisector B_{a,v_j}. We note, that since S_i and S_j lie on bisectors, that $\|v_i - S_i\| = \|a - S_i\|$ and $\|v_j - S_j\| = \|a - S_j\|$ and thus per assumption $\|a - S_j\| > \|a - S_i\|$. However, S_j and S_i both lie to the right of a and they have the same y-coordinate. Since S_j lies further to the right than S_i, we know that $\|a - S_i\| < \|a - S_j\|$ which is a contradiction.

Case S_j lies right of S_i and not on the bisector B_{a,v_j}. We say that S_j lies on B_{b,v_j} but the argument works for any bisector further than a in the ordering, the argument is illustrated by Figure 6. We pinpoint the location of the point a with three observations:

1. The bisector B_{a,v_i} is the last bisector in the clockwise ordering of the left chain L_i, therefore the point a must lie above the line through b and v_i (Lemma 2.1).

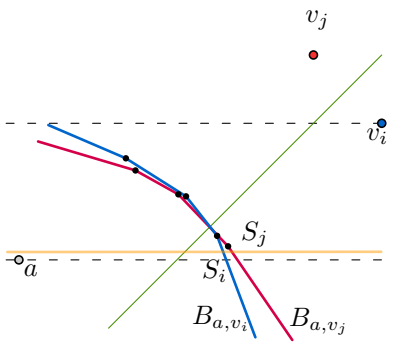


Figure 5 A horizontal line at a y-coordinate in yellow. The points of intersection S_j lies right of S_i and the intersection points originate from the bisectors B_{a,v_i} and B_{a,v_j}.

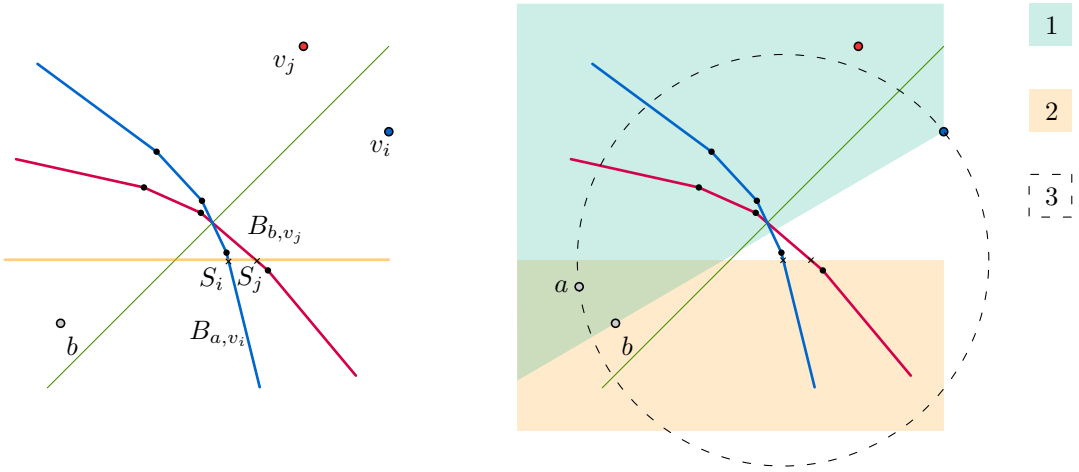


Figure 6 (left) Case S_j left of S_i *and* S_j lies on B_{b,v_j}. (right) Three regions where a can lie.

2. The point a must lie on the opposite side of ℓ_y with respect to v_i since B_{a,v_i} realizes a hyperbolic segment on the upper envelope $F^L(y)$ and all hyperbolic segments come from intersection points that lie in the rectangle bound by a and v_i (Observation 1).
3. Per assumption, $\|b - S_j\| < \|a - S_i\|$. The point a must lie on the circle centered at S_i with radius $\|a - S_i\|$. Combining our assumption with the fact that S_j lies right of S_i, we know that the point b cannot lie left of this circle. Since a lies clockwise of b with respect to V_i, it now follows that a lies left of b.

These three observations imply that the bisector B_{a,v_j} intersects the line ℓ_y left of S_j which contradicts the assumption that S_j lies on L_j. ◀

▶ **Lemma 2.7.** *The function F^L has complexity $O(n^{3/2})$.*

Proof. We upper bound the number of hyperbolic segments on F^L by bounding the number of elements in $\cup_i V_i^*$. Let v_a, v_b be the (at most) two rightmost vertices that participate in a backward pair whose bisector is a halfline of slope 1/-1 on $F^L(y)$. The sets V_a^* and V_b^* contain at most $O(n)$ elements. Now consider the bipartite graph $G = (L \cup R \backslash \{v_a, v_b\}, E)$ in which L and R are two copies of the vertices in $P \backslash \{v_a, v_b\}$. There is an edge between $v_j \in L$ and $v_i \in R$ if and only if $v_j \in V_i^*$. By Lemma 2.6, the graph G is $K_{4,2}$-free, and thus has at most $O(n^{3/2})$ edges [10, Theorem 4.5.2]. Since every edge corresponds to a (relevant) backward pair, it follows that the number of elements in $\cup_i V_i^*$ and therefore the number of hyperbolic segments on F^L is bounded by $O(n^{3/2})$. ◀

By Observation 2 the function F thus has complexity at most $O(n^{3/2})$ as well. Therefore, the data structure of de Berg et al. [5] uses at most $O(n^{3/2})$ space. We conclude:

▶ **Theorem 2.8.** *Given a polygonal curve P in $\mathbb{R}^2$ with n vertices, there is a data structure of size $O(n^{3/2})$, that can be built in $O(n^2 \log^2 n)$ time, that can report the Fréchet distance between P and a horizontal query segment in $O(\log^2 n)$ time.*

Acknowledgements. This research was initiated during the Dagstuhl seminar 19352.

References

1 Pankaj K Agarwal, Sariel Har-Peled, Nabil H Mustafa, and Yusu Wang. Near-linear time approximation algorithms for curve simplification. *Algorithmica*, 42(3-4):203–219, 2005.

2 H. Alt, A. Efrat, G. Rote, and C. Wenk. Matching planar maps. *Journal of algorithms*, 49(2):262–283, 2003.

3 K. Buchin, M. Buchin, J. Gudmundsson, M. Löffler, and J. Luo. Detecting commuting patterns by clustering subtrajectories. *International Journal of Computational Geometry & Applications*, 21(03):253–282, 2011.

4 Kevin Buchin, Maike Buchin, Marc Van Kreveld, Maarten Löffler, Rodrigo I Silveira, Carola Wenk, and Lionov Wiratma. Median trajectories. *Algorithmica*, 66(3):595–614, 2013.

5 Mark de Berg, Ali D Mehrabi, and Tim Ophelders. Data structures for fréchet queries in trajectory data. In *CCCG*, pages 214–219, 2017.

6 A. Driemel and S. Har-Peled. Jaywalking your dog: Computing the fréchet distance with shortcuts. *SIAM Journal on Computing*, 42(5):1830–1866, 2013.

7 J. Gudmundsson, M. Mirzanezhad, A. Mohades, and C. Wenk. Fast fréchet distance between curves with long edges. In *Proceedings of the 3rd International Workshop on Interactive and Spatial Computing*, IWISC '18, pages 52–58. ACM, 2018.

8 M. Jiang, Y. Xu, and B. Zhu. Protein structure–structure alignment with discrete Fréchet distance. *Journal of bioinformatics and computational biology*, 6(01):51–64, 2008.

9 S. Kwong, QH He, K. Man, KS Tang, and CW Chau. Parallel genetic-based hybrid pattern matching algorithm for isolated word recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 12(05):573–594, 1998.

10 J. Matoušek. *Lectures on discrete geometry*, volume 108. Springer, 2002.

Balanced Independent and Dominating Sets on Colored Interval Graphs*

Sujoy Bhore[1], Jan-Henrik Haunert[2], Fabian Klute[1], Guangping Li[1], and Martin Nöllenburg[1]

1 **Algorithms and Complexity Group, TU Wien, Vienna, Austria**
{sujoy, fklute, guangping, noellenburg}@ac.tuwien.ac.at
2 **Geoinformation Group, University of Bonn, Bonn, Germany**
haunert@igg.uni-bonn.de

Abstract

We study two new versions of independent and dominating set problems on vertex-colored interval graphs, namely f-*Balanced Independent Set* (f-BIS) and f-*Balanced Dominating Set* (f-BDS). Let $G = (V, E)$ be a vertex-colored interval graph with a k-coloring $\gamma\colon V \to \{1, \ldots, k\}$ for some $k \in \mathbb{N}$. A subset of vertices $S \subseteq V$ is called f-*balanced* if S contains f vertices from each color class. In the f-BIS and f-BDS problems, the objective is to compute an independent set or a dominating set that is f-balanced. We show that both problems are NP-complete even on proper interval graphs. For the BIS problem on interval graphs, we design two FPT algorithms, one parameterized by (f, k) and the other by the vertex cover number of G. Moreover, we present a 2-approximation algorithm for a slight variation of BIS on proper interval graphs.

1 Introduction

A graph G is an interval graph if it has an intersection model consisting of intervals on the real line. Formally, $G = (V, E)$ is an interval graph if there is an assignment of an interval $I_v \subseteq \mathbb{R}$ for each $v \in V$ such that $I_u \cap I_v$ is nonempty if and only if $(u, v) \in E$. A *proper* interval graph is an interval graph that has an intersection model in which no interval properly contains another [8]. Consider an interval graph $G = (V, E)$ and additionally assume that the vertices of G are k-colored by a mapping $\gamma\colon V \to \{1, \ldots, k\}$. We define and study color-balanced versions of two classical graph problems: maximum independent set and minimum dominating set on vertex-colored (proper) interval graphs. In what follows, we define the problems formally and discuss their underlying motivation.

f-Balanced Independent Set (f-BIS): Let $G = (V, E)$ be an interval graph with vertex coloring $\gamma\colon V \to \{1, \ldots, k\}$. Find an f-*balanced independent set* of G, i.e., an independent set $L \subseteq V$ that contains exactly f elements from each color class.

The classic maximum independent set problem serves as a natural model for many real-life optimization problems and finds applications across fields, e.g., computer vision [2], information retrieval [14], and scheduling [16]. Specifically, it has been used widely in map-labeling problems [1, 4, 17, 18], where an independent set of a given set of label candidates corresponds to a conflict-free and hence legible set of labels. To display as much relevant information as possible, one usually aims at maximizing the size or, in the case of weighted label candidates, the total weight of the independent set. This approach may be appropriate if all labels represent objects of the same category. In the case of multiple categories, however,

* We thank Robert Ganian for useful discussions and suggestions.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

maximizing the size or total weight of the labeling does not reflect the aim of selecting a good mixture of different object types. For example, if the aim was to inform a map user about different possible activities in the user's vicinity, labeling one cinema, one theater, and one museum may be better than labeling four cinemas. In such a setting, the f-BIS problem asks for an independent set that contains f vertices from each object type.

We initiate this study for interval graphs which is a primary step to understand the behavior of this problem on intersection graphs. Moreover, solving the problem for interval graphs gives rise to optimal solutions for certain labeling models, e.g., if every label candidate is a rectangle that is placed at a fixed position on the boundary of the map [9].

While there exists a simple greedy algorithm for the maximum independent set problem on interval graphs, it turns out that f-BIS is much more resilient and NP-complete even for proper interval graphs and $f = 1$ (Section 2.1). Then, in Section 3, we complement this complexity result with two FPT algorithms for interval graphs, one parameterized by (f, k) and the other parameterized by the vertex cover number. We conclude with a 2-approximation algorithm for a slight variation of BIS on proper interval graphs.

The second problem we discuss is

f-Balanced Dominating Set (f-BDS): Let $G = (V, E)$ be an interval graphs with vertex coloring $\gamma\colon V \to \{1, \ldots, k\}$. Find an *$f$-balanced dominating set*, i.e., a subset $D \subseteq V$ such that every vertex in $V \setminus D$ is adjacent to at least one vertex in D, and D contains exactly f elements from each color class.

The dominating set problem is another fundamental problem in theoretical computer science which also finds applications in various fields of science and engineering [5, 10]. Several variants of the dominating set problem have been considered over the years: k-tuple dominating set [6], Liar's dominating set [3], independent dominating set [11], and more. The colored variant of the dominating set problem has been considered in parameterized complexity, namely, red-blue dominating set, where the objective is to choose a dominating set from one color class that dominates the other color class [7]. Instead, our f-BDS problem asks for a dominating set of a vertex-colored graph that contains f vertices of each color class. Similar to the independent set problem, we primarily study this problem on vertex-colored interval graphs, which can be of independent interest. In Section 2.2, we prove that f-BDS on vertex-colored proper interval graphs is NP-complete, even for $f = 1$.

2 Complexity Results

In this section we show that f-BIS and f-BDS are NP-complete even if the given graph G is a proper interval graph and $f = 1$. Our reductions are from restricted, but still NP-complete versions of 3SAT, namely 3-bounded 3SAT [15] and 2P2N-3SAT (hardness follows from the result for 2P1N-SAT [19]). In the former 3SAT variant a variable is allowed to appear in at most three clauses and clauses have two or three literals, in the latter each variable appears exactly four times, twice as positive literal and twice as negative literal. Here we give the constructions for both reductions; for detailed proofs we refer to the full version of this paper.

▶ Remark. NP-completeness of 1-balanced independent (dominating) set implies the NP-completeness of f-balanced independent (dominating) set for $f > 1$. Let I_1 be the interval graph in an 1-balanced independent (dominating) set instance. We construct an interval graph I_f consisting of f independent copies of I_1. Clearly I_1 has 1-balanced independent (dominating) set if and only if I_f has an f-balanced independent (dominating) set.

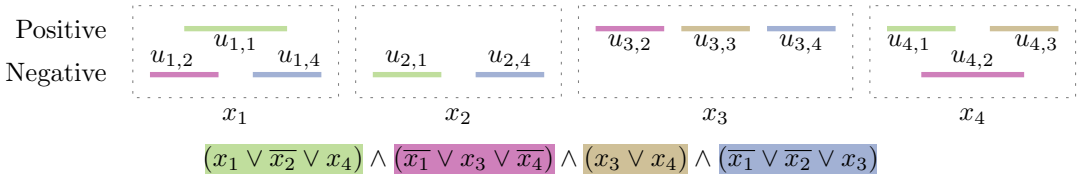


Figure 1 The graph resulting from the reduction for 1-balanced independent set in Theorem 1 depicted as interval representation with the vertex colors being the colors of the intervals.

2.1 f-Balanced Independent Set

Let $\phi(x_1, \ldots, x_n)$ be a 3-bounded 3SAT formula with variables $x_1, \ldots, x_n$ and clause set $\mathcal{C} = \{C_1, \ldots, C_m\}$. From ϕ we construct a proper interval graph $G = (V, E)$ and a vertex coloring γ of V as follows. We choose the set of colors to contain exactly m colors, one for each clause in $\mathcal{C}$ and we number these colors from 1 to m. We add a vertex $u_{i,j} \in V$ for each occurrence of a variable x_i in a clause C_j in ϕ. Furthermore, we insert an edge $u_{i,j}u_{i,j'} \in E$ whenever x_i appears positively in C_j and negatively in $C_{j'}$ (or vice versa). Finally, we color each vertex $u_{i,j} \in V$ with color j. See Figure 1 for an example. The graph G created from ϕ is a proper interval graph as it consists only of disjoint paths of length at most three and can clearly be constructed in polynomial time and space.

▶ **Theorem 1.** *The f-balanced independent set problem on a graph $G = (V, E)$ with vertex coloring $\gamma\colon V \to \{1, \ldots, k\}$ is* NP*-complete, even if G is a proper interval graph and $f = 1$.*

2.2 f-Balanced Dominating Set

We reduce from 2P2N-3SAT where each variable appears exactly twice positive and twice negative. Let $\phi(x_1, \ldots, x_n)$ be a 2P2N-3SAT formula with variables $x_1, \ldots, x_n$ and clause set $\mathcal{C} = \{C_1, \ldots, C_m\}$. For variable x_i in ϕ we denote with $\mathcal{C}_{x_i} = \{C_t^1, C_t^2, C_f^1, C_f^2\}$ the four clauses x_i appears in, where C_t^1, C_t^2 are clauses with positive occurrences of x_i and C_f^1, C_f^2 are clauses containing negative occurrences of x_i.

We construct a graph $G = (V, E)$ from $\phi(x_1, \ldots, x_n)$ as follows. For each variable x_i we introduce six vertices t_1, t_2, f_1, f_2, h_t, and h_f and for each clause C_j with occurrences of variables x_{j_1}, x_{j_2}, and x_{j_3} we add up to three vertices c_k for each $k \in \{j_1, j_2, j_3\}$ (In case a clause has less than three literals we add only one or two vertices). If the connection to the variable is clear, we also write c_t^1, c_t^2, c_f^1, and c_f^2 for the vertices introduced for this variable's occurrences in the clauses C_t^1, C_t^2, C_f^1, and C_f^2, respectively. Furthermore, we add for each variable x_i the edges (h_t, t_1), (h_t, t_2), (h_f, f_1), and (h_f, f_2), as well as for each clause C_j all possible edges between the three vertices introduced for C_j. For each variable x_i we introduce five colors, namely z_t^1, z_t^2, z_f^1, z_f^2, and z_h. We set $\gamma(h_t) = \gamma(h_f) = z_h$. Finally, we

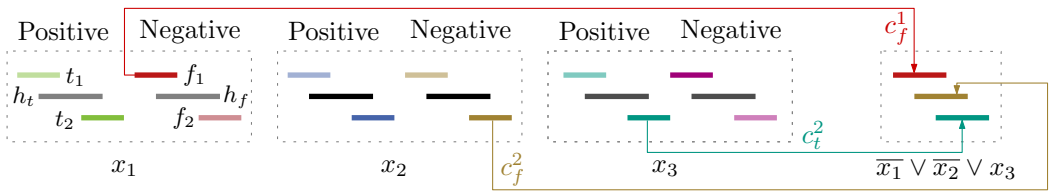


Figure 2 Illustrations of three variable gadgets and a clause gadget from Theorem 2 as interval representations. The vertex colors are given as the colors of the intervals.

set $\gamma(t_1) = \gamma(c_t^1) = z_t^1$. Equivalently for t_2, f_1, and f_2. See Figure 2 for an example.

In total we create $6n + 3m$ many vertices and $4n + 3m$ many edges, thus the reduction is in polynomial time. All variable and clause gadgets are independent components and only consist of paths of length three and triangles, hence G is a proper interval graph. Furthermore, G can clearly be constructed in polynomial time and space.

▶ **Theorem 2.** *The f-balanced dominating set problem on a graph $G = (V, E)$ with vertex coloring $\gamma : V \to \{1, \ldots, C\}$ is* NP*-complete, even if G is a proper interval graph and $f = 1$.*

3 Algorithmic Results for the Balanced Independent Set Problem

In this section we first take a parameterized perspective on f-BIS and provide two FPT algorithms[1] with different parameters. Then we give a 2-approximation algorithm for the related problem of maximizing the number of different colors in the independent set. For omitted proofs see the full version of this paper.

3.1 An FPT Algorithm Parameterized by (f, k)

Assume we are given an instance of f-BIS with $G = (V, E)$ being an interval graph with vertex coloring $\gamma\colon V \to \{1, \ldots, k\}$. We can construct an interval representation $\mathcal{I} = \{I_1, \ldots, I_n\}$, $n = |V|$, from G in linear time [12]. Then our algorithm works as follows. To start we sort the right end-points of the n intervals in $\mathcal{I}$ in ascending order. We define for all intervals I_i with $i > 0$ the index $0 \leq p_i < n$ as the index of the interval I_{p_i} whose right endpoint is rightmost before I_i's left endpoint. For each color $\kappa \in \{1, \ldots, k\}$, let $\hat{e}_\kappa$ denote the k-dimensional unit vector of the form $(0, \ldots, 0, 1, 0, \ldots, 0)$, where the element at the κ-th position is 1 and the rest are 0. For a subset $\mathcal{I}' \subseteq \mathcal{I}$ we define a *cardinality vector* as the k-dimensional vector $C_{\mathcal{I}'} = (c_1, \ldots, c_k)$, where each element c_i represents the number of intervals of color i in $\mathcal{I}'$. We say $C_{\mathcal{I}'}$ is *valid* if all $c_i \leq f$ and the set $\mathcal{I}'$ is independent.

The key observation here is that there are at most $O(f^k)$ many different valid cardinality vectors as there are only k colors and we are interested in at most f intervals per color. In the following let U_j, $j \in \{1, \ldots, n\}$, be the union of all valid cardinality vectors of the first j intervals in $\mathcal{I}$. Let $U_0 = \{(0, \ldots, 0)\}$ in the beginning. To compute an f-balanced independent set the algorithm simply iterates over all right endpoints of the intervals in $\mathcal{I}$ and in the i-th step computes U_i as $U_i = \{u + \hat{e}_{\gamma(I_i)} \mid u \in U_{p_i} \text{ and } u + \hat{e}_{\gamma(I_i)} \text{ is valid}\} \cup U_{i-1}$. Finally, we check the cardinality vectors in U_n and return *true* in case there is one with entries being all f and *false* otherwise. An f-balanced independent set can be easily retrieved by backtracking the decisions we made to compute the cardinality vector.

▶ **Theorem 3.** *Let $G = (V, E)$ be an interval graph with a vertex coloring $\gamma : V \to \{1, \ldots, k\}$. We can compute an f-balanced independent set of G or determine that no such set exists in $O(n \log n + n f^k \alpha(f^k))$ time, where α is the inverse Ackermann function.*

3.2 An FPT Algorithm Parameterized by the Vertex Cover Number

Here we will give an alternative FPT algorithm for f-BIS, this time parameterized by the *vertex cover number* $\tau(G)$ of G, i.e., the size of a minimum vertex cover of G.

[1] FPT is the class of parameterized problems that can be solved in time $O(g(k)n^{O(1)})$ for input size n, parameter k, and some computable function g.

▶ **Lemma 4.** *Let $G = (V, E)$ be a graph with vertex cover number $\tau(G)$. There are $O(2^{\tau(G)})$ maximal independent sets of G.*

Proof. Consider a minimum vertex cover V_c in G and its complement $V_{\text{ind}} = V \setminus V_c$. Note that since V_c is a (minimum) vertex cover, V_{ind} is a (maximum) independent set. Furthermore, any maximal independent set M of G can be constructed from V_{ind} by adding $M \cap V_c$ and removing its neighborhood in V_{ind}, namely $M = (V_{\text{ind}} \cup (M \cap V_c)) \setminus N(M \cap V_c)$ (see the full version for details). Thus there are $O(2^{\tau(G)})$ maximal independent sets of G. ◀

▶ **Theorem 5.** *Let $G = (V, E)$ be an interval graph with a vertex coloring $\gamma\colon V \to \{1, \ldots, k\}$. We can compute an f-balanced independent set of G or determine that no such set exists in $O(2^{\tau(G)} \cdot n)$ time.*

Proof. According to Lemma 4, there are $O(2^{\tau(G)})$ maximal independent sets of G. The basic idea is to enumerate all the $O(2^{\tau(G)})$ maximal independent sets and compute their maximum balanced subsets. Enumerating all maximal independent sets of an interval graph takes $O(1)$ time per output [13]. Given an arbitrary independent set of G we can compute an f-balanced independent subset in $O(n)$ time or conclude that no such subset exists. Therefore, the running time of the algorithm is $O(2^{\tau(G)} \cdot n)$. ◀

3.3 A 2-Approximation for 1-Max-Colored Independent Sets

Here we study a variation of BIS, which asks for a maximally colorful independent set.

1-Max-Colored Independent Set (1-MCIS): Let $G = (V, E)$ be a proper interval graph with vertex coloring $\gamma\colon V \to \{1, \ldots, k\}$. Find a 1-max-colored independent set of G, i.e., an independent set $L \subseteq V$, whose vertices contain a maximum number of colors.

We note that the NP-completeness of 1-BIS implies that 1-MCIS is an NP-hard optimization problem as well. In the following, we will show a simple sweep algorithm for 1-MCIS with approximation ratio 2.

Our algorithm selects one interval for each color greedily. We maintain an array S of size k to store the selected intervals. After sorting the n right end-points in ascending order, we scan the intervals from left to right. For each interval I_i in this order, we check if the color of I_i is still missing in our solution (by checking if $S[\gamma(I_i)]$ is not yet occupied). If so, we store I_i in $S[\gamma(i)]$ and remove all remaining intervals overlapping I_i. Otherwise, if $S[\gamma(I_i)]$ is not empty, we remove I_i and continue scanning the intervals. This is repeated until all intervals are processed. Using that G is a proper interval graph and a charging argument on the colors in an optimal solution that are missing in the greedy solution, we obtain our approximation result.

▶ **Theorem 6.** *Let $G = (V, E)$ be a proper interval graph with a vertex coloring $\gamma\colon V \to \{1, \ldots, k\}$. In $O(n \log n)$ time, we can compute an independent set with at least $\lceil \frac{c}{2} \rceil$ colors, where c is the number of colors in a 1-max-colored independent set.*

References

1 Pankaj K. Agarwal, Marc J. Van Kreveld, and Subhash Suri. Label placement by maximum independent set in rectangles. *Computational Geometry: Theory and Applications*, 11(3-4):209–218, 1998. doi:10.1016/S0925-7721(98)00028-5.

2 Egon Balas and Chang Sung Yu. Finding a maximum clique in an arbitrary graph. *SIAM Journal on Computing*, 15(4):1054–1068, 1986. doi:10.1137/0215075.

3 Sandip Banerjee and Sujoy Bhore. Algorithm and hardness results on liar's dominating set and k-tuple dominating set. In *Proceedings of the 30th International Workshop on Combinatorial Algorithms (IWOCA'19)*, volume 11638 of *LNCS*, pages 48–60. Springer, 2019. doi:10.1007/978-3-030-25005-8_5.

4 Ken Been, Martin Nöllenburg, Sheung-Hung Poon, and Alexander Wolff. Optimizing active ranges for consistent dynamic map labeling. *Computational Geometry: Theory and Applications*, 43(3):312–328, 2010. doi:10.1016/j.comgeo.2009.03.006.

5 Gerard J. Chang. Algorithmic aspects of domination in graphs. In *Handbook of Combinatorial Optimization*, pages 1811–1877. Springer, 1998. doi:10.1007/978-1-4613-0303-9_28.

6 Mustapha Chellali, Odile Favaron, Adriana Hansberg, and Lutz Volkmann. k-domination and k-independence in graphs: A survey. *Graphs and Combinatorics*, 28(1):1–55, 2012. doi:10.1007/s00373-011-1040-3.

7 Valentin Garnero, Ignasi Sau, and Dimitrios M. Thilikos. A linear kernel for planar red–blue dominating set. *Discrete Applied Mathematics*, 217:536–547, 2017. doi:10.1016/j.dam.2016.09.045.

8 Martin Charles Golumbic. *Algorithmic graph theory and perfect graphs*. Elsevier, 2004.

9 Jan-Henrik Haunert and Tobias Hermes. Labeling circular focus regions based on a tractable case of maximum weight independent set of rectangles. In *Proceedings of the 2nd ACM SIGSPATIAL Workshop on MapInteraction*, 2014. doi:10.1145/2677068.2677069.

10 Teresa W. Haynes, Stephen Hedetniemi, and Peter Slater. *Fundamentals of domination in graphs*, volume 208 of *Pure and applied mathematics*. Dekker, 1998.

11 Robert W. Irving. On approximating the minimum independent dominating set. *Information Processing Letters*, 37(4):197–200, 1991. doi:10.1016/0020-0190(91)90188-N.

12 C. Lekkeikerker and J. Boland. Representation of a finite graph by a set of intervals on the real line. *Fundamenta Mathematicae*, 51(1):45–64, 1962. doi:10.4064/fm-51-1-45-64.

13 Yoshio Okamoto, Takeaki Uno, and Ryuhei Uehara. Counting the number of independent sets in chordal graphs. *Journal of Discrete Algorithms*, 6(2):229–242, 2008. doi:10.1016/j.jda.2006.07.006.

14 Panos M. Pardalos and Jue Xue. The maximum clique problem. *Journal of Global Optimization*, 4(3):301–328, 1994. doi:10.1007/BF01098364.

15 Craig A. Tovey. A simplified NP-complete satisfiability problem. *Discrete Applied Mathematics*, 8(1):85–89, 1984. doi:10.1016/0166-218X(84)90081-7.

16 René Van Bevern, Matthias Mnich, Rolf Niedermeier, and Mathias Weller. Interval scheduling and colorful independent sets. *Journal of Scheduling*, 18(5):449–469, 2015. doi:10.1007/s10951-014-0398-5.

17 Marc J. van Kreveld, Tycho Strijk, and Alexander Wolff. Point labeling with sliding labels. *Computational Geometry: Theory and Applications*, 13(1):21–47, 1999. doi:10.1016/S0925-7721(99)00005-X.

18 Frank Wagner and Alexander Wolff. A combinatorial framework for map labeling. In *Proceedings of the 6th International Symposium on Graph Drawing (GD'98)*, volume 1547 of *LNCS*, pages 316–331. Springer, 1998. doi:10.1007/3-540-37623-2_24.

19 Ryo Yoshinaka. Higher-order matching in the linear lambda calculus in the absence of constants is NP-complete. In *Proceedings of the 16th International Conference on Term Rewriting and Applications (RTA'05)*, volume 3467 of *LNCS*, pages 235–249. Springer, 2005. doi:10.1007/978-3-540-32033-3_18.

The Complexity of Finding Tangles

Oksana Firman[1], Stefan Felsner[2], Philipp Kindermann[1], Alexander Ravsky[3], Alexander Wolff[1], and Johannes Zink[1]

1 **Universität Würzburg, Germany**
`firstname.lastname@uni-wuerzburg.de`
2 **TU Berlin, Germany**
`felsner@math.tu-berlin.de`
3 **Pidstryhach Institute for Applied Problems of Mechanics and Mathematics, National Academy of Sciences of Ukraine, Lviv, Ukraine**
`alexander.ravsky@uni-wuerzburg.de`

Abstract

We study the following combinatorial problem. Given a set of n y-monotone curves, which we call *wires*, a *tangle* determines the order of the wires on a number of horizontal *layers* such that the orders of the wires on any two consecutive layers differ only in swaps of neighboring wires. Given a multiset L of *swaps* (that is, unordered pairs of wires) and an initial order of the wires, a tangle *realizes* L if each pair of wires changes its order exactly as many times as specified by L. Finding a tangle that realizes a given multiset of swaps and uses the least number of layers is known to be NP-hard. We show that it is even NP-hard to decide if a realizing tangle exists.

1 Introduction

The subject of this paper is the visualization of so-called *chaotic attractors*, which occur in chaotic dynamic systems. Such systems are considered in physics, celestial mechanics, electronics, fractals theory, chemistry, biology, genetics, and population dynamics. Birman and Williams [3] were the first to mention tangles as a way to describe the topological structure of chaotic attractors. They investigated how the orbits of attractors are knotted. Later Mindlin et al. [6] showed how to characterize attractors using integer matrices that contain numbers of swaps between the orbits.

Olszewski et al. [7] studied computational aspects of visualizing chaotic attractors. In the framework of their paper, one is given a set of y-monotone curves called *wires* that hang off a horizontal line in a fixed order, and a multiset of swaps between the wires (called *list*). A *tangle* then is a visualization of these swaps, i.e., a sequence of permutations of the wires such that consecutive permutations differ only in swaps of neighboring wires (but disjoint swaps can be done simultaneously). For examples of lists and tangles realizing them, see Figs. 1 and 2. The list L in Fig. 1 admits a tangle realizing it. We call such a list *feasible*. The list L', in contrast, is not feasible. In Fig. 2, the list L_n is described by an $(n \times n)$-matrix. The gray horizontal bars correspond to the permutations (or *layers*). Olszewski et al. gave

$$L = \{(1,2),(1,3)\}$$

$$L' = \{(1,2)_2,(1,3)\}$$

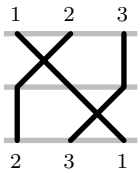


Figure 1 Lists L and L' for three wires (left). The list L is feasible (a tangle realizing L is to the right), whereas L', which has two copies of the swap $(1,2)$, is infeasible.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

$$L_n = \begin{pmatrix} 0 & 1 & 1 & \dots & 1 & 0 & 2 \\ 1 & 0 & 1 & \dots & 1 & 2 & 0 \\ 1 & 1 & 0 & \dots & 1 & 0 & 2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 1 & 1 & 1 & \dots & 0 & \mathbf{0} & \mathbf{2} \\ 0 & 2 & 0 & \dots & \mathbf{0} & 0 & n-1 \\ 2 & 0 & 2 & \dots & \mathbf{2} & n-1 & 0 \end{pmatrix}$$

(The bold zeros and twos must be exchanged if n is even.)

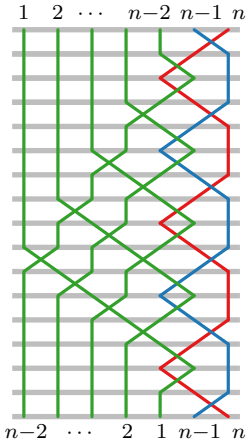


Figure 2 A list L_n for n wires (left) and a tangle realizing L_n (right). Entry (i, j) of L_n defines how often wires i and j must swap in the tangle. Here, $n = 7$.

an exponential-time algorithm for minimizing the *height* of a tangle, that is, the number of layers. They tested their algorithm on a benchmark set.

Later, we [5] showed that in fact tangle-height minimization is NP-hard. Our proof was by reduction from 3-Partition. We also presented an (exponential-time) algorithm for the problem. Using an extended benchmark set, we showed that in almost all cases our algorithm is faster than the algorithm of Olszewski et al.

Sado and Igarashi [8] used the same optimization criterion for tangles, given only the final permutation. They used odd-even sort, a parallel variant of bubble sort, to compute tangles with at most one layer more than the minimum. Wang [10] showed that there is always a height-optimal tangle where no swap occurs more than once. Bereg et al. [1, 2] considered a similar problem. Given a final permutation, they showed how to minimize the number of bends or *moves* (which are maximal "diagonal" segments of the wires).

In this paper we strengthen our previous results and show that it is even NP-hard to test, given a multiset of swaps and a start permutation of the wires, whether there is *any* tangle that realizes the given swaps. We call this problem List-Feasibility.

2 Complexity

We show that List-Feasibility is NP-hard by reducing from Positive NAE 3-SAT Diff, a variant of Not-All-Equal 3-SAT. Recall that in Not-All-Equal 3-SAT one is given a conjunctive normal form with three literals per clause and the task is to decide whether there exists a variable assignment such that in no clause all three literals have the same truth value. By Schaefer's dichotomy theorem [9], Not-All-Equal 3-SAT is NP-hard even if no negative literals are admitted. In Positive NAE 3-SAT Diff, additionally each clause contains three different variables. It is easy to see that this variant is NP-hard, too.

▶ **Lemma 1.** Positive NAE 3-SAT Diff *is NP-hard.*

For a formal proof, see the full version [4]. Our main result is as follows.

▶ **Theorem 2.** List-Feasibility *is NP-hard (even if every pair of wires has at most eight swaps).*

We split our proof into several parts. First we introduce some notation, then we give the intuition behind our reduction. Next, we explain variable and clause gadgets in more detail. Finally, we show the correctness of the reduction.

Notation. We label the wires by their index in the initial permutation of a tangle. In particular, for a wire ε, its neighbor to the right is wire $\varepsilon + 1$. If a wire μ is to the left of some other wire ν, we write $\mu < \nu$. If all wires in a set M are to the left of all wires in a set N, we write $M < N$. For any integer $k > 0$, let $[k] = \{1, 2, \ldots, k\}$.

Setup. Given an instance $F = d_1 \wedge \cdots \wedge d_m$ of Positive NAE 3-SAT Diff with variables $w_1, \ldots, w_n$, we construct in polynomial time a list L of swaps such that there is a tangle T realizing L if and only if F is a yes-instance.

In L we have two inner wires λ and $\lambda' = \lambda + 1$ that swap eight times. This yields two types of loops (see Fig. 3): four λ'–λ loops, where λ' is on the left and λ is on the right side, and three λ–λ' loops with λ on the left and λ' on the right side. Notice that we consider only *closed* loops, which are bounded by swaps between λ and λ'. In the following, we construct variable and clause gadgets. Each variable gadget will contain a specific wire that represents the variable, and each clause gadget will contain a specific wire that represents the clause. The corresponding variable and clause wires swap in one of the four λ'–λ loops. We call the first two λ'–λ loops *true-loops*, and the last two λ'–λ loops *false-loops*. If the corresponding variable is true, then the variable wire swaps with the corresponding clause wires in a true-loop, otherwise in a false-loop.

Apart from λ and λ', our list L contains (many) other wires, which we split into groups. For every $i \in [n]$, we introduce sets V_i and V_i' of wires that together form the gadget for variable w_i of F. These sets are ordered (initially) $V_n < V_{n-1} < \cdots < V_1 < \lambda < \lambda' < V_1' < V_2' < \cdots < V_n'$; the order of the wires inside these sets will be detailed in the next two paragraphs. Let $V = V_1 \cup V_2 \cup \cdots \cup V_n$ and $V' = V_1' \cup V_2' \cup \cdots \cup V_n'$. Similarly, for every $j \in [m]$, we introduce a set C_j of wires that contains a *clause wire* c_j and three sets of wires D_j^1, D_j^2, and D_j^3 that represent occurrences of variables in a clause d_j of F. The wires in C_j are ordered $D_j^3 < D_j^2 < D_j^1 < c_j$. Together, the wires in $C = C_1 \cup C_2 \cup \cdots \cup C_m$ represent the clause gadgets; they are ordered $V < C_m < C_{m-1} < \cdots < C_1 < \lambda$. Additionally, our list L contains a set $E = \{\varphi_1, \ldots, \varphi_7\}$ of wires that will make our construction rigid enough. The order of all wires in L is $V < C < \lambda < \lambda' < E < V'$. Now we present our gadgets in more detail.

Variable gadget. For each variable w_i of F, $i \in [n]$, we introduce two sets of wires V_i and V_i'. Each V_i' contains a *variable wire* v_i that has four swaps with λ and no swaps with λ'. Therefore, v_i intersects at least one and at most two λ'–λ loops. In order to prevent v_i from intersecting both a true- and a false-loop, we introduce two wires $\alpha_i \in V_i$ and $\alpha_i' \in V_i'$ with $\alpha_i < \lambda < \lambda' < \alpha_i' < v_i$; see Fig. 3. These wires neither swap with v_i nor with each other, but they have two swaps with both λ and λ'. We want to force α_i and α_i' to have the two true-loops on their right and the two false-loops on their left, or vice versa. This will ensure that v_i cannot reach both a true- and a false-loop.

To this end, we introduce, for $j \in [5]$, a *β_i-wire* $\beta_{i,j} \in V_i$ and a *β_i'-wire* $\beta_{i,j}' \in V_i'$. These are ordered $\beta_{i,5} < \beta_{i,4} < \cdots < \beta_{i,1} < \alpha_i$ and $\alpha_i' < \beta_{i,1}' < \beta_{i,2}' < \cdots < \beta_{i,5}' < v_i$. Every pair of β_i-wires as well as every pair of β_i'-wires swaps exactly once. Neither β_i- nor β_i'-wires swap with α_i or α_i'. Each β_i'-wire has four swaps with v_i. Moreover, $\beta_{i,1}, \beta_{i,3}, \beta_{i,5}, \beta_{i,2}', \beta_{i,4}'$ swap with λ twice. Symmetrically, $\beta_{i,2}, \beta_{i,4}, \beta_{i,1}', \beta_{i,3}', \beta_{i,5}'$ swap with λ' twice; see Fig. 3.

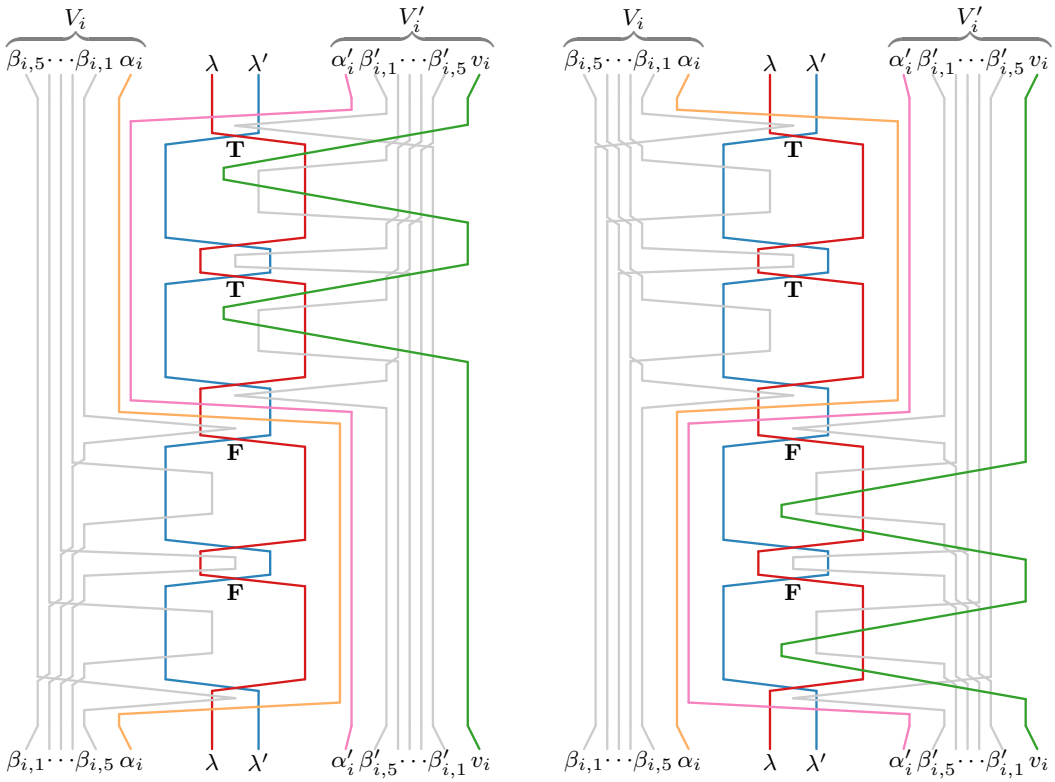


Figure 3 A variable gadget with a variable wire v_i that corresponds to the variable that is true (on the left) or false (on the right).

We use the β_i- and β_i'-wires to fix the minimum number of λ'–λ loops that are on the left of α_i and on the right of α_i', respectively. Note that, together with λ and λ', the β_i- and β_i'-wires have the same rigid structure as the wires in Fig. 2.

▶ **Observation 1** ([5]). The tangle in Fig. 2 realizes the list L_n specified there; all tangles that realize L_n have the same order of swaps along each wire.

This means that there is a unique order of swaps between the β_i-wires and λ or λ', i.e., for $j \in [4]$, every pair of $\beta_{i,j+1}$–λ swaps (or $\beta_{i,j+1}$–λ' swaps, depending on the parity of j) can be done only after the pair of $\beta_{i,j}$–λ' swaps (or $\beta_{i,j}$–λ swaps, respectively). We have the same rigid structure on the right side with β_i'-wires. Hence, there are at least two λ'–λ loops to the left of α_i and at least two to the right of α_i'. Since α_i and α_i' do not swap, there cannot be a λ'–λ loop that appears simultaneously on both sides.

Note that the λ–λ' swaps that belong to the same side have to be consecutive, otherwise α_i or α_i' would need to swap more than twice with λ and λ'. Thus, there are only two ways to order the swaps among the wires α_i, α_i', λ, λ'; the order is either α_i'–λ', α_i'–λ, four times λ–λ', α_i'–λ, α_i'–λ', α_i–λ, α_i–λ', four times λ–λ', α_i–λ', α_i–λ (see Fig. 3(left)) or the reverse (see Fig. 3(right)). It is easy to see that in the first case v_i can reach only the first two λ'–λ loops (the true-loops), and in the second case only the last two (the false-loops).

To avoid that the gadget for variable w_i restricts the proper functioning of the gadget for some variable w_j with $j > i$, we add the following swaps to L: for any $j > i$, α_j and α_j' swap with both V_i and V_i' twice, the β_j-wires swap with α_i' and V_i twice, and, symmetrically, the β_j'-wires swap with α_i and V_i' twice, v_j swaps with α_i and all wires in V_i' six times. We

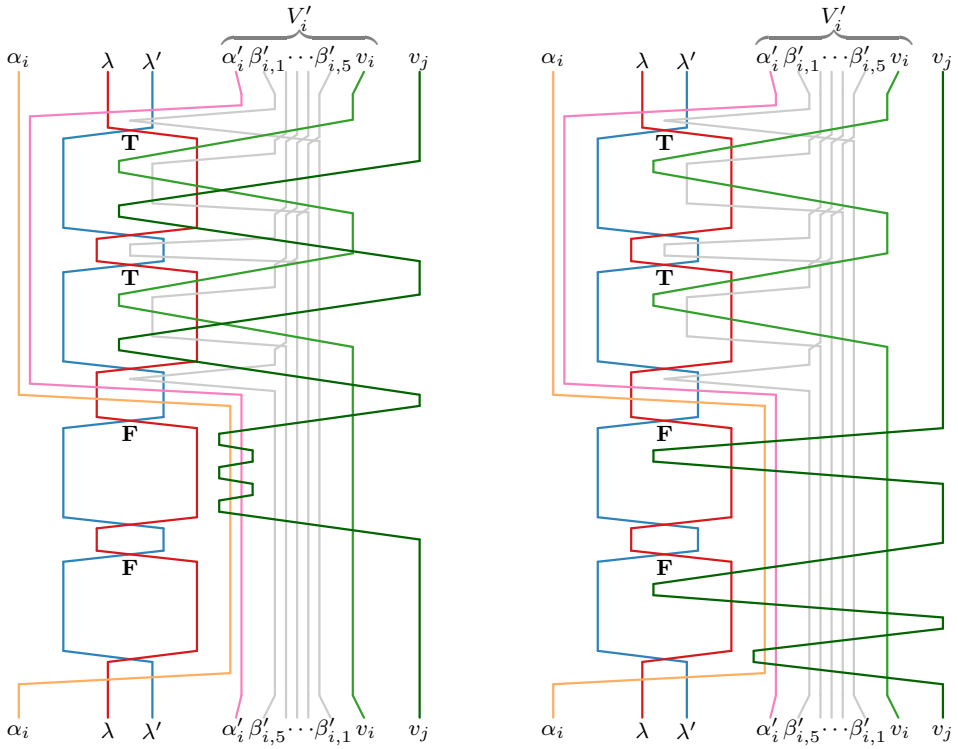


Figure 4 A realization of swaps between the variable wire v_j and all wires that belong to the variable gadget corresponding to the variable w_i. On the left the variables w_i and w_j are both true, and on the right w_i is true, whereas w_j is false.

briefly explain these multiplicities. Wires from V_j and $V_j' \setminus \{v_j\}$ swap their partners twice so that they reach the corresponding λ–λ' or λ'–λ loops and go back. None of the wires from V_i or V_i' is restricted in which loop to intersect. Considering the wire v_j, note that it has to reach the λ'–λ loops twice. For simplicity and in order not to have any conflicts with the β_i'-wires, we introduce exactly six swaps with α_i and all wires in V_i', see Fig. 4.

Clause gadget. For every clause d_j from F, $j \in [m]$, we introduce a set of wires C_j. It contains the clause wire c_j that has eight swaps with λ'. We want to force each c_j to appear in all λ'–λ loops. To this end, we use the set E with the seven *φ-wires* $\varphi_1, \ldots, \varphi_7$ ordered $\varphi_1 < \cdots < \varphi_7$. They create a rigid structure similar to the one of the β_i-wires. Each pair of φ-wires swaps exactly once. For each $k \in [7]$, if k is odd, φ_k swaps twice with λ and twice with c_j for every $j \in [m]$. If k is even, φ_k swaps twice with λ'. Since c_j does not swap with λ, each pair of swaps between c_j and a φ-wire with odd index appears inside a λ'–λ loop. Due to the rigid structure, each of these pairs of swaps occurs in a different λ'–λ loop; see Fig. 5.

If a variable w_i belongs to a clause d_j, then L contains two v_i–c_j swaps. Since every clause has exactly three different positive variables, we want to force variable wires that belong to the same clause to swap with the corresponding clause wire in different λ'–λ loops. This way, every clause contains at least one true and at least one false variable if F is satisfiable.

We call a part of a clause wire c_j that is inside a λ'–λ loop—i.e., a λ'–c_j loop—an *arm of the clause* c_j. We want to "*protect*" the arm that is intersected by a variable wire from other variable wires. To this end, for every occurrence $k \in [3]$ of a variable in d_j, we introduce four more wires. The wire γ_j^k will protect the arm of c_j that the variable wire of the k-th variable

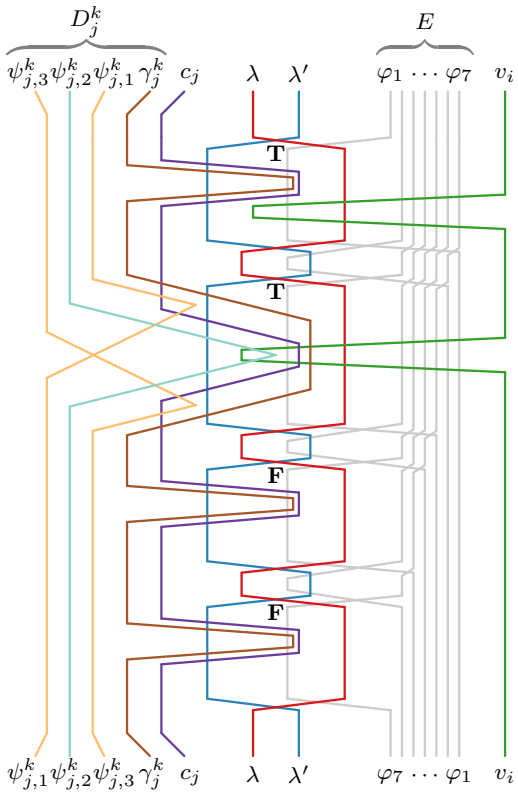


Figure 5 A gadget for clause c_j showing only one of the three variables, namely v_i.

of d_j intersects. To achieve this, γ_j^k has to intersect one of the φ-wires that swaps with the arm. In order not to restrict the choice of the λ'–λ loop, γ_j^k swaps twice with each φ_ℓ with odd $\ell \in [7]$. Similarly to c_j, the wire γ_j^k has eight swaps with λ' and appears once in every λ'–λ loop. Additionally, γ_j^k has two swaps with c_j.

We force γ_j^k to protect the correct arm in the following way. Consider the λ'–λ loop where an arm of c_j swaps with a variable wire v_i. We want the order of swaps along λ' inside this loop to be fixed as follows: λ' first swaps with γ_j^k, then twice with c_j, and then again with γ_j^k. This would prevent all variable wires that do not swap with γ_j^k from reaching the arm of c_j. To achieve this, we introduce three *ψ_j^k-wires* $\psi_{j,1}^k, \psi_{j,2}^k, \psi_{j,3}^k$ with $\psi_{j,3}^k < \psi_{j,2}^k < \psi_{j,1}^k < \gamma_j^k$.

The ψ_j^k-wires also have the rigid structure similar to the one that β_i-wires have, so that there is a unique order of swaps along each ψ_j^k-wire. Each pair of ψ_j^k-wires swaps exactly once, $\psi_{j,1}^k$ and $\psi_{j,3}^k$ have two swaps with c_j, and $\psi_{j,2}^k$ has two swaps with λ' and v_i. Note that no ψ_j^k-wire swaps with γ_j^k. Also, since $\psi_{j,2}^k$ does not swap with c_j, the $\psi_{j,2}^k$–v_i swaps can appear only inside the λ'–c_j loop that contains the arm of c_j we want to protect from other variable wires. Since c_j has to swap with $\psi_{j,1}^k$ before and with $\psi_{j,3}^k$ after the $\psi_{j,2}^k$–λ' swaps, and since there are only two swaps between γ_j^k and c_j, there is no way for any variable wire except for v_i to reach the arm of c_j without also intersecting γ_j^k; see Fig. 5.

Finally, we consider the behavior of wires from different clause gadgets among each other and with respect to wires from variable gadgets. For every $\ell > k$ and for every $j \in [m]$, the wires c_j and γ_j^ℓ have eight swaps and the ψ_j^ℓ-wires have two swaps with all wires in C_j. Since all wires in V are to the left of all wires in C, each wire in C swaps twice with all wires in V and, for $i \in [n]$, with α_i'. Finally, all α- and α'-wires swap twice with each φ-wire.

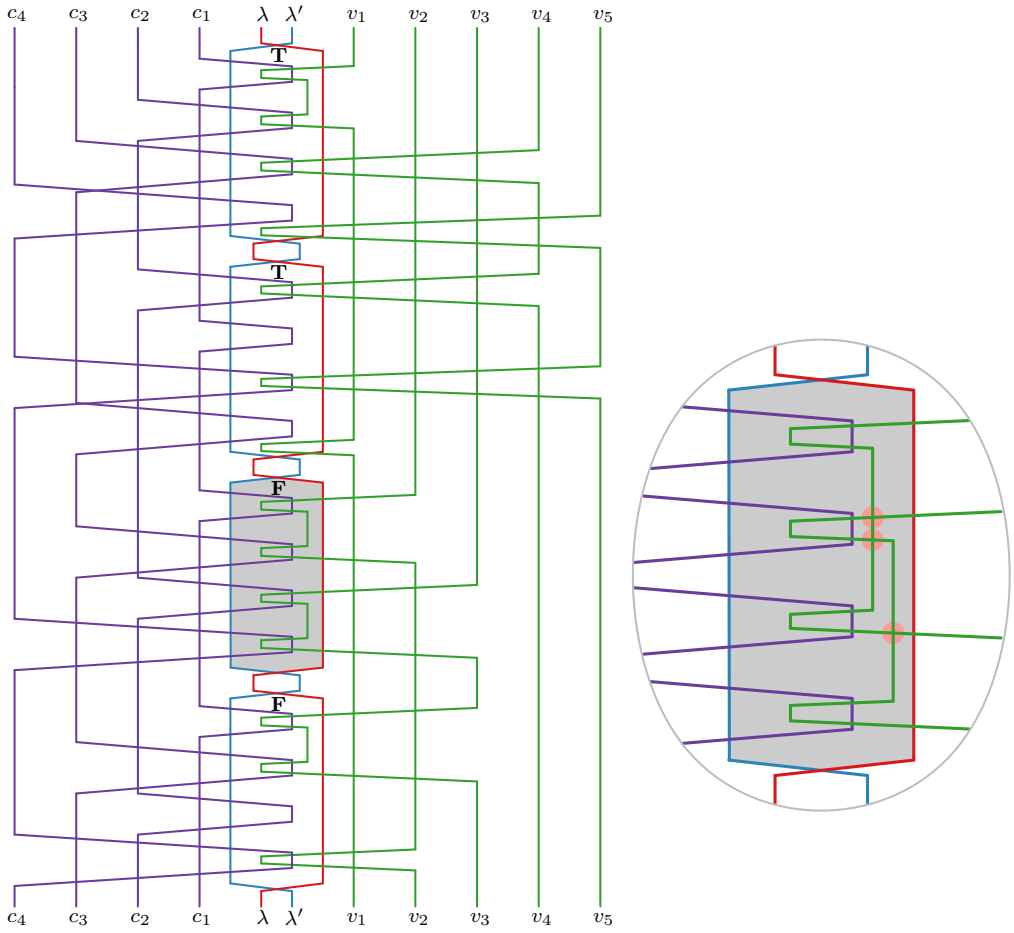


Figure 6 A tangle obtained from the satisfiable formula $F = (w_1 \vee w_2 \vee w_3) \wedge (w_1 \vee w_3 \vee w_4) \wedge (w_2 \vee w_3 \vee w_4) \wedge (w_2 \vee w_3 \vee w_5)$. Here w_1, w_4 and w_5 are set to true, whereas w_2 and w_3 are set to false. Note that we show here only "crucial" wires, namely λ, λ', and all variable and clause wires.

Note that the order of the arms of the clause wires inside a λ'–λ loop cannot be chosen arbitrarily. If a variable wire intersects more than one clause wire, the arms of these clause wires should be consecutive, as for v_2 and v_3 in the shaded region in Fig. 6. If we had an interleaving pattern of variable wires (see inset), say v_2 first intersects c_1, then v_3 intersects c_2, then v_2 intersects c_3, and finally v_3 intersects c_4, then v_2 and v_3 would have to swap at least three times within the same λ'–λ loop. However, we have reserved only eight swaps for each pair of variable wires—two for each of the four λ'–λ loops.

Correctness. Clearly, if F is satisfiable, then there is a tangle obtained from F as described above that realizes the list L, so L is feasible; see Fig. 6 for an example.

On the other hand, if there is a tangle that realizes the list L that we obtain from the reduction, then F is satisfiable. This follows from the rigid structure of a tangle that realizes L. The only flexibility is in which type of loop (true or false) a variable wire swaps with the corresponding clause wire. As described above, a tangle exists if, for each clause, the corresponding three variable wires swap with the clause wire in three different loops (at least one of which is a true-loop and at least one is a false-loop). In this case, the position of the variable wires yields a truth assignment satisfying F.

3 Open Problems

We recall three open questions of our previous paper [5].

- Can we decide the feasibility of a list faster than finding its optimal realization?
- For *simple* lists, that is, lists where each swap occurs at most once, odd-even sort efficently computes tangles with at most one layer more than minimum [8]. Can minimum-height tangles be computed efficiently for simple lists?
- In this paper, we showed that it is NP-hard to decide whether general lists are feasible. Earlier, we showed that the problem is easy for odd lists [5], i.e., if every swap occurs an odd number of times or zero times. A list is *non-separable* if, for every three wires $i < k < j$, there is no swap between i and k, and none between k and j, then there isn't any between i and j either. We conjecture that every non-separable even list is feasible.

References

1 Sergey Bereg, Alexander Holroyd, Lev Nachmanson, and Sergey Pupyrev. Representing permutations with few moves. *SIAM J. Discrete Math.*, 30(4):1950–1977, 2016. URL: https://arxiv.org/abs/1508.03674, doi:10.1137/15M1036105.

2 Sergey Bereg, Alexander E. Holroyd, Lev Nachmanson, and Sergey Pupyrev. Drawing permutations with few corners. In Stephen Wismath and Alexander Wolff, editors, *Proc. Int. Symp. Graph Drawing (GD'13)*, volume 8242 of *LNCS*, pages 484–495. Springer, 2013. URL: http://arxiv.org/abs/1306.4048, doi:10.1007/978-3-319-03841-4_42.

3 Joan S. Birman and R. Fredrick Williams. Knotted periodic orbits in dynamical systems—I: Lorenz's equation. *Topology*, 22(1):47–82, 1983. doi:10.1016/0040-9383(83)90045-9.

4 Oksana Firman, Stefan Felsner, Philipp Kindermann, Alexander Ravsky, Alexander Wolff, and Johannes Zink. The complexity of finding tangles. Arxiv report, 2020. URL: http://arxiv.org/abs/2002.12251.

5 Oksana Firman, Philipp Kindermann, Alexander Ravsky, Alexander Wolff, and Johannes Zink. Computing optimal-height tangles faster. In Daniel Archambault and Csaba D. Tóth, editors, *Proc. 27th Int. Symp. Graph Drawing & Network Vis. (GD'19)*, volume 11904 of *LNCS*, pages 203–215. Springer, 2019. URL: http://arxiv.org/abs/1901.06548, doi:10.1007/978-3-030-35802-0_16.

6 Gabo Mindlin, Xin-Jun Hou, Robert Gilmore, Hernán Solari, and N. B. Tufillaro. Classification of strange attractors by integers. *Phys. Rev. Lett.*, 64:2350–2353, 1990. doi:10.1103/PhysRevLett.64.2350.

7 Maya Olszewski, Jeff Meder, Emmanuel Kieffer, Raphaël Bleuse, Martin Rosalie, Grégoire Danoy, and Pascal Bouvry. Visualizing the template of a chaotic attractor. In Therese Biedl and Andreas Kerren, editors, *Proc. 26th Int. Symp. Graph Drawing & Network Vis. (GD'18)*, volume 11282 of *LNCS*, pages 106–119. Springer, 2018. URL: https://arxiv.org/abs/1807.11853, doi:10.1007/978-3-030-04414-5_8.

8 Kazuhiro Sado and Yoshihide Igarashi. A function for evaluating the computing time of a bubbling system. *Theor. Comput. Sci.*, 54:315–324, 1987. doi:10.1016/0304-3975(87)90136-8.

9 Thomas J. Schaefer. The complexity of satisfiability problems. In *Proc. 10th Annu. ACM Symp. Theory Comput. (STOC'78)*, pages 216–226, 1978. doi:10.1145/800133.804350.

10 Deborah C. Wang. Novel routing schemes for IC layout part I: Two-layer channel routing. In *Proc. 28th ACM/IEEE Design Automation Conf. (DAC'91)*, pages 49–53, 1991. doi:10.1145/127601.127626.

Sparse Regression via Range Counting*†

Jean Cardinal[1] and Aurélien Ooms[2]

1 **Université libre de Bruxelles (ULB), Brussels, Belgium**
`jcardin@ulb.ac.be`
2 **BARC, University of Copenhagen, Denmark**
`aurelien.ooms@di.ku.dk`

Abstract

The sparse regression problem, also known as best subset selection problem, can be cast as follows: Given a set S of n points in $\mathbb{R}^d$, a point $y \in \mathbb{R}^d$, and an integer $2 \leq k \leq d$, find an affine combination of at most k points of S that is nearest to y. We describe a $O(n^{k-1} \log^{d-k+2} n)$-time randomized $(1+\varepsilon)$-approximation algorithm for this problem with d and ε constant. This is the first algorithm for this problem running in time $o(n^k)$. Its running time is similar to the query time of a data structure recently proposed by Har-Peled, Indyk, and Mahabadi (ICALP'18), while not requiring any preprocessing. Up to polylogarithmic factors, it matches a conditional lower bound relying on a conjecture about affine degeneracy testing. In the special case where $k = d = O(1)$, we provide a simple $O_\delta(n^{d-1+\delta})$-time deterministic exact algorithm, for any $\delta > 0$. Finally, we show how to adapt the approximation algorithm for the sparse linear regression and sparse convex regression problems with the same running time, up to polylogarithmic factors.

1 Introduction

Searching for a point in a set that is the closest to a given query point is certainly among the most fundamental problems in computational geometry. It motivated the study of crucial concepts such as multidimensional search data structures, Voronoi diagrams, dimensionality reduction, and has immediate applications in the fields of databases and machine learning. A natural generalization of this problem is to search not only for a single nearest neighbor, but rather for the nearest *combination* of a bounded number of points. More precisely, given an integer k and a query point y, we may wish to find an affine combination of k points of the set that is the nearest to y, among all possible such combinations. This problem has a natural interpretation in terms of sparse approximate solutions to linear systems, and is known as the *sparse regression*, or *sparse approximation* problem in the statistics and machine learning literature. Sparsity is defined here in terms of the ℓ_0 pseudonorm $\|.\|_0$, the number of nonzero components. The *sparse affine regression* problem can be cast as follows:

▶ Problem 1 (Sparse affine regression). Given a matrix $A \in \mathbb{R}^{d \times n}$, a vector $y \in \mathbb{R}^d$, and an integer $2 \leq k \leq d$, find $x \in \mathbb{R}^n$ minimizing $\|Ax - y\|_2$, and such that $\|x\|_0 \leq k$, and $\sum_{i=1}^n x_i = 1$.

By interpreting the columns of A as a set of n points in $\mathbb{R}^d$, the problem can be reformulated in geometric terms as the *nearest induced flat* problem.

▶ Problem 2 (Nearest induced flat). Given a set S of n points in $\mathbb{R}^d$, an additional point $y \in \mathbb{R}^d$, and an integer k such that $2 \leq k \leq d$, find k points of S such that the distance from y to their affine hull is the smallest.

* First author supported by the Fonds de la Recherche Scientifique-FNRS under CDR Grant J.0146.18. Second author supported by the VILLUM Foundation grant 16582. Part of this research was accomplished while the second author was a PhD student at ULB under FRIA Grant 5203818F (FNRS).

† A full version of this paper is available at `https://arxiv.org/abs/1908.00351` [8].

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Here the distance from a point to a flat is the distance to the closest point on the flat. Note that if we allow $k = 1$ in the definition above, we have the nearest neighbor problem as a special case. We consider the setting in which the dimension d of the ambient space as well as the number k of points in the sought combination are constant, and study the asymptotic complexity of the problem with respect to n. As observed recently by Har-Peled, Indyk, and Mahabadi [15], the problem is closely related to the classical *affine degeneracy testing* problem, defined as follows.

▶ Problem 3 (Affine degeneracy testing). Given a set S of n points in $\mathbb{R}^d$, decide whether there exist $d + 1$ distinct points of S lying on an affine hyperplane.

The latter can be cast as deciding whether a point set is in so-called *general position*, as is often assumed in computational geometry problems. In the special case $d = 2$, the problem is known to be 3SUM-hard [14, 7]. In general, it is not known whether it can be solved in time $O(n^{d-\delta})$ for some positive δ [13, 2], even for randomized algorithms. Supposing it cannot, we directly obtain a conditional lower bound on the complexity of the nearest induced flat problem. This holds even for approximation algorithms, which return an induced flat whose distance is within some bounded factor of the distance of the actual nearest flat.

▶ **Lemma 1.1** (Har-Peled, Indyk, and Mahabadi [15]). *If the nearest induced flat problem can be approximated within any multiplicative factor in time $O(n^{k-1-\delta})$ for some positive δ, then affine degeneracy testing can be solved in time $O(n^{d-\delta})$.*

Proof. Suppose we have an approximation algorithm for the nearest induced flat problem. Then given an instance of affine degeneracy testing, we can go through every point $y \in S$ and run this algorithm on an instance composed of the set $S \setminus \{y\}$, the point y, and $k = d$. The answer to the degeneracy testing instance is positive if and only if for at least one of these instances, the distance to the approximate nearest flat is zero. The running time is $O(n^{d-\delta})$. ◀

For more motivation on the subject, see the full version [8].

Our results

We prove that the nearest induced flat problem (Problem 2), can be solved within a $(1 + \varepsilon)$ approximation factor for constant d and ε in time $O(n^{k-1} \log^{d-k+2} n)$, which matches the conditional lower bound on affine degeneracy testing, up to polylogarithmic factors. Har-Peled, Indyk, and Mahabi [15] gave a data structure to preprocess a set of data points to allow solving the nearest induced flat problem on this set for any query point. Their data structure requires $\tilde{O}(n^k)$ preprocessing and $\tilde{O}(n^{k-1})$ query time. We propose an algorithm that gets rid of the preprocessing for single queries: the overall running time of our algorithm is equal to the query time of their data structure, up to polylogarithmic factors. To the best of our knowledge, this is a near-linear improvement on all previous methods for this special case.

The two main tools that are used in our algorithms are on the one hand the approximation of the Euclidean distance by a polyhedral distance, as is done in Agarwal, Rubin, and Sharir [1], and on the other hand a reduction of the decision version of the problem to orthogonal range queries. Note that orthogonal range searching data structures are also used in [15], albeit in a significantly distinct fashion.

In §2, as warm-up, we focus on the special case of Problem 2 in which $d = 3$ and $k = 2$.

Table 1 Results. For the approximation algorithms, the dependency on ε in the running time is of the order of $\varepsilon^{(1-d)/2}$. Only the details for the first result are included in this abstract. The details for the other results can be found in the full version [8].

Problem	Details	Approximation	Running Time
Problem 4: Nearest induced line in $\mathbb{R}^3$	§2	$1+\varepsilon$	$O_\varepsilon(n \log^3 n)$
Problem 2: Nearest induced flat	[8]	$1+\varepsilon$	$O_{d,\varepsilon}(n^{k-1} \log^{d-k+2} n)$
Problem 5: Nearest induced hyperplane	[8]	1	$O_{d,\delta}(n^{d-1+\delta}),\ \forall \delta > 0$
Problem 6: Nearest induced simplex	[8]	$1+\varepsilon$	$O_{d,\varepsilon}(n^{k-1} \log^d n)$

▶ Problem 4 (Nearest induced line in $\mathbb{R}^3$). Given a set S of n points in $\mathbb{R}^3$, and an additional point y, find two points $a, b \in S$ such that the distance from y to the line going through a and b is the smallest.

Our algorithm for this special case already uses all the tools that are subsequently generalized for arbitrary values of k and d. The general algorithm for the nearest induced flat problem is described in the full version [8]. In the full version, we also consider the special case of Problem 2 in which $k = d$, which can be cast as the *nearest induced hyperplane* problem.

▶ Problem 5 (Nearest induced hyperplane). Given a set S of n points in $\mathbb{R}^d$, and an additional point y, find d points of S such that the distance from y to the affine hyperplane spanned by the d points is the smallest.

For this case, we design an *exact* algorithm with running time $O(n^{d-1+\delta})$, for any $\delta > 0$. The solution solely relies on classical computational geometry tools, namely point-hyperplane duality and cuttings [11, 10].

Our algorithms can be adapted to perform *sparse linear regression*, instead of sparse affine regression. In the former, we drop the condition that the sum of the coefficients must be equal to one. This is equivalent to the nearest *linear* induced k-flat problem. It can be solved in the same time as in the affine case. To see this, realize that the problem is similar to the nearest induced flat problem where the first vertex is always the origin. The obtained complexity is the same as the one for the nearest induced flat problem.

Adapting our algorithm to sparse *convex* regression, which differs from sparse affine regression by requiring x to be positive, is a bit more involved. The sparse convex regression problem can be cast as the problem of finding the nearest simplex induced by k points of S.

▶ Problem 6 (Nearest induced simplex). Given a set S of n points in $\mathbb{R}^d$, an additional point y, and an integer k such that $2 \leq k \leq d$, find k points of S such that the distance from y to their convex hull is the smallest.

We prove that this problem can also be approximated within a $(1+\varepsilon)$ approximation factor for constant d and ε in time $O(n^{k-1} \log^d n)$, hence with an extra $O(\log^{k-2} n)$ factor in the running time compared to the affine case. This is described in the full version [8].

Our results and the corresponding sections are summarized in Table 1.

2 A $(1+\varepsilon)$-approximation algorithm for the nearest induced line problem in $\mathbb{R}^3$

We first consider the nearest induced line problem (Problem 4). We describe a near-linear time algorithm that returns a $(1+\varepsilon)$-approximation to the nearest induced line in $\mathbb{R}^3$, that is, a line at distance at most $(1+\varepsilon)$ times larger than the distance to the nearest line.

▶ **Theorem 2.1.** *For any constant $\varepsilon > 0$, there is a randomized $(1+\varepsilon)$-approximation algorithm for the nearest induced line problem in $\mathbb{R}^3$ running in time $O_\varepsilon(n \log^3 n)$ with high probability.*

The sketch of our algorithm is as follows: First, reduce the problem of minimizing the Euclidean distance to that of minimizing the polyhedral distance for some well-chosen polyhedron depending on ε. Second, reduce the problem of minimizing the polyhedral distance to that of edge-shooting. Third, reduce the problem of edge-shooting to that of deciding whether an edge shot at a certain distance would hit any induced line through some sort of binary search. Fourth, efficiently solve this decision problem using orthogonal range counting data structures.

$(1+\varepsilon)$-approximation via polyhedral distances.

The *polyhedral distance* $d_Q(y, v)$ between two points y and v with respect to a polyhedron Q centered on the origin is the smallest λ such that the dilation λQ of Q centered on y contains v, hence such that $v \in y + \lambda Q$. Our proof uses the following result, of which a weaker variant due to Dudley [12] is a major ingredient in the design of the data structure described by Agarwal, Rubin, and Sharir [1].

▶ **Lemma 2.2** (Arya, Arya, da Fonseca, Mount [3])**.** *For any positive integer d and positive real ε, there exists a d-dimensional polyhedron Q with $O(1/\varepsilon^{(d-1)/2})$ faces such that for every $y, v \in \mathbb{R}^d$:*

$$\|y - v\|_2 \leq d_Q(y, v) \leq (1+\varepsilon) \cdot \|y - v\|_2 .$$

This bound is asymptotically optimal. See [4, 6, 5] for more details.

Next, we reduce Problem 4 to a counting problem in two steps.

Edge-shooting.

We use Lemma 2.2 for $d = 3$. We give an exact algorithm for computing the nearest induced line with respect to a polyhedral distance d_Q, where Q is defined from ε as in Lemma 2.2. Given a polyhedron Q, one can turn it into a simplicial polyhedron by triangulating it. Therefore, for constant values of ε, this reduces the problem to a constant number of instances of the *edge-shooting problem*, defined as follows: Given an edge e of Q, find the smallest value λ such that $y + \lambda e$ intersects a line through two points of S. We iterate this for all edges of Q, and pick the minimum value. This is exactly the polyhedral distance from y to its nearest induced line.

Binary search.

Using a randomized binary search procedure, we reduce the edge-shooting problem to a *counting problem*, defined as follows: given the triangle Δ defined as the convex hull of y and $y + \lambda e$, count how many pairs of points $a, b \in S$ are such that the line $\ell(a, b)$ through them intersects Δ. Suppose there exists a procedure for solving this problem. We can use this procedure to solve the edge-shooting problem efficiently as follows.

First initialize λ to some upper bound on the distance. Then count how many lines $\ell(a, b)$ intersect Δ, using the procedure. If there is only one, then return its (polyhedral) distance to y. Otherwise, pick one such line uniformly at random and compute the value λ' such that this line intersects $y + \lambda' e$. Then iterate the previous steps with $\lambda \leftarrow \lambda'$, unless $\lambda' = 0$ in which case we return 0. Since we picked the line at random, and since there are $O(n^2)$ such

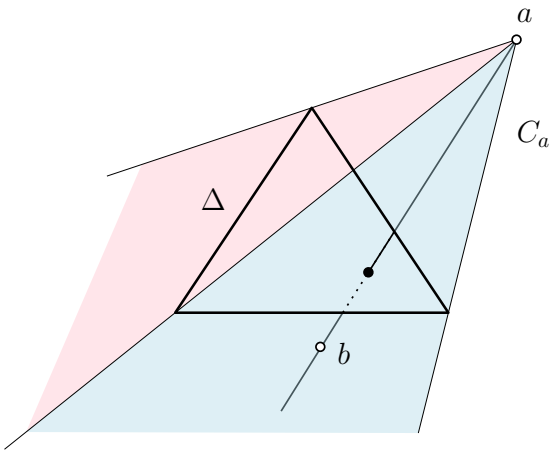


Figure 1 The cone C_a.

lines at the beginning of the search, the number of iterations of this binary search is $O(\log n)$ with high probability.

We therefore reduced the nearest induced line problem to $O(\varepsilon^{-1} \log n)$ instances of the counting problem.

Orthogonal range counting queries.

Data structures for *orthogonal range counting queries* store a set of points in $\mathbb{R}^g$ in such a way that the number of points in a given g-rectangle (cartesian product of g intervals) can be returned quickly. Known data structures for orthogonal range counting queries in $\mathbb{R}^g$ require $O(n \log^{g-1} n)$ preprocessing time and can answer queries in $O(\log^{g-1} n)$ time [16, 9]. Note that the actual coordinates of the points do not matter: We only need to know the order of their projections on each axis. We now show how to solve the counting problem using a data structure for orthogonal range queries in $\mathbb{R}^3$.

Let us fix the triangle Δ and a point $a \in \mathbb{R}^3$, and consider the locus of points $b \in \mathbb{R}^3$ such that the line $\ell(a, b)$ intersects Δ. This is a double simplicial cone with apex a and whose boundary contains the boundary of Δ. This double cone is bounded by three planes, one for each edge of Δ. In fact, we will only consider one of the two cones, because $\ell(a, b)$ intersects Δ if and only if either b is contained in the cone of apex a, or a is contained in the cone of apex b. Let us call C_a the cone of apex a. This is illustrated on Figure 1.

Let us consider one edge f of Δ and all the planes containing f. These planes induce a circular order on the points of S, which is the order in which they are met by a plane rotating around the supporting line of f. This is illustrated on Figure 2. Now let us denote by H_f the plane containing a and f and by H_f^+ the halfspace bounded by H_f and containing Δ. The set of points of S contained in H_f^+ is an interval in the circular order mentioned above. Hence the set of points contained in C_a is the intersection of three intervals in the three circular orders defined by the three edges of Δ.

Proof of Theorem 2.1. Let Q be some polyhedron in $\mathbb{R}^3$, $\lambda \in \mathbb{R}$, $S \subset \mathbb{R}^3$, $y \in \mathbb{R}^3$, and e an edge of Q. We use an orthogonal range counting data structure for storing the points of S with coordinates corresponding to their ranks in each of the three permutations induced by the three edges of $\Delta = \text{conv}(\{ y, y + \lambda e \})$. We get those rank-coordinates by sorting S three

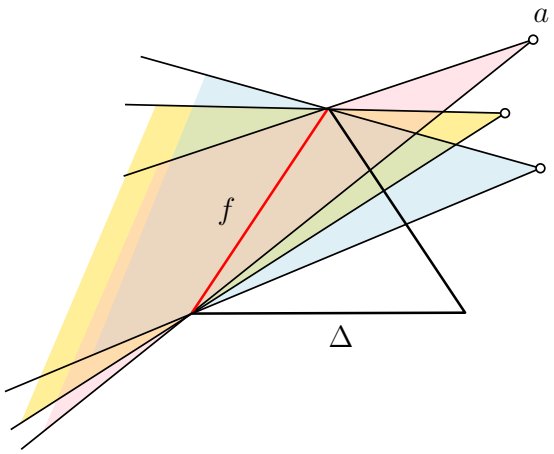


Figure 2 The order of the points defined by the planes containing an edge f of Δ.

times, once for each induced permutation, in time $O(n \log n)$, then construct the orthogonal range counting data structure with those coordinates in time $O(n \log^2 n)$. Then for each of the n points $a \in S$, we count the number of points b in the cone C_a by querying the data structure in $O(\log^2 n)$ time. Hence overall, the counting problem is solved in time $O(n \log^2 n)$. Note that the circularity of the order can be easily handled by doubling every point.

This can be combined with the previous reductions provided we can choose a line intersecting Δ uniformly at random within that time bound. This is achieved by first choosing a with probability proportional to the number of points b such that $\ell(a, b) \cap \Delta \neq \emptyset$. Then we can pick a point b uniformly at random in this set in linear time.

Combining with the previous reductions, we obtain an approximation algorithm running in time $O_\varepsilon(n \log^3 n)$ for the nearest induced line problem in $\mathbb{R}^3$. ◀

References

1 Pankaj K. Agarwal, Natan Rubin, and Micha Sharir. Approximate nearest neighbor search amid higher-dimensional flats. In *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria*, pages 4:1–4:13, 2017. doi:10.4230/LIPIcs.ESA.2017.4.

2 Nir Ailon and Bernard Chazelle. Lower bounds for linear degeneracy testing. *J. ACM*, 52(2):157–171, 2005. doi:10.1145/1059513.1059515.

3 Rahul Arya, Sunil Arya, Guilherme D. da Fonseca, and David M. Mount. Optimal bound on the combinatorial complexity of approximating polytopes. In *SODA*, pages 786–805. SIAM, 2020.

4 Sunil Arya, Guilherme D. da Fonseca, and David M. Mount. On the combinatorial complexity of approximating polytopes, 4 2016. arXiv:1604.01175v4.

5 Sunil Arya, Guilherme D. da Fonseca, and David M. Mount. Approximate convex intersection detection with applications to width and minkowski sums, 7 2018. arXiv:1807.00484v1.

6 Sunil Arya, Guilherme Dias da Fonseca, and David M. Mount. Near-optimal epsilon-kernel construction and related problems. In *Symposium on Computational Geometry*, volume 77 of *LIPIcs*, pages 10:1–10:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

7 Luis Barba, Jean Cardinal, John Iacono, Stefan Langerman, Aurélien Ooms, and Noam Solomon. Subquadratic algorithms for algebraic 3SUM. *Discrete & Computational Geometry*, 61(4):698–734, 2019. doi:10.1007/s00454-018-0040-y.

8 Jean Cardinal and Aurélien Ooms. Sparse regression via range counting, 8 2019. arXiv:1908.00351.

9 Bernard Chazelle. A functional approach to data structures and its use in multidimensional searching. *SIAM J. Comput.*, 17(3):427–462, 1988. doi:10.1137/0217026.

10 Bernard Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete & Computational Geometry*, 9:145–158, 1993. doi:10.1007/BF02189314.

11 Bernard Chazelle and Joel Friedman. A deterministic view of random sampling and its use in geometry. *Combinatorica*, 10(3):229–249, 1990. doi:10.1007/BF02122778.

12 Richard M. Dudley. Metric entropy of some classes of sets with differentiable boundaries. *Journal of Approximation Theory*, 10(3):227 – 236, 1974. URL: http://www.sciencedirect.com/science/article/pii/0021904574901208, doi:https://doi.org/10.1016/0021-9045(74)90120-8.

13 Jeff Erickson and Raimund Seidel. Better lower bounds on detecting affine and spherical degeneracies. *Discrete & Computational Geometry*, 13:41–57, 1995. doi:10.1007/BF02574027.

14 Anka Gajentaan and Mark H. Overmars. On a class of $o(n^2)$ problems in computational geometry. *Comput. Geom.*, 5:165–185, 1995. doi:10.1016/0925-7721(95)00022-2.

15 Sariel Har-Peled, Piotr Indyk, and Sepideh Mahabadi. Approximate sparse linear regression. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 77:1–77:14, 2018. doi:10.4230/LIPIcs.ICALP.2018.77.

16 Dan E. Willard. New data structures for orthogonal range queries. *SIAM J. Comput.*, 14(1):232–253, 1985.

The Very Best of Perfect Non-crossing Matchings*†

Ioannis Mantas[1], Marko Savić[2], and Hendrik Schrezenmaier[3]

1 **Faculty of Informatics, Università della Svizzera italiana, Lugano, Switzerland**
ioannis.mantas@usi.ch
2 **Faculty of Informatics, Department of Mathematics and Informatics, Faculty of Sciences, University of Novi Sad, Serbia**
marko.savic@dmi.uns.ac.rs
3 **Institut für Mathematik, Technische Universität Berlin, Germany**
schrezen@math.tu-berlin.de

Abstract

Given a set of points in the plane, we are interested in matching them with straight line segments. We focus on perfect (all points are matched) non-crossing (no two edges intersect) matchings. Apart from the well known MinMax variation, where the length of the longest edge is minimized, we extend work by looking into three new optimization variants such as MaxMin, MinMin, and MaxMax. We consider both the monochromatic and bichromatic versions of these problems and provide efficient algorithms for various input point configurations.

1 Introduction

In the *matching problem*, we are given a set of objects and the goal is to partition the set into pairs such that no object belongs to two pairs. This simple problem is a classic in graph theory, which has received a lot of attention, both in an abstract and in a geometric setting.

In this paper, we consider the geometric setting where given a set P of $2n$ points in the plane, the goal is to match points of P with straight line segments, in the sense that each pair of points induces an *edge* of the matching. A matching is *perfect* if it consists of exactly n pairs. A matching is *non-crossing* if the edges of the matching are pairwise disjoint. When there are no restrictions on which points can be matched, the problem is called *monochromatic*. In the *bichromatic* variant, P is partitioned into two sets B and R of blue and red points, respectively, and only points of different colors are allowed to be matched. When $|B| = |R| = n$, the point set P is called *balanced*.

Related work on perfect non-crossing matchings. Geometric matchings find applications in various areas, as in operations research, in the field of shape matching, in pattern recognition, in VLSI design problems and map construction or comparison algorithms among others. In any application, requiring the matching to be non-crossing or perfect, seems natural. Given a point set, monochromatic or balanced bichromatic, a perfect non-crossing matching always exists and it can be found in $O(n \log n)$ time by recursively computing *ham-sandwich cuts* [13] or by using the algorithm of Hershberger and Suri [12].

* I. M. was partially supported by the Swiss National Science Foundation, project SNF 200021E-154387, M. S. was partially supported by Ministry of Education, Science and Technological Development, Republic of Serbia, project 174019, and H. S. was partially supported by the German Research Foundation, DFG grant FE-340/11-1.

† A full version of this extended abstract is available at https://arxiv.org/abs/2001.03252.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

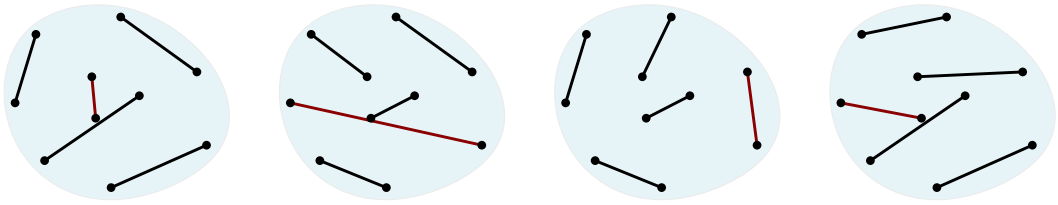


Figure 1 Optimal MinMin1, MaxMax1, MinMax1, and MaxMin1 matching of a point set.

A well-studied optimization criterion is minimizing the sum of lengths of all edges, referred to as MinSum, and also known as the *Euclidean assignment* or *matching* problem. It is interesting, and easy to see, that such a matching is always non-crossing. For monochromatic point sets, an $O(n^{3/2} \log n)$-time algorithm was given by Varadarajan [19]. For bichromatic point sets, Agarwal et al. [2] presented an $O(n^{2+\varepsilon})$-time algorithm. When points are in convex position, Marcotte and Suri [14] solved the problem in $O(n \log n)$ time in both settings.

Another popular goal is to minimize the length of the longest edge, which we refer to as the MinMax variant and is also known as the *bottleneck matching*. For monochromatic point sets, Abu-Affash et al. [1] showed that finding such a matching is $\mathcal{NP}$-hard and accompanied this with an $O(n^3)$-time algorithm for points in convex position. This was recently improved to $O(n^2)$ time by Savić and Stojaković [15]. For bichromatic point sets, Carlsson et al. [7] showed that finding such a matching is also $\mathcal{NP}$-hard. Biniaz et al. [6] gave an $O(n^3)$-time algorithm for points in convex position and an $O(n \log n)$-algorithm for points on a circle. These were recently improved to $O(n^2)$ and $O(n)$, respectively, by Savić and Stojaković [16].

More optimization goals have been studied, as the *uniform* or *fair matching*, where the goal is to minimize the length difference between the longest and the shortest edge, and the *minimum deviation matching*, where the difference between the shortest edge length and the average edge length should be minimized. Both can be solved in polynomial time, see Efrat et al. [10, 11]. Another example is the MaxSum variant, where the goal is to maximize the sum of the edge lengths. Alon et al. [4] conjectured that the problem is $\mathcal{NP}$-hard.

Problem variants considered and our contribution. We continue exploring similar optimization variants in different settings. We consider solely perfect non-crossing matchings, without further mention. We deal with four variants: MinMin where the length of the shortest edge is minimized, MaxMax where the length of the longest edge is maximized, MaxMin where the length of the shortest edge is maximized, and MinMax, see Fig. 1.

To the best of our knowledge, out of the four variants, only MinMax, has been considered before. Apart from the theoretical interest, MinMin and MaxMax are motivated by worst-case analyses of matchings, where there is no control over the choice of edges, and short or long edges are undesirable. Apart from the applications of MinMax, together with MaxMin, such matchings resemble fair matchings in the sense that all edges have similar lengths.

We investigate both the monochromatic and bichromatic versions of these variants. In the bichromatic version, we assume that P is balanced. We denote the monochromatic problems with the index 1, e.g., MinMin1, and the bichromatic with the index 2, e.g., MinMin2.

These problems are examined in different point configurations. In Section 2, we consider points in general position. In Section 3, points are in convex position. In Section 4, points lie on a circle. In Section 5, we consider *doubly collinear* bichromatic point sets, where the blue points lie on one line and the red points on another line. By studying structural properties of each variant and combining diverse techniques, we come up with a series of interesting results that are summarized in Table 1.

Table 1 The running times for finding the values of optimal matchings regarding different objective functions. The times marked with (*) indicate the extra time needed to return also the matching. h denotes the size of the convex hull. Results without reference are given in this paper.

Monochromatic	**MinMin1**	**MaxMax1**	**MinMax1**	**MaxMin1**
General Position	$O(nh + n \log n)$	$O(nh) + O(n \log n)^*$	$\mathcal{NP}$-hard [1]	?
Convex Position	$O(n)$	$O(n)$	$O(n^2)$ [15]	$O(n^3)$
Points on circle	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Bichromatic	**MinMin2**	**MaxMax2**	**MinMax2**	**MaxMin2**
General Position	?	?	$\mathcal{NP}$-hard [7]	?
Convex Position	$O(n)$	$O(n)$	$O(n^2)$ [16]	$O(n^3)$
Points on circle	$O(n)$	$O(n)$	$O(n)$ [16]	$O(n^3)$
Doubly collinear	$O(n)$	$O(1) + O(n)^*$	$O(n^4 \log n)$	?

2 Monochromatic points in general position

In this section, P is a set of points in general position, where we assume that no three points are collinear. We denote by $\mathrm{CH}(P) \subseteq P$ the set of points on the boundary of the convex hull of P and set $h := |\mathrm{CH}(P)|$. By $d(v, w)$ we denote the Euclidean distance of two points v, w.

An edge (v, w) of points is *feasible* if there exists a matching which contains (v, w).

▶ **Lemma 2.1.** *An edge (v, w) is infeasible if and only if $v, w \in \mathrm{CH}(P)$ and there is an odd number of points on each side of the line through (v, w). See Fig. 2a.*

This criterion can be checked efficiently using the *radial orderings* of the points $p \in P$, i.e., the circular orderings of the points in $P \setminus p$ by angle around p. The radial orderings of all points in P can be computed in $O(n^2)$ time, see, e.g., [3, 5]. Instead, we define the *weak radial orderings*. Given a set $A \subseteq P$, in the *A-weak radial ordering* of p the points from A occurring between two points from $\overline{A} := P \setminus A$ are given as an unordered set, see Fig. 2b-c.

▶ **Lemma 2.2.** *The $\overline{A}$-weak radial orderings of all points in A can be computed in $O(n|A|)$ time.*

MinMin1. We are looking for the shortest feasible edge. We first compute $\mathrm{CH}(P)$ in $O(n \log h)$ time [8] and the $\overline{\mathrm{CH}(P)}$-weak radial orderings of points in $\mathrm{CH}(P)$ in $O(nh)$ time. Then, we find $m_1 := \min(\{\, d(v, w) \colon v \in P \setminus \mathrm{CH}(P), w \in P \,\})$ in $O(n \log n)$ time using a Voronoi diagram and $m_2 := \min(\{\, d(v, w) \colon v, w \in \mathrm{CH}(P) \,\})$ in $O(nh)$ time using weak radial orderings. The solution is then $m_{\mathrm{sol}} = \min(m_1, m_2)$, resulting in the following theorem.

▶ **Theorem 2.3.** MinMin1 *can be solved in $O(nh + n \log n)$ time.*

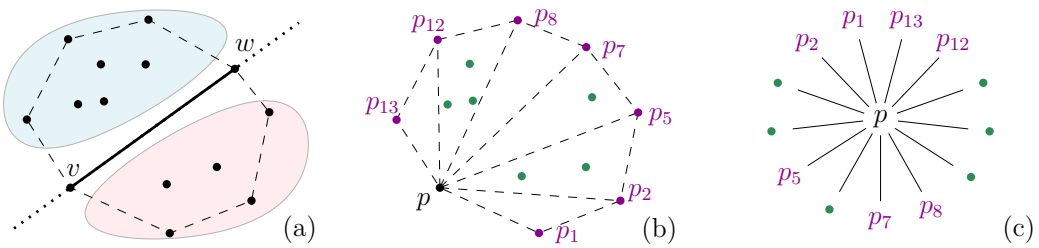


Figure 2 (a) An infeasible edge (v, w). (b-c) The $\overline{\mathrm{CH}(P)}$-weak radial ordering of a point p.

MaxMax1. We can use the same $O(nh+n\log n)$-time algorithm, considering maximizations in m_1, m_2, and m_{sol} instead of minimizations. Using Lemma 2.4 we can reduce the time needed to find m_1 in $O(nh)$, by simply comparing all $(n-h)h$ edges. This results in Theorem 2.5.

▶ **Lemma 2.4.** *If (v,w) is a longest feasible edge, then one of $v,w \in \mathrm{CH}(P)$.*

▶ **Theorem 2.5.** MaxMax1 *can be solved in $O(nh)$ time.*

3 Points in convex position

In this section, we assume that points in P are in convex position with the counterclockwise ordering, $p_0,\ldots,p_{2n-1}$, given. We address a point p_i by its index i, and do arithmetic operations modulo $2n$. We call edges of the form $(i,i+1)$ *boundary edges.*

Dynamic programming. We can easily solve all four optimization variants in $O(n^3)$ time using a classic dynamic programming approach, which is also used for MinMax [1, 6, 7].

▶ **Theorem 3.1.** *If P is convex,* MaxMin1 *and* MaxMin2 *can be solved in $O(n^3)$ time.*

3.1 MinMin and MaxMax matchings in convex position

Monochromatic. We split P into two (convex) sets, P_{odd} and P_{even}, according to the parity of the indices. A pair (i,j) is feasible if and only if i and j are of different parity [1]. Hence, any (i,j) with $i \in P_{\text{odd}}$ and $j \in P_{\text{even}}$ is feasible. Given two convex sets P,Q, we can find in $O(|P|+|Q|)$ time the points that realize the minimum [18] and the maximum [9] distance between them. Applying these algorithms to P_{odd} and P_{even}, we obtain the following.

▶ **Theorem 3.2.** *If P is convex,* MinMin1 *and* MaxMax1 *can be solved in $O(n)$ time.*

Bichromatic. We now combine the monochromatic algorithms with the theory of *orbits* [16], a concept which captures well the nature of bichromatic matchings in convex position. More specifically, P can be partitioned in $O(n)$ time into orbits, which are balanced sets of points. In each orbit point colors alternate and a bichromatic edge (b,r) is feasible if and only if b and r are in the same orbit. Thus, to each orbit separately we can use the algorithms of [9, 18] and return the overall longest or shortest edge, see Fig. 3, resulting in the following.

▶ **Theorem 3.3.** *If P is convex,* MinMin2 *and* MaxMax2 *can be solved in $O(n)$ time.*

Given an extremal feasible edge, we can extend it to an optimal matching, in $O(n)$ time, by applying the following lemma to the sets $\{i+1,\ldots,j-1\}$ and $\{j+1,\ldots,i-1\}$.

▶ **Lemma 3.4.** *If P is in convex position, we can construct an arbitrary matching in $O(n)$ time, both in the monochromatic and bichromatic case.*

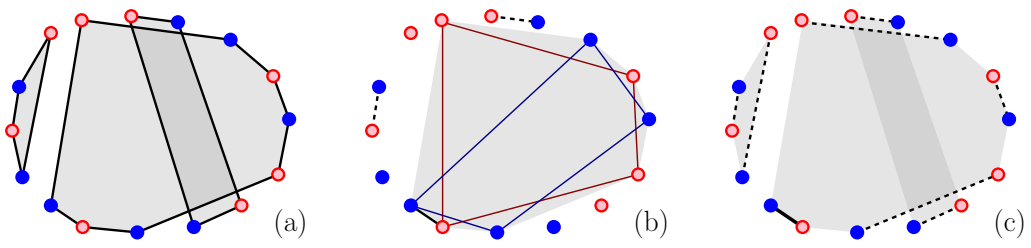


Figure 3 MinMin2 for a set in convex position. (a) Find orbits. (b) Find the shortest edge between the blue and red polygon of an orbit. (c) Extend to a perfect matching using Lemma 3.4.

4 Points on a circle

Next, we assume that all points of P lie on a circle. Obviously, the results from Section 3 also apply here. Apart from convexity, these results rely only on a property of point sets lying on a circle, which we call the *decreasing chords property*. A point set P has this property if, for any edge (i, j), for at least one of its sides, all the possible edges between two points on that side are not longer than (i, j) itself, see Fig. 4a. Using this, we can easily infer the following.

▶ **Lemma 4.1.** *Any shortest edge of a matching on P is a boundary edge.*

MinMin1 and MinMin2. Due to Lemma 4.1, these can be solved in $O(n)$ time significantly simpler, without using Theorems 3.2 and 3.3, by finding the shortest feasible boundary edge.

MinMax1. We show that there exists a MinMax1 matching using only boundary edges. There are two such matchings and we find the optimal in $O(n)$ time, resulting in the following.

▶ **Theorem 4.2.** *If P lies on a circle,* MinMax1 *can be solved in $O(n)$ time.*

MaxMin1. Lemma 4.1 suggests an approach for MaxMin1 by *forbidding* short boundary edges and checking if we can find a matching without them. Let some boundary edges be *forbidden* and the remaining be *allowed*. A *forbidden chain* is a maximal sequence of consecutive forbidden edges. See Fig. 4b for an example (with forbidden edges shown dashed).

▶ **Lemma 4.3.** *There exists a matching without the forbidden edges if and only if $l < n$, where l is the length of a longest forbidden chain.*

MaxMin1 is equivalent to finding the largest value μ such that there exists a matching with all edges of length at least μ. By Lemma 4.1, it suffices to search for μ among the lengths of the boundary edges. By Lemma 4.3, this means that we need to find the maximal length μ of a boundary edge such that there are no n consecutive boundary edges all shorter than μ. We can find μ as follows. Consider all $2n$ sets of n consecutive boundary edges and associate to each set the longest edge in it. Then, out of the $2n$ longest edges, we search for the shortest one. This fits under the *sliding window maximum problem*, for which several simple algorithms are known, see, e.g., [17]. Adapting to our problem we obtain the following.

▶ **Theorem 4.4.** *If P lies on a circle,* MaxMin1 *can be solved in $O(n)$ time.*

Using Lemma 4.5, we can construct an optimal matching within the same time complexity.

▶ **Lemma 4.5.** *Given a value $\mu > 0$, a matching consisting of edges of length at least μ can be constructed in $O(n)$ time if it exists.*

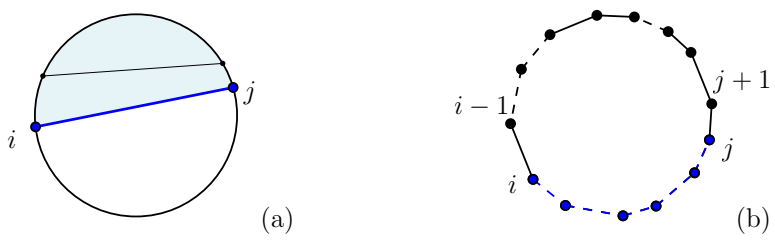


Figure 4 (a) The decreasing chords property. (b) Example of a forbidden chain $\{i, \ldots, j\}$.

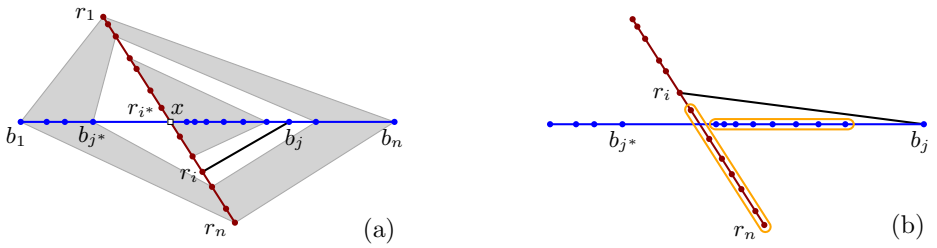


Figure 5 A doubly collinear point set, with (a) a feasible and (b) an infeasible edge shown.

5 Doubly collinear points

A bichromatic point set P is *doubly collinear* if the blue points lie on a line l_B and the red points lie on a line l_R. We assume that l_B and l_R are not parallel and that the ordering of the points along each line is given. Let $x = l_B \cap l_R$ and assume, for simplicity, that $x \notin P$.

5.1 MinMin2 and MaxMax2 matchings on doubly collinear points

We first give a feasibility criterion for an edge, see Fig. 5, which can be checked in $O(1)$ time. It also indicates an $O(n)$-time algorithm, to construct a matching, given a feasible edge (r, b).

▶ **Lemma 5.1.** *Let $r_1, \ldots, r_i, \ldots, r_{i^*}, x, r_{i^*+1}, \ldots, r_n$ and $b_1, \ldots, b_j, \ldots, b_{j^*}, x, b_{j^*+1}, \ldots, b_n$ be the points on l_R and l_B, respectively, in sorted order. Then, the edge (r_i, b_j) is feasible if and only if $i^* - i \leq n - j$ and $j^* - j \leq n - i$.*

As a consequence of Lemma 5.1, the order of the closest feasible neighbors of the red points on the line l_B coincides with the order of the red points on l_R. Thus, we can find the closest feasible neighbor of all red points in total $O(n)$ time, implying the following theorem.

▶ **Theorem 5.2.** *If P is doubly collinear,* MinMin2 *can be solved in $O(n)$ time.*

The same algorithm solves MaxMax2. We show that the longest feasible edge is an edge between the, at most four, points on $\mathrm{CH}(P)$, so we can further improve as follows.

▶ **Theorem 5.3.** *If P is doubly collinear,* MaxMax2 *can be solved in $O(1)$ time.*

5.2 MinMax2 and MaxMin2 matchings on doubly collinear points

One-sided case. We first consider the case, where all red points are on one side of l_B. This can be solved via a dynamic program with $O(n^2)$ subproblems in total, see Fig. 6.

▶ **Theorem 5.4.** *If P is one-sided doubly collinear,* MaxMin2 *can be solved in $O(n^2)$ time.*

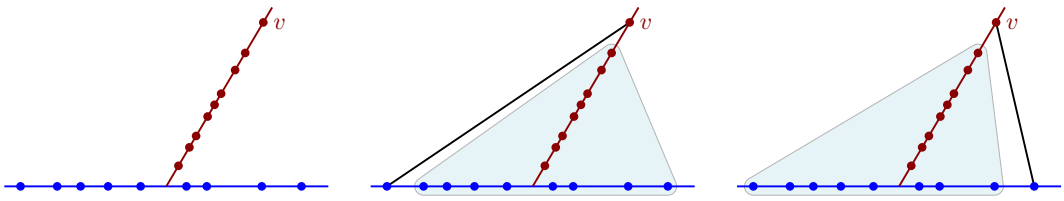


Figure 6 The one-sided case with the two possibilities to match v and the resulting subproblems.

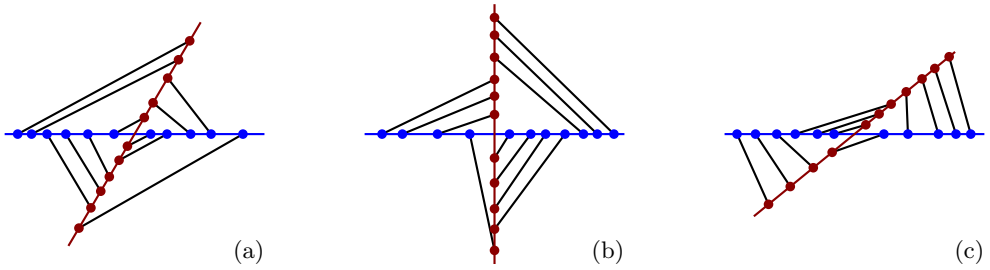


Figure 7 Optimal matchings having a special structure (a) for MinMax2, (b) for MinMax2 and MaxMin2 if $\alpha = \frac{\pi}{2}$, and (c) for MinMax2 if $\alpha < \frac{\pi}{4}$.

In the case of MinMax2 we can further improve upon this, by proving (also for the two-sided case) the existence of an optimal matching with a special form where the points are partitioned into a constant number of blocks and these blocks are matched, see Fig. 7a.

▶ **Theorem 5.5.** *If P is one-sided doubly collinear,* MinMax2 *can be solved in $O(n \log n)$ time.*

General case. For MinMax2, the aforementioned form can also be applied to the general case, yielding Theorem 5.6. On the contrary, for MaxMin2 we are not aware if an optimal matching with a special form exists. Thus, we do not know a polynomial time algorithm.

▶ **Theorem 5.6.** *If P is doubly collinear,* MinMax2 *can be solved in $O(n^4 \log n)$ time.*

Special intersection angle. Let α be the angle of intersection of l_B and l_R. By proving the existence of optimal matchings of a special form, see Fig. 7b-c, we can obtain the following.

▶ **Theorem 5.7.** *If $\alpha = \frac{\pi}{2}$,* MinMax2 *and* MaxMin2 *can be solved in $O(n)$ time.*

▶ **Theorem 5.8.** *If $\alpha \leq \frac{\pi}{4}$,* MinMax2 *can be solved in $O(n)$ time.*

6 Concluding remarks

It comes as no surprise that the MaxMin variant exhibits this significant difficulty; devising efficient algorithms even for simple configurations is not at all obvious and, hence, interesting on its own. On the contrary, the MinMin and the MaxMax variants are *relatively easier* to tackle; we managed to design optimal algorithms by exploiting structural properties combined with existing techniques from diverse problems. We conclude with some open questions, hoping to see Table 1 filled in. For bichromatic P in general position, can we check the feasibility of an edge in polynomial time? This would imply polynomial time algorithms for MinMin2 and MaxMax2. For P in convex position, do there exist $o(n^3)$-time algorithms for MaxMin? Maybe by using the theory of orbits for the bichromatic case? For P in general position, can MaxMin be solved in polynomial time? What if P is doubly collinear?

Acknowledgements. Preliminary results were obtained during the IRP - DCCG (Barcelona, 4-6/2018). We are grateful to CRM, UAB for hosting the event and to the organizers for providing us with the platform to meet and collaborate. We are also grateful to Carlos Alegría, Carlos Hidalgo Toscano, Oscar Iglesias Valiño, and Leonardo Martínez Sandoval for initial discussions, and to Carlos Seara for raising a question that motivated this work.

References

1. A. Karim Abu-Affash, Paz Carmi, Matthew J. Katz, and Yohai Trabelsi. Bottleneck non-crossing matching in the plane. *Computational Geometry*, 47(3A):447–457, 2014.
2. Pankaj K. Agarwal, Alon Efrat, and Micha Sharir. Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications. *SIAM Journal on Computing*, 29(3):912–953, 2000.
3. Pankaj K. Agarwal and Micha Sharir. Arrangements and their applications. In *Handbook of Computational Geometry*, chapter 2, pages 49–119. North-Holland, 2000.
4. Noga Alon, Sridhar Rajagopalan, and Subhash Suri. Long non-crossing configurations in the plane. In *Proc. 9th Annual Symposium on Computational Geometry*, pages 257–263, 1993.
5. Tetsuo Asano, Subir K. Ghosh, and Thomas C. Shermer. Visibility in the plane. In *Handbook of Computational Geometry*, chapter 19, pages 829–876. North-Holland, 2000.
6. Ahmad Biniaz, Anil Maheshwari, and Michiel H.M. Smid. Bottleneck bichromatic plane matching of points. In *Proc. 26th Canadian Conference on Computational Geometry*, pages 431–435, 2014.
7. John G. Carlsson, Benjamin Armbruster, Saladi Rahul, and Haritha Bellam. A bottleneck matching problem with edge-crossing constraints. *International Journal of Computational Geometry & Applications*, 25(4):245–261, 2015.
8. Timothy M. Chan. Optimal output-sensitive convex hull algorithms in two and three dimensions. *Discrete & Computational Geometry*, 16(4):361–368, 1996.
9. Herbert Edelsbrunner. Computing the extreme distances between two convex polygons. *Journal of Algorithms*, 6(2):213–224, 1985.
10. Alon Efrat, Alon Itai, and Matthew J. Katz. Geometry helps in bottleneck matching and related problems. *Algorithmica*, 31(1):1–28, 2001.
11. Alon Efrat and Matthew J. Katz. Computing fair and bottleneck matchings in geometric graphs. In *Proc. 7th International Symposium on Algorithms and Computation*, pages 115–125. Springer, 1996.
12. John Hershberger and Subhash Suri. Applications of a semi-dynamic convex hull algorithm. *BIT Numerical Mathematics*, 32(2):249–267, 1992.
13. Chi-Yuan Lo, Jiří Matoušek, and William Steiger. Algorithms for ham-sandwich cuts. *Discrete & Computational Geometry*, 11(4):433–452, 1994.
14. Odile Marcotte and Subhash Suri. Fast matching algorithms for points on a polygon. *SIAM Journal on Computing*, 20(3):405–422, 1991.
15. Marko Savić and Miloš Stojaković. Faster bottleneck non-crossing matchings of points in convex position. *Computational Geometry*, 65:27–34, 2017.
16. Marko Savić and Miloš Stojaković. Bottleneck bichromatic non-crossing matchings using orbits, 2018. URL: `arxiv.org/abs/1802.06301`, `arXiv:1802.06301`.
17. Kanat Tangwongsan, Martin Hirzel, and Scott Schneider. Low-latency sliding-window aggregation in worst-case constant time. In *Proc. 11th ACM International Conference on Distributed and Event-based Systems*, pages 66–77, 2017.
18. Godfried T. Toussaint. An optimal algorithm for computing the minimum vertex distance between two crossing convex polygons. *Computing*, 32(4):357–364, 1984.
19. Kasturi R. Varadarajan. A divide-and-conquer algorithm for min-cost perfect matching in the plane. In *Proc. 39th Symposium on Foundations of Computer Science*, pages 320–329, 1998.

One-Bend Drawings of Outerplanar Graphs Inside Simple Polygons

Patrizio Angelini[1], Philipp Kindermann[2], Andre Löffler[2], Lena Schlipf[3], and Antonios Symvonis[4]

1 **John Cabot University, Rome, Italy**
pangelini@johncabot.edu
2 **Universität Würzburg, Würzburg, Germany**
philipp.kindermann@uni-wuerzburg.de, andre.loeffler@uni-wuerzburg.de
3 **Universität Tübingen, Tübingen, Germany**
schlipf@informatik.uni-tuebingen.de
4 **National Technical University of Athens, Athens, Greece**
symvonis@math.ntua.gr

Abstract

We consider the problem of drawing an outerplanar graph with n vertices with at most one bend per edge if the outer face is already drawn as a simple polygon with m corners. We prove that it can be decided in $O(mn)$ time if such a drawing exists. In the positive case, our algorithm also outputs such a drawing.

1 Introduction

One of the fundamental problems in graph drawing is to draw a planar graph crossing-free under certain geometric or topological constraints. Many classical algorithms draw planar graphs under the constraint that all edges have to be straight-line segments [4, 14, 15]. But we do not always have the freedom of drawing the whole graph from scratch. In practical applications, parts of the graph may already be drawn and we want to extend it to a planar drawing of the whole graph. For example, in visualizations of large networks, certain patterns may be required to be drawn in a standard way, or a social network may be updated as new people enter a social circle or as new links emerge between already existing persons.

This problem is known as the Partial Drawing Extensibility problem. Formally, given a planar graph $G = (V, E)$, and subgraph $H = (V', E')$ with $V' \subseteq V$ and $E' \subsetneq E$, and a planar drawing Γ_H of H, the problem asks for a planar drawing Γ_G of G such that the drawing of H in Γ_G coincides with Γ_H. This problem was first proposed by Brandenburg et al. [2] and has received a lot of attention in the previous years.

Related work. For the case of straight-line drawings, Patrignani showed the problem to be NP-hard [12], but he could not prove membership in NP, as a solution may require coordinates not representable with a polynomial number of bits. Recently, Lubiw et al. [8] proved that a generalization of the problem where overlaps (but not proper crossings) between edges of $E \setminus E'$ and E' are allowed is hard for the existential theory of the reals ($\exists\mathbb{R}$-hard).

These results motivate allowing bends in the drawing. Angelini et al. [1] presented a linear-time algorithm to test whether there exists any topological planar drawing of G, and Jelínek et al. [7] gave a characterization via forbidden substructures. Chan et al. [3] showed that a linear number of bends ($72|V'|$) per edge suffices, which is also worst-case optimal as shown by Pach and Wenger [11] for the special case that $E' = \emptyset$.

Special attention has been given to the case that H is exactly the outer face of G. Already Tutte's seminal paper [15] showed how to obtain a straight-line convex drawing of a

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

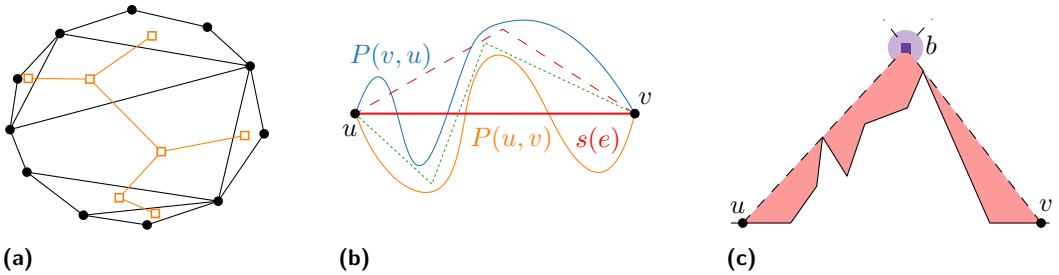


Figure 1 (a) A biconnected outerplanar graph (in black) and the dual tree (in orange); (b) for an edge $e = (u, v)$, the straight line $s(e)$ intersects both $P(u, v)$ and $P(v, u)$, the dashed red 1-bend drawing of e only avoids crossing $P(u, v)$, possible 2-bend drawing in green; (c) the edge connecting u and v has to cut away at least the red region, b is the minimal bend point, and the modified polygon has a reflex angle at b.

triconnected planar graph with its outer face drawn as a prescribed convex polygon. This result has been extended by Hong and Nagamochi [6] to the case that the outer face is drawn as a star-shaped polygon without chords (that is, interior edges between outer vertices). Mchedlidze et al. [9] gave a linear-time testing algorithm for the existence of a straight-line drawing of G in the case that H is an arbitrary cycle of G and is drawn as a convex polygon, while Mchedlidze and Urhausen [10] study the number of bends required based on the shape of the drawing of H and show that 1 bend suffices if H is drawn as a star-shaped polygon.

Our contribution. In this paper, we consider the case that G is an outerplanar graph and H is exactly the outer face of G, which is drawn as a simple polygon P with arbitrarily many bends between its vertices. Note that G has to be biconnected for its outer face to be a simple cycle. For any constant number k of bends, there exists some instance such that G has a k-bend drawing but no $(k-1)$-bend drawing; see, e.g., Fig. 1b for $k = 2$. Hence, it is of interest to test for a given k whether a k-bend drawing of G exists. This is trivial for $k = 0$. In this paper, we prove that for $k = 1$ the problem can be solved in time $O(mn)$, where n is the number of vertices in G and m is the number of corners of P.

Notation. We assume that we are given a biconnected outerplanar graph $G = (V, E)$, a simple polygon P with boundary ∂P, and an injective mapping of V to ∂P such that ∂P coincides with a plane drawing of the outer face H of G with arbitrarily many bends per edge. We say that G *can be drawn in* P if there is a crossing-free drawing of G with its vertices on ∂P as defined by the mapping, its outer face drawn as ∂P, and its interior edges drawn with at most one bend per edge. Considering two vertices u and v, following ∂P in counterclockwise order from u to v gives an open interval $P(u, v)$ of ∂P.

For a pair of vertices u, v, denote by $\overline{uv}$ the straight-line segment between u and v and by $\pi(u, v)$ the shortest path between u and v in P.

The faces $f_1, \ldots, f_n$ of G induce a unique dual tree T [13] with edges $e_1^*, \ldots, e_{n-1}^*$, where e_i^* is the dual of the interior edge e_i; see Fig. 1a. Our algorithm will use T, incrementally processing and pruning T and P. Consider T to be rooted at some degree-1 node f_n. Denote by $p(f_i)$ the parent of f_i in T, and by e_i the edge between f_i and its parent. We say that f_i and f_j are *siblings* if $p(f_i) = p(f_j)$. For an edge $e_i = (u, v)$, let $\pi(u, v) \circ P(v, u)$ be the part of P containing the root f_n (where $\circ$ denotes the concatenation). Then for e_i with $P(u, v)$ containing no other vertices of V, the face f_i is a leaf in T.

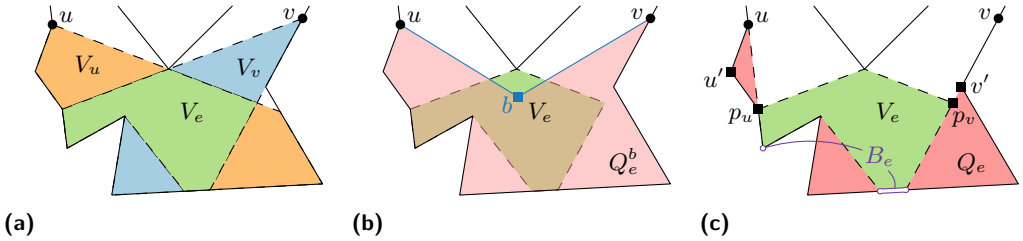


Figure 2 Illustrations for Lemma 2.2: (a) Visibility regions V_u, V_v and intersection V_e; (b) the region cut off by drawing e_i with its bend at point b; (c) construction of p_u, p_v and obstructed region Q_e.

2 Algorithm

An interior edge $e = (u, v)$ divides the polygon into two parts. During the algorithm, we will use these edges to cut some parts of the polygon off. We will make clear in each step which part of the polygon is the remaining one.

We say that any edge $e = (u, v)$ is a *reflex* edge if it has to be drawn with a bend that defines a reflex angle inside the remaining polygon—see Fig. 1c—and a *convex* edge if it can be drawn with a convex bend or as a straight line. Note that an edge $e = (u, v)$ is reflex if and only if $\overline{uv}$ intersects $P(u, v)$ or is completely outside of P.

In the following analysis, we fix some leaf f' of T as the root and will consider the faces of G in some order $f_1, \ldots, f_n$ with $f_n = f'$. Let G_i be the subgraph of G induced by the vertices incident to the faces $f_i, \ldots f_n$, hence $G = G_1$. Symmetrically, define T_i to be the dual tree of G_i with $T = T_1$. In step i of our algorithm, we consider T_i and pick f_i to be a leaf, process the interior edge e_i between f_i and its parent, and refine the polygon to P_{i+1} such that G_{i+1} can be drawn in P_{i+1} if and only if G_i can be drawn in P_i. So, $P_1 = P$ and, in each step, the remaining part P_i of the polygon is the one containing the root f_n (i.e., e_n.)

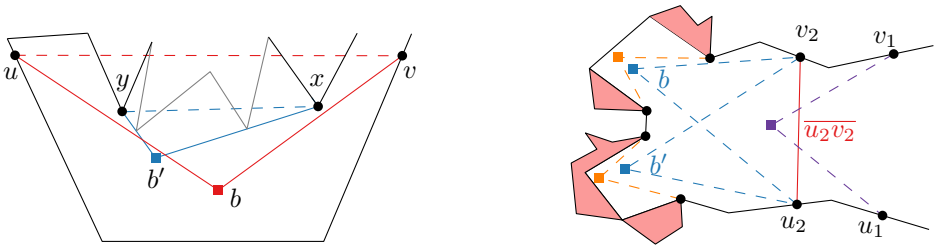
The algorithm chooses the next face f_i to process as follows: If T_i has a leaf corresponding to a reflex edge, we choose that face as f_i. Otherwise, all leaves in T_i correspond to convex edges, and we choose one of the lowest leaves, that is, a leaf with the largest distance (in the graph-theoretic sense) to the root. This way, we can make sure that a convex edge is only chosen if all its siblings that correspond to reflex edges have already been processed.

Let u and v be the end-vertices of the edge e_i separating f_i from its parent in T_i. Let V_u and V_v be the regions of P_i visible from u and v, respectively, and let $V_{e_i} = V_u \cap V_v$ be their intersection. For any point $b \in V_{e_i}$, let $Q^b_{e_i}$ be the subpolygon of P_i bounded by $P_i(u, v) \circ \overline{vb} \circ \overline{bu}$, that is, the part of P_i that is "cut off" by drawing e_i with its bend at b; see Fig. 2b. A point $b \in V_{e_i}$ is called a *minimal bend point* for e_i if there is no other point $b' \in V_{e_i}$ with $Q^{b'}_{e_i} \subsetneq Q^b_{e_i}$. Let B_{e_i} be the set of all minimal bend points for e_i. Further, let (u, u') be the segment of $P_i(u, v)$ incident to u, and let (v', v) be the segment of $P_i(u, v)$ incident to v.

We define $Q_{e_i} = \bigcap_{b \in B_{e_i}} Q^b_{e_i}$ to be the region of P_i *obstructed* by e_i: Wherever we place the bend point of e_i, all the points of Q_{e_i} will be cut off; see Fig. 2c. Conversely, for every point $p \in P_i \setminus Q_{e_i}$, there is a placement of the bend point of e_i such that p is not cut off.

To construct Q_{e_i}, proceed as follows: Rotate a ray around u starting from (u, u') in counterclockwise order until we hit V_{e_i}, call this point p_u. Rotate a ray around v starting from (v', v) in clockwise order until we hit V_{e_i}, call this point p_v. Then Q_{e_i} is the (not necessarily simple) subpolygon of P_i bounded by $\overline{vp_v} \circ V_e(p_u, p_v) \circ \overline{p_u u} \circ P_i(u, v)$; see Fig. 2c.

Similarly, we define $R_{e_i} = \left(\bigcup_{b \in B_{e_i}} Q^b_{e_i}\right) \setminus Q_{e_i}$ to be the region of P_i *restricted* by e_i: For



(a) (b)

Figure 3 (a) The gray part of $P(x, y)$ forces b' to be placed inside $R^b_{(u,v)}$, inducing an intersection of (u, v) and (x, y). For that to be necessary, b' must create a reflex angle. (b) Two edges $e_1 = (u_1, v_1)$, $e_2 = (u_2, v_2)$ with $p(e_2) = e_1$. Fixing the 1-bend drawing of e_1 to intersect $\overline{u_2v_2}$ makes e_2 become a reflex edge, eliminating any choice.

each point $r \in R_{e_i}$, there are two minimal bend points b and b' for e_i such that bending e_i at b cuts off r, whereas bending e_i at b' does not.

▶ **Lemma 2.1.** *Let e be a reflex edge. Then there is a unique minimal bend point b for e.*

Consider how b is constructed in Fig. 1c; note that b is a vertex of V_e. Any point b' above the dashed lines would not be minimal, and any point inside the red region is not visible by at least one of u and v.

▶ **Lemma 2.2.** *Let e be a convex edge. Then $B_e \subset \partial V_e$ and we can safely remove Q_e.*

In step i, our algorithm computes V_{e_i}. If $V_{e_i} = \emptyset$, then it is impossible to draw G_i in P_i, so by induction it is impossible to draw G in P and our algorithm stops. Otherwise, the algorithm computes Q_{e_i} and creates $P_{i+1} = P_i \setminus Q_{e_i}$. If an edge e_j $(j > i)$ had to place its bend point inside Q_{e_i}, then e_i and e_j would cross, independently of the choice of the bend point of e_i; in this case, our algorithm will conclude that it is impossible to draw G in P when it processes e_j. In the following, we will show that G_{i+1} can be drawn in P_{i+1} if and only if G_i can be drawn in P_i.

▶ **Lemma 2.3.** *Let $\mathcal{S}(f)$ be the set of all convex siblings with parent f in T. For any pair of edges $e_1, e_2 \in \mathcal{S}(f)$, the restricted regions R_{e_1} and R_{e_2} are interior-disjoint.*

Sketch of proof. For $f = (u', v')$ consider $P(u', v')$. Since e_1 and e_2 are siblings below f, they are "next to" each other along $P(u', v')$, not nested. Consider edge (u, v) in Fig. 3a: without the gray part of $P(x, y)$ intersecting $\overline{xy}$, the edge (x, y) would be convex. Since the restricted regions $R_{(u,v)}$ and $R_{(x,y)}$ are "hidden" behind the corresponding straight lines, they need to be disjoint. ◀

As a side note: If some parts of P would force two edges to cross, then at least one of these edges has to be reflex; see Fig. 3a with the gray parts in place.

While we can fix the bend points for any reflex edges, convex bends remain undecided until the root is reached, as only some of its minimal bend points might be compatible with the valid bend points of its parent edge. Hence, our algorithm will work bottom-up, fixing reflex edges and computing the obstructed regions for all edges, refining the polygon. When the root f_n is encountered, it is a leaf and the polygon is refined using the obstructed region of its (unique) child. If P_n is not empty, then a solution exists, and we find it by traversing the tree top-down. If e_{n-1} is reflex, then we already fixed its bend point in the bottom-up traversal; otherwise, we place the bend point of e_{n-1} at an arbitrary point inside

the restricted region of e_{n-1} in P_n. Then we obtain the subpolygon P^*_{n-1} that has to contain the drawing of all children of P^*_{n-1} by removing the subpolygon bounded by $P_n(v,u)$ and e_{n-1} from P_{n-1}. When processing face f_i, we again place the bend points of its convex children at arbitrary points inside their restricted region in P^*_i. Since these regions are interior-disjoint by Lemma 2.3, the edges will not intersect. Note that e_i might have been placed in the restricted region of one of its children e_j in P_i; see Fig. 3b. However, since this is the only bend point of the edges incident to f_i that can lie in the restricted region of e_j, by definition there is still a valid bend point for e_j. We again construct the subpolygons P^*_j by removing the subpolygon bounded by $P_i(v,u)$ and e_j from P_j.

This procedure allows us to limit correctness-considerations to consecutive decisions only, and we get the following lemma.

▶ **Lemma 2.4.** *Given the edge e_i in step i, if V_{e_i} is non-empty, then G_{i+1} can be drawn in P_{i+1} if and only if G_i can be drawn in P_i.*

Proof. Whenever we have $V_{e_i} = \emptyset$ for any edge e_i inside P_i, we stop. Further refining P_i will only make it smaller and thus cannot increase the size of any visibility regions.

Otherwise, we either compute the unique best bend point b (Lemma 2.1), or the obstructed region Q_{e_i} (as described above), refining P_i to P_{i+1} accordingly. Lemma 2.2 ensures that the latter is safe.

By construction, no point of the drawing of G_{i+1} can lie in the region Q_{e_i} obstructed by e_i, but the restricted region of e_i can overlap with other restricted regions. Since minimal bend points cannot lie in restricted regions of siblings (Lemma 2.3), only the bend point of the edge corresponding to $p(e_i)$ can possibly be placed in the restricted region R_{e_i} of e_i; any other bend point that lies inside $R(e_i)$ must lie on the opposite side of the drawing of $p(e_i)$, so it cannot influence the choice of the bend point for e_i.

With at most one other bend point in any restricted region, routing the corresponding edge is still possible by definition of R_{e_i}. ◀

We are now ready to state the main result of this paper.

▶ **Theorem 2.5.** *Given an outerplanar graph G with n vertices, a polygon P with m corners, and a mapping of the vertices of G to ∂P, we can decide in $O(mn)$ time whether G can be drawn in P with at most one bend per edge.*

Proof. We use the algorithm described above. The correctness follows immediately from Lemma 2.4. The most time consuming part of the algorithm is to compute the region V_e for each edge $e = (u,v)$, namely the one that is visible from both points u and v. Since V_e is a simple polygon with at most $2m$ edges, it can be computed in $O(m)$ time [5, page 15]. Hence, all these regions can be computed in $O(mn)$ total time. The remaining parts of the algorithm (computing the dual graph of G, choosing the order of the faces f_i in which we traverse the graph, computing Q_e, "cutting off" parts of P, and propagating the graph at the end to fix the presentation) can clearly be done within this time. Thus, the total running time is $O(mn)$. ◀

Acknowledgments. This work was initiated at the Workshop on Graph and Network Visualization 2019. We thank all the participants for helpful discussions and Anna Lubiw for bringing the problem to our attention.

References

1 Patrizio Angelini, Giuseppe Di Battista, Fabrizio Frati, Vít Jelínek, Jan Kratochvíl, Maurizio Patrignani, and Ignaz Rutter. Testing planarity of partially embedded graphs. *ACM Trans. Algorithms*, 11(4):32:1–32:42, 2015. doi:10.1145/2629341.

2 Franz-Josef Brandenburg, David Eppstein, Michael T. Goodrich, Stephen G. Kobourov, Giuseppe Liotta, and Petra Mutzel. Selected open problems in graph drawing. In Giuseppe Liotta, editor, *Proc. 11th Int. Symp. Graph Drawing (GD)*, volume 2912 of *Lecture Notes Comput. Sci.*, pages 515–539. Springer, 2003. doi:10.1007/978-3-540-24595-7_55.

3 Timothy M. Chan, Fabrizio Frati, Carsten Gutwenger, Anna Lubiw, Petra Mutzel, and Marcus Schaefer. Drawing partially embedded and simultaneously planar graphs. *J. Graph Algorithms Appl.*, 19(2):681–706, 2015. doi:10.7155/jgaa.00375.

4 Hubert de Fraysseix, János Pach, and Richard Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10(1):41–51, 1990. doi:10.1007/BF02122694.

5 Alexander Gilbers. *Visibility Domains and Complexity.* PhD thesis, Rheinische Friedrich-Wilhelms-Universität Bonn, 2014.

6 Seok-Hee Hong and Hiroshi Nagamochi. Convex drawings of graphs with non-convex boundary constraints. *Discrete Appl. Math.*, 156(12):2368–2380, 2008. doi:10.1016/j.dam.2007.10.012.

7 Vít Jelínek, Jan Kratochvíl, and Ignaz Rutter. A Kuratowski-type theorem for planarity of partially embedded graphs. *Comput. Geom.*, 46(4):466–492, 2013. doi:10.1016/j.comgeo.2012.07.005.

8 Anna Lubiw, Tillmann Miltzow, and Debajyoti Mondal. The complexity of drawing a graph in a polygonal region. In Therese C. Biedl and Andreas Kerren, editors, *Proc. 26th Int. Symp. Graph Drawing Netw. Vis.*, volume 11282 of *Lecture Notes Comput. Sci.*, pages 387–401. Springer, 2018. doi:10.1007/978-3-030-04414-5_28.

9 Tamara Mchedlidze, Martin Nöllenburg, and Ignaz Rutter. Drawing planar graphs with a prescribed inner face. In Stephen K. Wismath and Alexander Wolff, editors, *Proc. 21st Int. Symp. Graph Drawing*, volume 8242 of *Lecture Notes Comput. Sci.*, pages 316–327. Springer, 2013. doi:10.1007/978-3-319-03841-4_28.

10 Tamara Mchedlidze and Jérôme Urhausen. β-stars or on extending a drawing of a connected subgraph. In Therese C. Biedl and Andreas Kerren, editors, *Proc. 26th Int. Symp. Graph Drawing Netw. Vis.*, volume 11282 of *Lecture Notes Comput. Sci.*, pages 416–429. Springer, 2018. doi:10.1007/978-3-030-04414-5_30.

11 János Pach and Rephael Wenger. Embedding planar graphs at fixed vertex locations. *Graphs Comb.*, 17(4):717–728, 2001. doi:10.1007/PL00007258.

12 Maurizio Patrignani. On extending a partial straight-line drawing. *Int. J. Found. Comput. Sci.*, 17(5):1061–1070, 2006. doi:10.1142/S0129054106004261.

13 Andrzej Proskurowski and Maciej Syslo. Efficient Vertex- and Edge-Coloring of Outerplanar Graphs. *SIAM J. Alg. Disc. Meth.*, 7:131–136, 01 1986. doi:10.1137/0607016.

14 Walter Schnyder. Embedding planar graphs on the grid. In David S. Johnson, editor, *Proceedings 1st Ann. ACM-SIAM Symp. Discrete Alg. (SODA)*, pages 138–148. SIAM, 1990. URL: http://dl.acm.org/citation.cfm?id=320176.320191.

15 William Thomas Tutte. How to draw a graph. *Proc. London Math. Soc.*, 3(1):743–767, 1963.

Labeling Nonograms

Maarten Löffler[1] and Martin Nöllenburg[2]

1 **Department of Computing and Information Sciences, Utrecht University, the Netherlands**
`m.loffler@uu.nl`

2 **Algorithms and Complexity Group, TU Wien, Vienna, Austria**
`noellenburg@ac.tuwien.ac.at`

Abstract

Slanted and curved nonograms are a new type of picture puzzles introduced by van de Kerkhof *et al.* (2019). They consist of an arrangement of lines or curves within a frame B, where some of the cells need to be colored in order to obtain the solution picture. Up to two clues are attached as numeric labels to each line on either side of B. In this paper we study the algorithmic problem of optimizing or deciding the existence of a placement of the given clue labels to a nonogram. We provide polynomial-time algorithms for restricted cases and prove NP-completeness in general.

1 Introduction

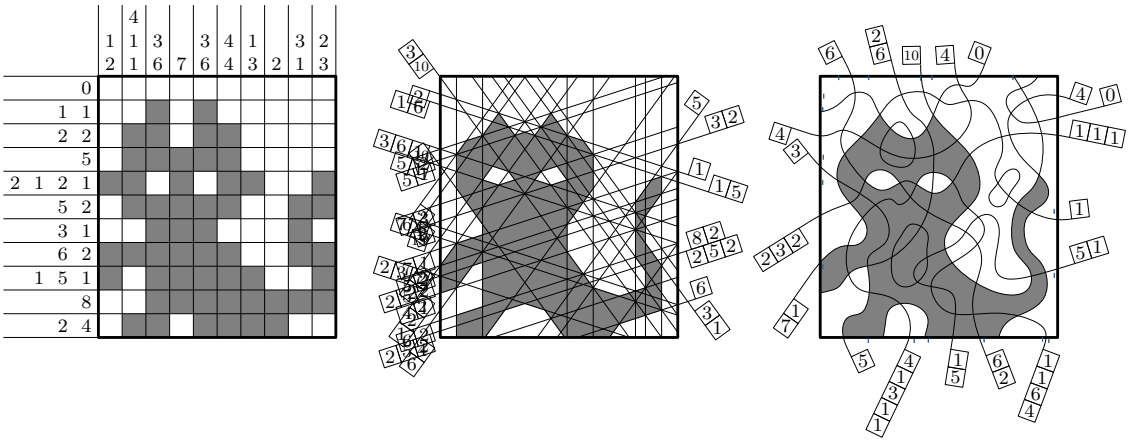


Figure 1 (left) A classic nonogram in solved state. (middle) A slanted nonogram. On the left, we show all possible labels, creating a large amount of overlap. On the right, some possible ways to resolve overlapping labels: extend labels from the same port, extend parallel labels, draw both labels of the same line at the same side, or allow crossings outside the puzzle frame. (right) A curved nonogram, showing a subset of labels (some extended to avoid overlap) which still results in a unique puzzle.

Nonograms, also known as *Japanese puzzles*, *paint-by-numbers*, or *griddlers*, are a popular puzzle type where one is given an empty grid and a set of clues on which grid cells need to be colored, typically resulting in a picture (see Figure 1 (left)). The difficulty of solving nonograms has been studied [2, 6], and remains an active topic of discussion [7].

Van de Kerkhof *et al.* introduced *curved* nonograms, a variant in which the puzzle is no longer played on a grid but on any arrangement of lines or even curves [11]. See also [8, 10, 12]. Of special interested are *slanted* nonograms [10], in which all curves are straight lines, possibly limited to a fixed number k of orientations (also known as *sloped* nonograms or, in the case of $k = 4$, *tangram* nonograms). Figure 1 illustrates the different variants.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Van de Kerkhof *et al.* describe heuristics to generate puzzles, but they leave open the question of how to attach labels with clues. This is a non-trivial task, as labels could be placed in several valid locations. Each curve enters and leaves the picture *frame* (bold black rectangle in the figures) once, and the information about which incident cells of the arrangement should be filled is summarized in two *clues*, one on each side of the curve (we refer the reader to [11] for a full description of the rules). This gives two logical potential locations for each clue. Furthermore, it may be possible to extend curves outside the frame to make room for the clues, and not all clues need to be given for the puzzle to be solvable.

How to best label curved nonograms is an interesting open problem; it is most apparent in the case of slanted nonograms, since the rigid structure limits the possible label locations.

Problem statement. In the *nonogram labeling problem* we are given the following input:

(i) a *nonogram frame*, which is a simple convex polygon B;
(ii) a set $\mathcal{L}$ of *nonogram lines* passing through B, each $l \in \mathcal{L}$ defining a pair (p_l, q_l) of *ports* at the intersection points of l with B; and
(iii) a pair of non-negative integers (a_l, b_l) for each $l \in \mathcal{L}$, where a_l defines the width of the label above l and b_l defines the width of the label below l.

As output, we ask for a *labeling* of $\mathcal{L}$, such that for each label ℓ of nonogram line $l \in \mathcal{L}$:

(i) ℓ is assigned to one of the two ports p_l or q_l;
(ii) ℓ is assigned an *extension length* d (which could be 0);
(iii) we draw a *leader* of length d from its assigned port p_l or q_l aligned with the slope of l, and the label itself as an $a_l \times 1$ (or $b_l \times 1$) rectangle, also aligned with the slope of l, and anchored at the end of its leader.

A labeling is *valid* if no two labels overlap each other, no label overlaps an extension leader of another label, and no label intersects the frame. In addition to being valid, we identify three further desired properties.

- **Crossing-free.** We disallow intersections between leaders.
- **Balanced.** We require the two labels of a line $l \in \mathcal{L}$ to be assigned to opposite ports.
- **Compact.** We require all leaders to be of length 0, or the shortest length necessary to avoid an intersection between the label and the frame.

We would like to find a crossing-free compact balanced labeling, but this may not exist (see, e.g., Figure 1 (middle)). We study the computational problem of testing the existence of a solution, or, for some variants, minimizing the total leader length, for several combinations of properties. In some cases, we also restrict the number k of distinct slopes of lines in $\mathcal{L}$.

Results. First, we observe that a balanced solution which is not crossing-free and not compact always exists: we simply extend the leaders sufficiently far. Since this does not give a satisfactory result, we focus on more restricted variants in the remainder.

In Section 2.1, we show that testing whether a compact solution exists is possible in polynomial time. In Section 2.2, we show that a non-crossing balanced solution of minimal total leader length can be computed in polynomial time, if the assignment of labels to ports is given and $k = 2$. Finally, in Section 3, we show that the problem of testing whether a crossing-free solution exists is NP-complete, even when $k = 2$.

Related work. Labeling nonograms is closely related to the boundary labeling problem in information visualization, where a set of point features in a rectangular frame B is to be annotated with labels (names or short descriptions) that are placed outside B and connected to their features with straight or polygonal leaders [4, 5]. Yet nonogram labeling is different in several respects: Since the curves/lines in nonograms intersect the frame B in two fixed locations, the possible positions for the clues are very restricted, while in boundary labeling the label can basically be placed anywhere along B as long as the resulting leader lines are valid. Labels in boundary labeling are typically axis-aligned, but the clues in nonograms are aligned with their respective nonogram line or curve. Finally, by extending the nonogram curves beyond the frame to gain extra space, we obtain a new degree of freedom that has been rarely used in boundary labeling, with some exceptions of multi-row labeling [3, 9].

2 Algorithms

2.1 Compact labeling

In our first result, we assume that each label ℓ must be placed as close to the frame B as possible, i.e., ℓ must touch B. This leaves only one degree of freedom for each label ℓ of a nonogram line l, namely whether it is placed at port p_l or q_l.

▶ **Theorem 2.1.** *Given a nonogram labeling instance, we can decide in polynomial time whether a compact labeling exists. This is true regardless of whether we require it to be balanced.*

Proof. We derive a 2-SAT formula φ that has a satisfying variable assignment if and only if a valid labeling without leader extensions exists. For each nonogram line $l \in \mathcal{L}$ we define two variables x_l^a and x_l^b, where $x_l^a = 1$ ($x_l^a = 0$) indicates that the label above l is assigned to the port p_l (q_l) of l. Similarly, $x_l^b = 1$ ($x_l^b = 0$) indicates that the label below l is assigned to p_l (q_l). It is clear that a variable assignment is in bijection to a port assignment of the labels and it remains to add some clauses to φ to model the valid labelings. For each overlap of a label of a line l with a label of another line l' (we call that a *conflict*), we add a clause that prevents both labels to be selected simultaneously. As an example consider the case that the label above l and the label below l' intersect if both assigned to their ports p_l and $p_{l'}$. Then we add the clause $\neg x_l^a \vee \neg x_{l'}^b$. Now a satisfying assignment for φ corresponds to an assignment of each label to a port of its nonogram line such that no two labels intersect each other; otherwise some clause would not be satisfied. To ensure that the labeling is balanced, we would add the additional clauses $x_l^a \vee x_l^b$ and $\neg x_l^a \vee \neg x_l^b$ for each $l \in \mathcal{L}$.

Solving the 2-SAT instance takes linear time [1] in the size of φ, where the number of clauses of φ is linear in the number of nonogram lines and the number of label conflicts. ◀

We remark that this 2-SAT model is independent of the type of nonogram lines (or curves) and the shape of B. It depends only on the set of conflicting candidate label positions.

2.2 Fixed side assignment

Our second algorithm allows extensible leaders, but disallows leader intersections and assumes that a balanced assignment of the labels to the ports of each nonogram line is given. We further assume that the nonogram lines have slopes ± 1 and that the frame B is a rectangle.

▶ **Lemma 2.2.** *For a nonogram labeling instance with n lines and a balanced fixed side assignment for each label we can discretize the relevant extension lengths of each label to $O(n^2)$ values such that we can find a labeling of minimum total extension length among this set of extension lengths (if the instance has a solution at all).*

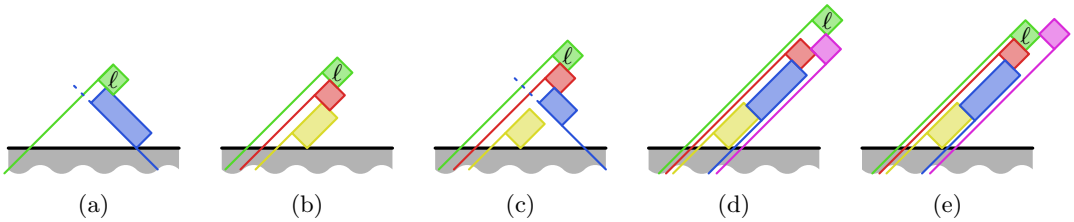


Figure 2 Possible cases of extension lengths for label ℓ in the proof of Lemma 2.2.

Proof. In a minimum-length labeling, every label ℓ of a nonogram line l should be shifted as close to B as possible without intersecting another label. Hence it either touches B and the extension length is 0 or 1 (depending on which side of l is labeled), or it touches another label ℓ' blocking it from moving closer to B. This blocking label ℓ' can belong to a line of different slope, meaning that the extension length of ℓ is given by the intersection point of the two nonogram lines (possibly $+1$), see Figure 2(a). There are $O(n)$ such intersection points for ℓ. Or, the lines of ℓ and ℓ' are parallel and of distance at most 2. If the chain of blocking relations comprises only parallel lines and all of them have the labels on the same side (Figure 2(b)), then we get a single extension length for ℓ. If the parallel lines come in a group of right-flipped labels followed by a group of left-flipped labels as in Figures 2(d–e) then any prefix of the sequence of left-flipped labels can add to the extension length of ℓ, which again yields $O(n)$ possible extension lengths. Finally, ℓ may be blocked by some chain of parallel labels, the last of which is blocked by a label of an orthogonal line (Figure 2(c)). Considering all combinations this last case can give rise to $O(n^2)$ different extension lengths. ◀

▶ **Theorem 2.3.** *Given a nonogram labeling instance with n lines and a fixed side assignment for each label, we can decide in $O(n^9)$ time, whether an assignment of an extension length to each label exists such that the resulting labeling is valid. If this is the case we can find one of minimum total extension length.*

Proof. (Sketch) The idea of the algorithm is to use dynamic programming. Consider an edge e of B and all the lines crossing e. From Lemma 2.2 we know that it is sufficient to consider at most $O(n^2)$ many extension lengths for each label. We define a subinstance of the labeling problem for edge e by selecting two boundary lines l_1 and l_2 together with an extension length for each of the two labels. This defines $O(n^6)$ possible subinstances. Any line with a port between those of l_1 and l_2 is restricted to stay in the region bounded by l_1, l_2, and a horizontal line through the topmost point of the shorter of the two lines l_1, l_2, see Figure 3. To solve such an instance recursively, we optimize over all lines $\hat{l}$ contained in the instance and all admissible and intersection-free extension lengths for that label and recurse into the two subinstances defined by l_1 and $\hat{l}$ as well as $\hat{l}$ and l_2 (see the two shaded subinstances defined by l_4 between l_3 and l_2 in Figure 3). The optimization step takes $O(n^3)$ time for each subinstance. We initialize the recursion with two outward pointing dummy lines and repeat the process for all sides of B. This yields an overall $O(n^9)$ running time. ◀

3 Hardness

The problem of testing whether a valid labeling exists, in the setting where we disallow crossing leaders, but are allowed to choose at which port each label is placed and are allowed to extend the leaders to any desired length, is NP-hard. We will construct an instance with a rectangular frame which only has lines of slopes 1 and -1. We will reduce from 3-SAT.

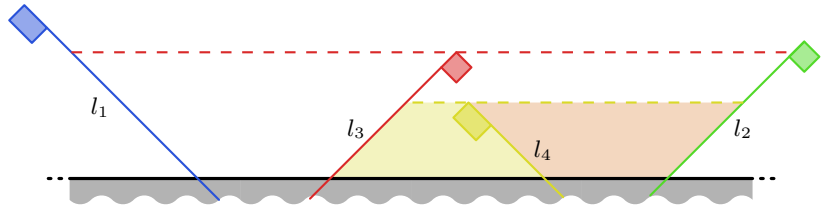


Figure 3 Illustration of the dynamic programming recursion.

Variables. The bulk of the construction consists of *cross gadgets*. A cross gadget consists of two short labels intersecting each other at 90° angles.[1] Figure 4 shows a single cross gadget.

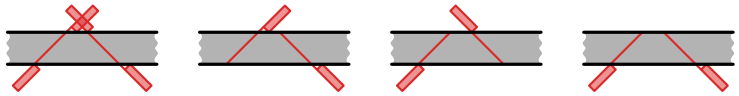


Figure 4 A variable gadget (cross gadget) and its three possible valid solutions.

Note that for a cross gadget it is not relevant on which side of the lines the labels are, and that although labels can be extended, doing so does not change the combinatorial choices of which combinations of sides are possible.

A cross gadget has three possible valid states, and we wish to use them to represent variables, which have two valid states. Furthermore, we would like to enforce multiple cross gadgets to represent the same variable. We can achieve both of these properties by connecting several cross gadgets into a *variable loop*. Figure 5 illustrates a variable loop.

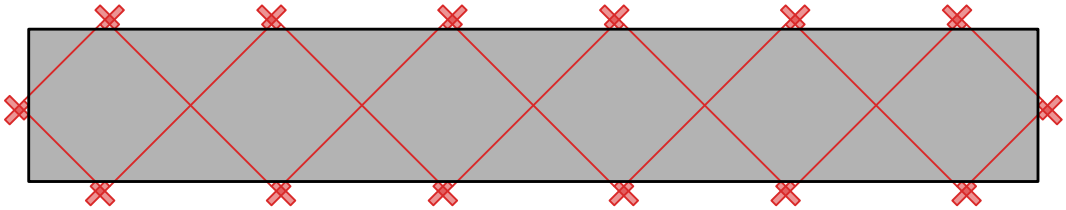


Figure 5 A single variable loop.

By connecting the cross gadgets into a loop, we ensure that only two valid solutions remain for each cross gadget. Note that we can increase the number of occurrences of a cross of the same variable by making the frame wider, and we can increase the distance between consecutive crosses by making the frame higher. We can embed multiple independent loops next to each other, as illustrated in Figure 6.

In the construction, we will also need to place some labels which cannot be removed. We create a special variable loop, for which one of the states is disabled by a crossing between a positive and a negative label. We achieve this by making them longer (see Figure 7). Note that forced (black) labels are only forced to be on a particular *side* of B; they can still be extended, but we will use them in a way where extending black labels is never useful.

[1] Here, we are only using one side of each line $l \in \mathcal{L}$, which corresponds to a setting where not all clues are present in the puzzle. The contruction can easily be adapted to the case where both labels are present, by placing them at the same side (i.e. not balanced).

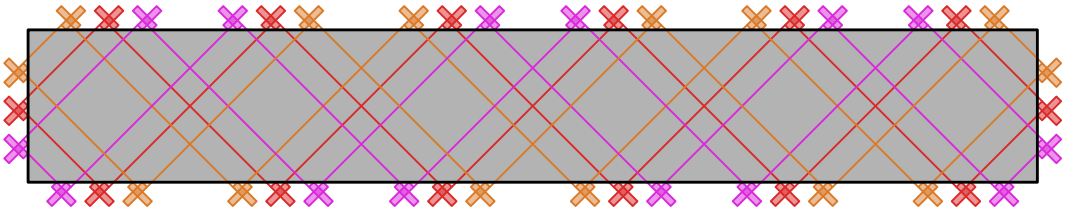


Figure 6 Multiple variable loops.

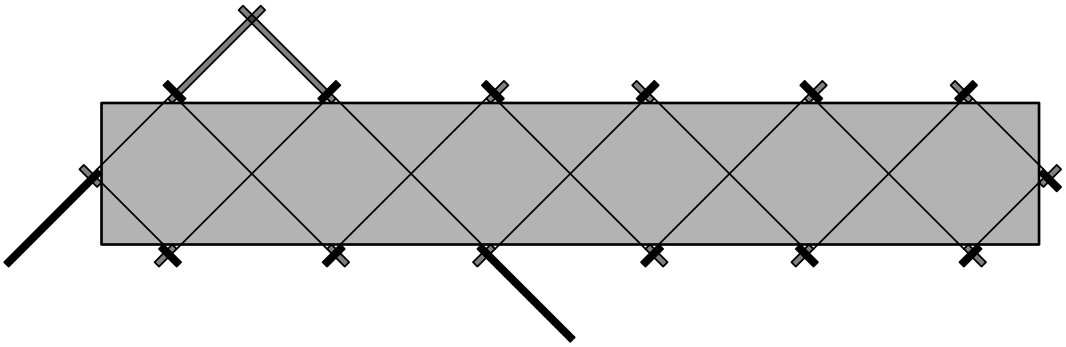


Figure 7 A variable loop where one state is impossible; solid black labels are *forced*.

Clauses. Next, a clause gadget essentially consists of a single *clause* label which is forced to be at a specific port, but can be extended. Depending on how far it is extended, it will intersect different variable labels. The clause label is restricted to only three essentially different positions by two fixed labels. Figure 8 illustrates a clause gadget.

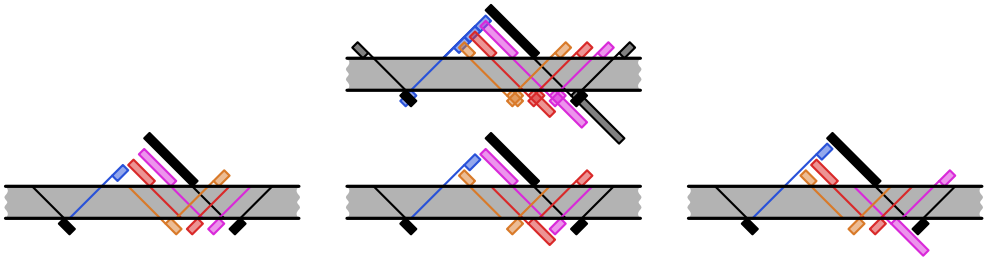


Figure 8 Clause gadget and three possible valid solutions.

For a clause label, it is important on which side of the line the label is placed: it must be faced towards the variable labels. We need to connect the clause gadget to the correct literals of the three variables involved in the clause, as well as to the fixed loop on two sides. For this, we need to make some very long labels. Figure 9 illustrates how a clause gadget is connected, showing only the relevant labels. Note that the variables may need to be connected to two different groups of cross gadgets in the variable loops.

The global picture. Globally, we embed the different clauses horizontally next to each other. Each clause, including its connections, covers a horizontal distance of a constant number of variable zig-zags. These connections are placed between the variable loops, so they do not interfere. Figure 10 shows how the first clause and the beginning of the second clause could look globally, without hiding any labels.

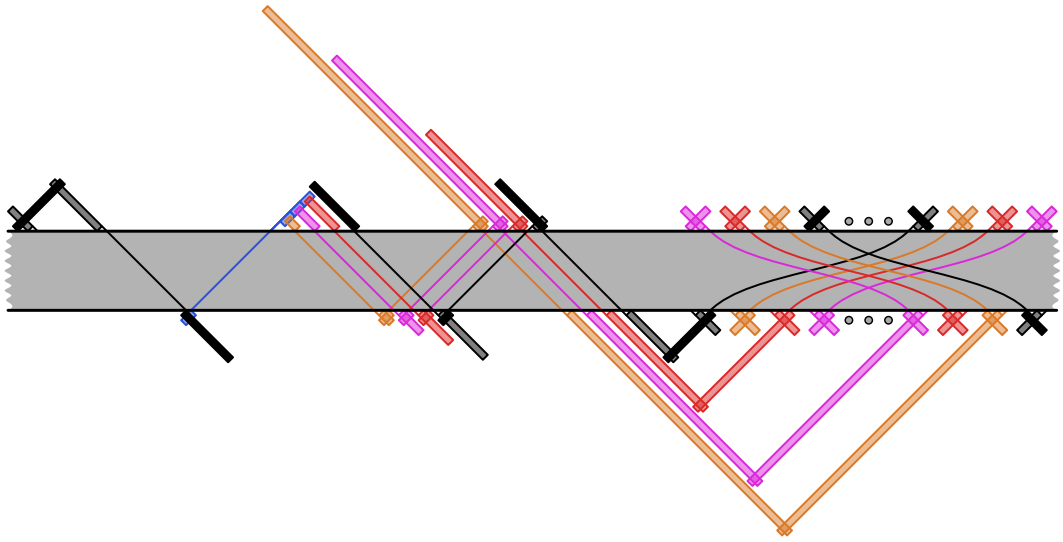


Figure 9 Connecting a clause gadget to the correct literals for clause ¬`purple` ∨ `red` ∨ ¬`orange`.

The total horizontal distance covered will be $O(nm)$. The vertical distance is $O(n)$.

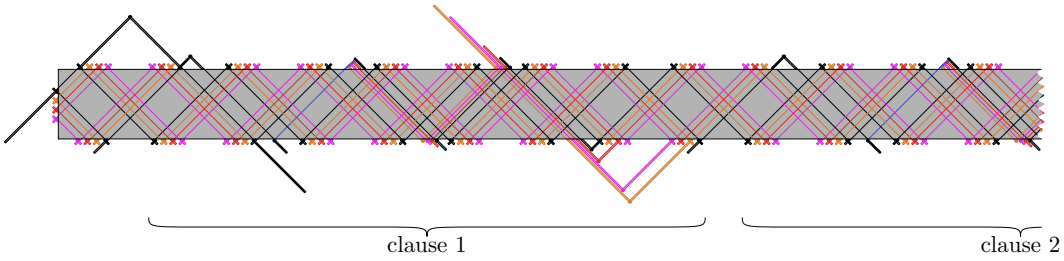


Figure 10 The global picture.

▶ **Theorem 3.1.** *Given a nonogram instance without side assignment and extensible leaders, it is* NP*-complete to decide whether a valid labeling exists.*

4 Future work

Several interesting questions in nonogram labeling remain open. Our hardness reduction uses long labels whose lengths depend on the size of the 3-SAT instance. In contrast, most labels in real-world nonograms are $1 \times c$ rectangles for small constant values of c. This raises the question of investigating the computational complexity of nonogram labeling for bounded label lengths. A second question follows from Theorem 2.1. If a compact balanced labeling does not exist, but a non-balanced one does, then a natural optimization problem is to maximize the number of balanced pairs of labels.

References

1 Bengt Aspvall, Michael F. Plass, and Robert Endre Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, 8(3):121–123, 1979. doi:10.1016/0020-0190(79)90002-4.

2 Kees Joost Batenburg and Walter A. Kosters. On the difficulty of nonograms. *ICGA Journal*, 35(4):195–205, 2012. doi:10.3233/ICG-2012-35402.

3 Michael A. Bekos, Michael Kaufmann, Katerina Potika, and Antonios Symvonis. Multi-stack boundary labeling problems. In S. Arun-Kumar and Naveen Garg, editors, *Foundations of Software Technology and Theoretical Computer Science (FSTTCS'06)*, volume 4337 of *LNCS*, pages 81–92. Springer, 2006. doi:10.1007/11944836_10.

4 Michael A. Bekos, Michael Kaufmann, Antonios Symvonis, and Alexander Wolff. Boundary labeling: Models and efficient algorithms for rectangular maps. *Computational Geometry Theory and Applications*, 36(3):215–236, 2007. doi:10.1016/j.comgeo.2006.05.003.

5 Michael A. Bekos, Benjamin Niedermann, and Martin Nöllenburg. External labeling techniques: A taxonomy and survey. *Computer Graphics Forum*, 38(3):833–860, 2019. doi:10.1111/cgf.13729.

6 Daniel Berend, Dolev Pomeranz, Ronen Rabani, and Ben Raziel. Nonograms: Combinatorial questions and algorithms. *Discrete Applied Mathematics*, 169:30–42, 2014. doi:10.1016/j.dam.2014.01.004.

7 Yen-Chi Chen and Shun-Shii Lin. A fast nonogram solver that won the TAAI 2017 and ICGA 2018 tournaments. *ICGA Journal*, 41(1):2–14, 2019. doi:10.3233/ICG-190097.

8 Tim K. de Jong. The concept and automatic generation of the curved nonogram puzzle. Master's thesis, Utrecht University, 2016. URL: https://dspace.library.uu.nl/handle/1874/337632.

9 Andreas Gemsa, Jan-Henrik Haunert, and Martin Nöllenburg. Multi-row boundary-labeling algorithms for panorama images. *ACM Trans. Spatial Algorithms and Systems*, 1(1):1:1–1:30, 2015. doi:10.1145/2794299.

10 Raphael J. Parment. Generation of sloped nonograms. Master's thesis, Utrecht University, 2015. URL: https://dspace.library.uu.nl/handle/1874/323196.

11 Mees van de Kerkhof, Tim de Jong, Raphael Parment, Maarten Löffler, Amir Vaxman, and Marc J. van Kreveld. Design and automated generation of japanese picture puzzles. *Comput. Graph. Forum*, 38(2):343–353, 2019. doi:10.1111/cgf.13642.

12 Mees A. van de Kerkhof. Improved automatic generation of curved nonograms. Master's thesis, Utrecht University, 2017. URL: https://dspace.library.uu.nl/handle/1874/357864.

Certified Approximation Algorithms for the Fermat Point

Kolja Junginger[1], Ioannis Mantas[1], Evanthia Papadopoulou[1], Martin Suderland[1], and Chee Yap[2]

1 **Faculty of Informatics, Università della Svizzera italiana, Lugano, Switzerland**
kolja.junginger@usi.ch,ioannis.mantas@usi.ch,
evanthia.papadopoulou@usi.ch,martin.suderland@usi.ch

2 **Courant Institute, NYU, New York, NY, USA**
yap@cs.nyu.edu

Abstract

Given a set of k points $A \subseteq \mathbb{R}^d$ together with a positive weight function $w : A \to \mathbb{R}_{>0}$, the *Fermat distance function* is $\varphi(\boldsymbol{x}) = \sum_{\boldsymbol{a}\in A} w(\boldsymbol{a})\|\boldsymbol{x} - \boldsymbol{a}\|$. A classic problem in facility location is finding the *Fermat point* $\boldsymbol{x}^*$, the point that minimizes the function φ. We present algorithms to compute an ε-approximation of the Fermat point $\boldsymbol{x}^*$, that is, a point $\widetilde{\boldsymbol{x}}^*$ satisfying $\|\widetilde{\boldsymbol{x}}^* - \boldsymbol{x}^*\| < \varepsilon$. Our algorithms are based on the *subdivision paradigm*, which we combine with Newton methods, used for speed-ups and certification. Our algorithms are certified in the sense of interval methods.

1 Introduction

A classic problem in Facility Location [14, 29] is the placement of a facility to serve a given set of demand points or customers so that the total transportation costs are minimized. The total cost at any point is interpreted as the sum of the distances to the demand points. The point that minimizes this sum is called the *Fermat Point*, see Fig. 1.

A *weighted foci set* is a non-empty finite set of (demand) points $A = \{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_k\} \subseteq \mathbb{R}^d$ associated with a positive weight function $w : A \to \mathbb{R}_{>0}$. Each $\boldsymbol{a} \in A$ is called a *focus* with weight $w(\boldsymbol{a})$. Let $W = \sum_{\boldsymbol{a}\in A} w(\boldsymbol{a})$. The *Fermat distance function* of A is given by

$$\varphi(\boldsymbol{x}) := \sum_{\boldsymbol{a}\in A} w(\boldsymbol{a})\|\boldsymbol{x} - \boldsymbol{a}\|,$$

where $\|\boldsymbol{x}\|$ is the Euclidean norm in $\mathbb{R}^d$. The global minimum value of φ is called the *Fermat radius* of A and denoted r^*; any point $\boldsymbol{x} \in \mathbb{R}^d$ that achieves this minimum, $\varphi(\boldsymbol{x}) = r^*$, is called a *Fermat point* and denoted $\boldsymbol{x}^* = \boldsymbol{x}^*(A)$. If A is not collinear then $\boldsymbol{x}^*$ is unique [22, 24]. We also consider the closely related problem of computing *k-ellipses* of A: for any $r > r^*(A)$, the k-ellipsoid of A of radius r is the level set $\varphi^{-1}(r) := \{\boldsymbol{x} \in \mathbb{R}^d : \varphi(\boldsymbol{x}) = r\}$. When $d = 2$, they are called k-ellipses motivated by classical ellipses being 2-ellipses. Figure 1 shows some 28-ellipses with different radii computed by our algorithm.

The question of approximating the Fermat point is of great interest as its coordinates are the solution of a polynomial with exponentially high degree [3, 28]. An ε-approximation $\widetilde{\boldsymbol{x}}^*$ to the Fermat point $\boldsymbol{x}^*$ can be interpreted in 3 senses: (A) $\|\widetilde{\boldsymbol{x}}^* - \boldsymbol{x}^*\| \le \varepsilon$, (B) $\varphi(\widetilde{\boldsymbol{x}}^*) \le \varphi(\boldsymbol{x}^*)+\varepsilon$, and (C) $\varphi(\widetilde{\boldsymbol{x}}^*) \le (1+\varepsilon)\varphi(\boldsymbol{x}^*)$. In this paper, we consider approximations in the sense (A) which are stronger than senses (B) and (C). To the best of our knowledge, only senses (B) and (C) have been studied so far; they are actually approximations of the Fermat radius.

There is a plethora of results for the Fermat point including, among others, approximations algorithms [2, 6, 9, 11, 15, 18, 30, 38] and special configurations and other variants [1, 5, 8, 10, 13, 14]. A smaller, but equally old, literature also exists for the k-ellipses [25, 28, 33, 35].

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

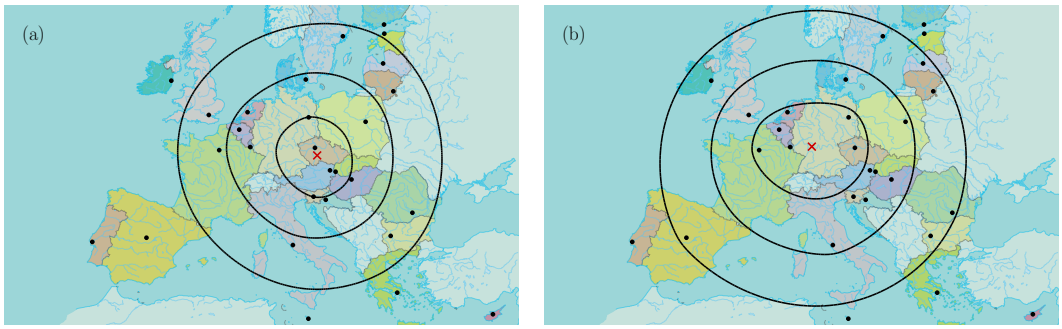


Figure 1 A set of 28 foci, corresponding to EU capitals, with the Fermat point (x) and three 28-ellipses of different radius. (a) Unweighted. (b) The weight of a focus is the population of that country.

In this work we introduce certified algorithms for approximating the Fermat point, combining a subdivision approach with interval methods, cf. [21, 31]. The approach can be formalized in the framework of "soft predicates" [36]. Our certified algorithms are fairly easy to implement, and are shown to have good performance experimentally.

Our Contributions can be summarized as follows: **(1)** We introduce a *box subdivision* scheme to compute an ε-approximation of the Fermat point. **(2)** We augment the *Weiszfeld point sequence* [37] with a Newton-operator test, that outputs an ε-approximation of the Fermat point. **(3)** We implement and experimentally evaluate our algorithms with various datasets. In the full paper, we also address the problem of computing k-ellipses.

2 Preliminaries

Let $\square\mathbb{R}^d$ (or simply $\square\mathbb{R}$ when $d = 1$) denote the set of closed d-dimensional boxes (i.e. Cartesian products of intervals) in $\mathbb{R}^d$. Our subdivision algorithms start with an initial box $B_0 \in \square\mathbb{R}^d$ and recursively split it. We organize the boxes in a *hyper-octree* data structure [32]. A box is specified by an interval in each dimension $B = I_1 \times ... \times I_d$. We denote: m_B the *center* of B, r_B the *radius* of B (distance between m_B and a corner), $\omega(B)$ the *width* of B (the maximum length of its defining intervals), and $c \cdot B$ the box with center m_B and the length of each I_i scaled by c. Operation SPLIT takes each interval of B and splits it in the middle. It returns 2^d *children* of B. Operation SPLIT_2 applies SPLIT to all of B's children and returns the 2^{2d} *grandchildren* of B. We maintain the subdivision *smooth*, i.e., the depth of any two boxes that have overlapping faces differs by a depth of at most 1. Maintaining smoothness does not increase the asymptotic costs [4].

Let P be a logical *predicate* on boxes, i.e. $P : \square\mathbb{R}^d \to \{\texttt{true}, \texttt{false}\}$. A test T looks like a predicate: $T : \square\mathbb{R}^d \to \{\texttt{success}, \texttt{failure}\}$ and it is always associated to some predicate P: Call T a *test for predicate* P if $T(B) = \texttt{success}$ implies $P(B) = \texttt{true}$. However, we conclude nothing if $T(B) = \texttt{failure}$.

▶ **Definition 2.1.** Let T be a test for a predicate P. We call T a *soft predicate* (or *soft version* of P) if it is convergent in this sense: if $(B_n)_{n\in\mathbb{N}}$ is a monotone sequence of boxes $B_{n+1} \subseteq B_n$ that converges to a point $\boldsymbol{a}$, then $P(\boldsymbol{a}) = T(B_n)$ for n large enough.

$\square P(B)$ denotes a soft version of $P(B)$. We construct soft predicates using functions of the form $F\colon \square\mathbb{R}^d \to \square(\mathbb{R} \cup \{-\infty, \infty\})$ approximating a scalar function $f\colon D \to \mathbb{R}$ with $D \subset \mathbb{R}^d$.

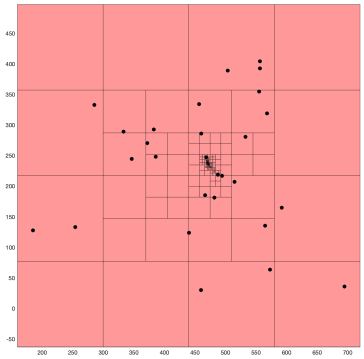
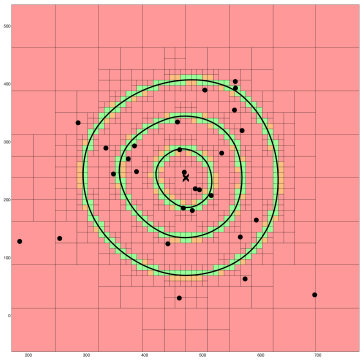
(a)  (b) 

Figure 2 The resulting box subdivision for (a) the Fermat point and (b) the k-ellipses of Fig. 1a.

▶ **Definition 2.2.** Call F a *soft version* of f if it is

(i) *conservative*, i.e. for all $B \in \square\mathbb{R}^d$, $F(B)$ contains $f(B) := \{f(p) : p \in B \cap D\}$, and
(ii) *convergent*, i.e. if for monotone sequence $(B_n)_{n\in\mathbb{N}}$ that converges to a point $\boldsymbol{a} \in D$, $\lim_{n\to\infty} \omega(F(B_n)) = 0$ holds.

We denote F by $\square f$ when F is a soft version of f. There are many ways to achieve $\square f$. E.g. if f has an arithmetic expression E, we can simply evaluate E using interval arithmetic.

3 Approximate Fermat points

We assume that the Fermat point is unique and unequal to a focus. These assumptions are mild and easy to ensure. A focus $\boldsymbol{a} \in A$ is the Fermat point of A if and only if $\left\|\nabla\varphi_{A\setminus\boldsymbol{a}}(\boldsymbol{a})\right\| \leq w(\boldsymbol{a})$ [37]. So, a simple $O(k^2d)$ preprocessing time suffices to verify this assumption. In both subdivision algorithms presented below the time can be reduced to $O(kd)$. The Fermat point problem reduces to computing the critical point of the gradient of φ. We now present three approximation algorithms for the Fermat point $\boldsymbol{x}^*$.

3.1 Using the Subdivision Paradigm

This paradigm requires an initial box B_0. If B_0 is not given, it is easy to find a box containing $\boldsymbol{x}^*$, as $\boldsymbol{x}^*$ lies in the convex hull of A [20]. Function INITIAL-BOX(A), in $O(k)$ time, returns an axis-aligned box with the corners of minimum and maximum x, y coordinates.

▶ **Definition 3.1.** Given a box B, the *gradient exclusion predicate* $C^{\nabla}(B)$ returns true if and only if $\mathbf{0} \notin \nabla\varphi(B)$. If we replace $\nabla\varphi(B)$ by its interval form $\square\nabla\varphi(B)$ see Section 4, we obtain the corresponding interval predicate "$\square C^{\nabla}(B)$".

In our algorithms we keep on splitting boxes using different kinds of predicates while we exclude boxes (red in Fig. 3) that are guaranteed not to contain $\boldsymbol{x}^*$ and we keep boxes (green in Fig. 3) that might contain $\boldsymbol{x}^*$. We test whether we can already approximate $\boldsymbol{x}^*$ well enough by putting a bounding box around all (green) boxes, which we have not excluded yet.

▶ **Definition 3.2.** Given a set of boxes Q, one of which contains the Fermat point, the *stopping predicate* $C^{\varepsilon}(Q)$ returns true, if and only if the minimum axis-aligned bounding box containing all boxes in Q has a radius at most ε.

If C^{ε} returns true, then we can stop. Since the radius of the minimum bounding box is at most ε, the center of the box is an ε-approximation of the Fermat point.

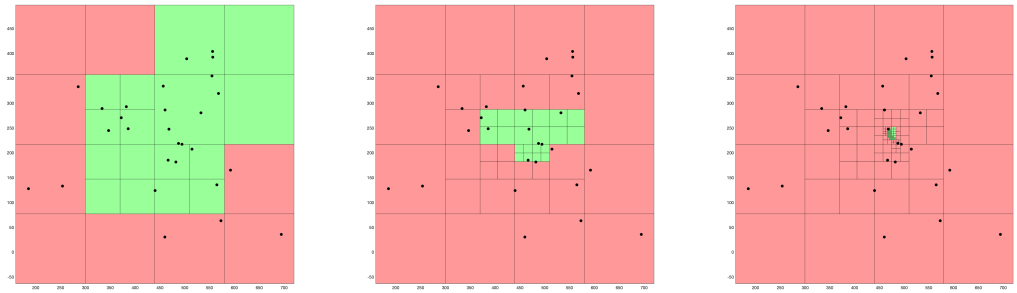


Figure 3 Illustrations of three different steps during the execution of Algorithm 1.

Algorithm 1: Subdivision for Fermat Point (*SUB*)

Input : Foci set A, $\varepsilon > 0$. **Output:** Point $\boldsymbol{p}$.

```
1 B_0 ← INITIAL-BOX(A);   Q ← QUEUE();   Q.PUSH(B_0);
2 while not C^ε(Q) do
3 |   B ← Q.POP();
4 |   if not □C^∇(B) then Q.PUSH(SPLIT(B)) ;
5 return p ← Center of the bounding box of Q;
```

3.2 Enhancing the Subdivision Paradigm

Using a Newton inclusion predicate, we know that our algorithm will eventually converge quadratically. Other authors have also considered Newton-type algorithms, but usually independently of other methods, thus lacking global convergence. We integrate subdivision with the Newton operator (an idea based on Dekker [12]), thus ensuring global convergence.

We want to find the Fermat point, i.e. the root of $\boldsymbol{f} = \nabla\varphi$. The Newton-type predicates are well-studied in the interval literature, and they have the form $N : \square\mathbb{R}^d \to \square\mathbb{R}^d$. We use the formula by Moore [23] and Nickel [26]: $N(B) = m_B - J_{\boldsymbol{f}}^{-1}(B) \cdot \boldsymbol{f}(m_B)$, where $J_{\boldsymbol{f}}$ is the Jacobian matrix of $\boldsymbol{f}$. Since $\boldsymbol{f} = \nabla\varphi$, the matrix $J_{\boldsymbol{f}}$ is actually the Hessian of φ. There are better Newton type operators [19, 17, 16] in the sense that they return a smaller box, but they are computationally more expensive.

This Newton box operator has the following properties, stemming from [7, 27, 34]: **(1)** If $N(B) \subset B$ then $\boldsymbol{x}^* \in N(B)$. **(2)** If $\boldsymbol{x}^* \in B$ then $\boldsymbol{x}^* \in N(B)$. **(3)** If $N(B) \cap B = \emptyset$ then $\boldsymbol{x}^* \notin B$.

▶ **Definition 3.3.** Given a box B, the *Newton inclusion predicate* $\square C^N(B)$ returns true if and only if $\square N(2B) \subset 2B$, where $\square N(B) = m_B - \square J_{\boldsymbol{f}}^{-1}(B) \cdot \boldsymbol{f}(m_B)$ see Section 4.

▶ **Theorem 3.4.** *Algorithms 1 and 2 return an ε-approximation of the Fermat point $\boldsymbol{x}^*$.*

Algorithm 2 initially excludes boxes only based on $\square C^\nabla$ and splits the other boxes. This subdivision phase takes exponential time in d. Once the test $\square C^N(B)$ succeeds, it will also do so for one of B's grandchildren. The Newton phase needs $O(kd^2)$ time per step and converges quadratically in ε. The time necessary to transition from the subdivision phase to the Newton phase does not depend on ε.

We can reduce the $O(k^2 d)$ preprocessing time (testing if $\boldsymbol{x}^* \in A$) to $O(kd)$ in both subdivision algorithms, by doing this test only when all except a constant number of foci got excluded by $\square C^\nabla$. Another speedup comes by preprocessing with a *principal component analysis* and then using rectangular boxes. Experimentally, we observe that it drastically improves the runtime for *near-degenerate* inputs.

Algorithm 2: Enhanced subdivision for Fermat Point (*E-SUB*)

Input : Foci set A, $\varepsilon > 0$. **Output:** Point $\boldsymbol{p}$.

$B_0 \leftarrow$ Initial-Box(A); $Q \leftarrow$ Queue(); Q.push(B_0);
while $C^{\varepsilon}(Q)$ **do**
  $B \leftarrow Q$.pop();
  if ***not*** $\square C^{\nabla}(B)$ **then**
    if $\square C^{N}(B)$ **then**
      $Q \leftarrow$ Queue(); Q.push(split$_2(N(2B))$;
    else Q.push(split(B));
return $\boldsymbol{p} \leftarrow$ *Center of the bounding box of* Q;

3.3 Using a Point Sequence Scheme

Weiszfeld [37] gave an iterative method to compute a sequence of points converging to the Fermat point, later corrected in [20, 29]. This scheme is defined by the following map:

$$T(\boldsymbol{x}) = \begin{cases} \tilde{T}(\boldsymbol{x}) & \text{if } \boldsymbol{x} \notin A \\ \dfrac{\sum_{\boldsymbol{a}\in A, \boldsymbol{a}\neq\boldsymbol{x}} w(\boldsymbol{a}) \frac{\boldsymbol{a}}{\|\boldsymbol{x}-\boldsymbol{a}\|}}{\sum_{\boldsymbol{a}\in A, \boldsymbol{a}\neq\boldsymbol{x}} w(\boldsymbol{a}) \frac{1}{\|\boldsymbol{x}-\boldsymbol{a}\|}} & \text{if } \boldsymbol{x} \in A \end{cases} \quad \text{where} \quad \tilde{T}(\boldsymbol{x}) = \frac{\sum_{i=1}^{k} w(\boldsymbol{a}) \frac{\boldsymbol{a}_i}{\|\boldsymbol{x}-\boldsymbol{a}_i\|}}{\sum_{i=1}^{k} w(\boldsymbol{a}) \frac{1}{\|\boldsymbol{x}-\boldsymbol{a}_i\|}} \tag{1}$$

We augment this by adding a guarantee for the computation, turning it into an ε-approximation algorithm. To verify the quality of a point $\boldsymbol{p}_i$, we use the Newton inclusion predicate. We define a small box B with $\boldsymbol{p}_i$ as center and map it to the new box $\square N(B)$. If $\square N(B) \subseteq B$, we know that $\boldsymbol{x}^*$ lies in $\square N(B)$. If $\square N(B) \not\subseteq B$, we move on to point $\boldsymbol{p}_{i+1}$ and adjust the box size. If there was a focus in box $\frac{B}{10}$, then $\square N(\frac{B}{10})$ covers the whole plane, which hinders $\square N(B) \subseteq B$ to succeed. In that case we shrink the box by a factor of 10. If $\frac{B}{10} \cap \square N(\frac{B}{10}) = \emptyset$, then the box $\frac{B}{10}$ does not contain the $\boldsymbol{x}^*$ and we therefore increase the box size. As a starting point, we take the *center of mass* $\boldsymbol{p}_0$ of A, i.e., $\boldsymbol{p}_0 = \frac{1}{W}\sum_{\boldsymbol{a}\in A} w(\boldsymbol{a})\boldsymbol{a}$, as $\varphi(\boldsymbol{p}_0)$ itself, is a 2-approximation of the Fermat radius [11].

Algorithm 3: Point sequence algorithm (*P-SEQ*)

Input : Foci set A, $\varepsilon > 0$. **Output:** Point $\boldsymbol{p}$.

$\boldsymbol{p} \leftarrow \boldsymbol{p}_0$; $w \leftarrow \varepsilon$;
while True **do**
  $B \leftarrow$ Box $B(m_B = p,\ \omega(B) = w)$; $N(B) \leftarrow$ Interval-Newton(B);
  if $N(B) \subseteq B$ **then return** $\boldsymbol{p}$;
  else if $N\left(\frac{B}{10}\right) \cap \frac{B}{10} = \emptyset$ **then** $w \leftarrow \min\{\varepsilon, w \cdot 10\}$;
  else $w \leftarrow \frac{w}{10}$;
  $\boldsymbol{p} \leftarrow T(\boldsymbol{p})$

The point sequence in Eq. (1) converges to the Fermat point and when sufficiently close, $\square N(B) \subset B$ holds, hence Algorithm 3 returns an ε-approximation of the Fermat point.

4 Details on the box approximations $\square\varphi$, $\square\nabla\varphi$, $\square\nabla^2\varphi$ and $\square N$

We call $L(B)$ a Lipschitz constant for box B if $\forall \boldsymbol{p}, \boldsymbol{q} \in B: \ |\varphi(\boldsymbol{p}) - \varphi(\boldsymbol{q})| \leq L(B) \cdot \|\boldsymbol{p} - \boldsymbol{q}\|$. We will later choose $L(B)$ smaller than the trivial Lipschitz constant $W = \sum_{\boldsymbol{a}\in A} w(\boldsymbol{a})$.

▶ **Lemma 4.1.** $\square\varphi(B) = [\varphi(m_B) - L(B) \cdot r_B, \varphi(m_B) + L(B) \cdot r_B]$ *is a soft version of* φ.

Proof. The $L(B)$ is a Lipschitz constant of φ on box B, i.e. for all $p \in B$ it holds $|\varphi(p) - \varphi(m_B)| \leq L(B) \cdot r_B$, which implies $\varphi(p) \in [\varphi(m_B) - L \cdot r_B, \varphi(m_B) + L \cdot r_B]$.

Let B_n be a sequence of boxes, which converges to a point. Hence $r_{B_n} \to 0$. The Lipschitz constants $L(B_n)$ can be bounded from above by W. Thus, $\omega(\square\varphi(B_n)) \leq 2W \cdot r_{B_n} \to 0$. ◀

The following formulas are written for $d = 2$ but clearly generalize to higher dimensions. For any point $\boldsymbol{p} = (\boldsymbol{p}_x, \boldsymbol{p}_y)^T$, let $\sin(\boldsymbol{p}) := \boldsymbol{p}_y / \|\boldsymbol{p}\|$ and $\cos(\boldsymbol{p}) := \boldsymbol{p}_x / \|\boldsymbol{p}\|$. Clearly,

$$\nabla\varphi(p) = \begin{pmatrix} \sum_{\boldsymbol{a}\in A} w(\boldsymbol{a}) \cos(\boldsymbol{p} - \boldsymbol{a}) \\ \sum_{\boldsymbol{a}\in A} w(\boldsymbol{a}) \sin(\boldsymbol{p} - \boldsymbol{a}) \end{pmatrix}. \tag{2}$$

Let $Cor(B)$ denote the set of four corners of B. Then

$$\sin(B - \boldsymbol{a}) = \begin{cases} [-1, 1] & \text{if } \ \boldsymbol{a} \in B, \\ [\min(\sin(Cor(B) - \boldsymbol{a})), 1] & \text{if } \ \boldsymbol{a}_x \in B_x \wedge \boldsymbol{a}_y < B_y, \\ [-1, \max(\sin(Cor(B) - \boldsymbol{a}))] & \text{if } \ \boldsymbol{a}_x \in B_x \wedge \boldsymbol{a}_y > B_y, \\ [\min(\sin(Cor(B) - \boldsymbol{a})), \max(\sin(Cor(B) - \boldsymbol{a}))] & \text{else.} \end{cases} \tag{3}$$

In other words, $\sin(B - \boldsymbol{a})$ can be computed from the sinus of at most four angles. Similarly for $\cos(B - \boldsymbol{a})$.

For any $p \in \mathbb{R}^2 \setminus A$ it holds:

$$\nabla^2\varphi(\boldsymbol{p}) = \begin{pmatrix} \sum_{\boldsymbol{a}\in A} w(\boldsymbol{a}) \frac{(\boldsymbol{p}_y - \boldsymbol{a}_y)^2}{\|\boldsymbol{p}-\boldsymbol{a}\|^3} & -\sum_{\boldsymbol{a}\in A} w(\boldsymbol{a}) \frac{(\boldsymbol{p}_x - \boldsymbol{a}_x)(p_y - \boldsymbol{a}_y)}{\|\boldsymbol{p}-\boldsymbol{a}\|^3} \\ -\sum_{\boldsymbol{a}\in A} w(\boldsymbol{a}) \frac{(\boldsymbol{p}_x - \boldsymbol{a}_x)(p_y - \boldsymbol{a}_y)}{\|\boldsymbol{p}-\boldsymbol{a}\|^3} & \sum_{\boldsymbol{a}\in A} w(\boldsymbol{a}) \frac{(\boldsymbol{p}_x - \boldsymbol{a}_x)^2}{\|\boldsymbol{p}-\boldsymbol{a}\|^3} \end{pmatrix} \tag{4}$$

▶ **Lemma 4.2.** *Soft versions* $\square\nabla\varphi(B)$ *and* $\square\nabla^2\varphi(B)$ *are derived by replacing every occurrence of* p *in Eqs.* (2) *and* (4) *by* B *and evaluating with Eq.* (3) *and interval arithmetic.*

A box approximation of the length of the gradient of φ can then be achieved by:

$$\square\,\|\nabla\varphi(B)\| = \left\| \sum_{\boldsymbol{a}\in A\setminus B} w(\boldsymbol{a}) \begin{pmatrix} \cos(B - \boldsymbol{a}) \\ \sin(B - \boldsymbol{a}) \end{pmatrix} \right\| + \left[-\sum_{\boldsymbol{a}\in A\cap B} w(\boldsymbol{a}), \sum_{\boldsymbol{a}\in A\cap B} w(\boldsymbol{a}) \right]$$

where the length of an interval vector $I = (I_x, I_y)$ is computed by $\|I\| = \sqrt{I_x^2 + I_y^2}$ and the square root of an interval J by $\sqrt{J} = \left[\sqrt{\min|J|}, \sqrt{\max|J|}\right]$. We use the Lipschitz constant $L(B) = \min\{W, \max \square\,\|\nabla\varphi(B)\|\}$ of box B to compute $\square\varphi(B)$.

▶ **Lemma 4.3.** *The soft gradient test* $\square C^{\nabla}(B)$ *is convergent, i.e., for any monotone sequence of boxes* $(B_n)_{n\in\mathbb{N}}$ *that converges to a point* $\boldsymbol{p}$*, the point* $\boldsymbol{p}$ *is not the Fermat point iff* $\square C^{\nabla}(B_n) = \texttt{success}$ *for* n *large enough.*

We compute $\square N(B) = m_B - (\square\nabla^2\varphi(B))^{-1} \cdot \nabla\varphi(m_B)$ and the inverse of an interval matrix through interval arithmetic: $\begin{pmatrix} I & J \\ K & L \end{pmatrix}^{-1} = \frac{1}{IL-JK} \begin{pmatrix} L & -J \\ -K & I \end{pmatrix}$.

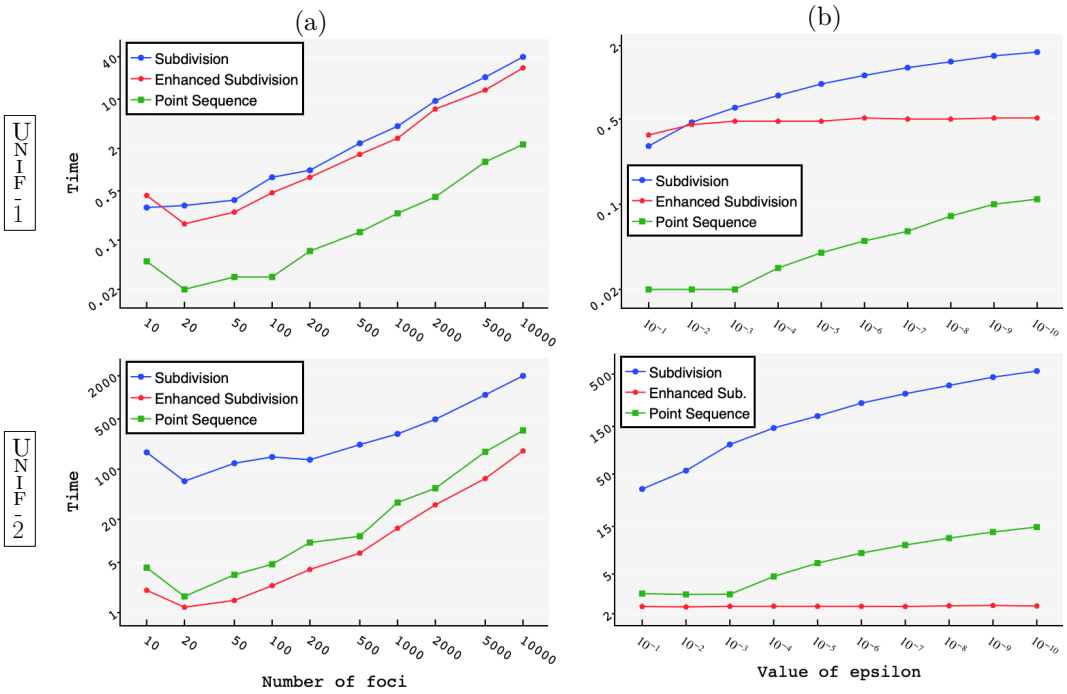


Figure 4 Experimental results. Points are, in UNIF-1, uniformly sampled from a disk and, in UNIF-2, uniformly sampled from two disjoint disks. Finding the Fermat point with times as function of (a) k, with $\varepsilon = 10^{-4}$ and (b) ε, with $k = 100$. Axes are in logarithmic scale.

5 Experiments and Conclusion

We have implemented our algorithms in two dimensions using Matlab. We have experimented with various datasets. An overview of experiments on synthetic datasets is shown in Fig. 4.

The runtime of all algorithms depends linearly on the number of foci. When ε is sufficiently small, then the enhanced subdivision algorithm outperforms the point sequence algorithm due to its quadratic convergence near the Fermat point.

References

1 A. Karim Abu-Affash and Matthew J. Katz. Improved bounds on the average distance to the Fermat-Weber center of a convex object. *Information Processing Letters*, 109(6):329–333, 2009.

2 Mihai Badoiu, Sariel Har-Peled, and Piotr Indyk. Approximate clustering via core-sets. In *Proc. Symposium on Theory of Computing*, pages 250–257. ACM, 2002.

3 Chanderjit Bajaj. The algebraic degree of geometric optimization problems. *Discrete & Computational Geometry*, 3(2):177–191, 1988.

4 Huck Bennett and Chee Yap. Amortized analysis of smooth quadtrees in all dimensions. *Computational Geometry*, 63:20–39, 2017.

5 Bhaswar B. Bhattacharya. On the Fermat-Weber point of a polygonal chain and its generalizations. *Fundamenta Informaticae*, 107(4):331–343, 2011.

6 Prosenjit Bose, Anil Maheshwari, and Pat Morin. Fast approximations for sums of distances, clustering and the Fermat-Weber problem. *Computational Geometry*, 24(3):135–146, 2003.

7 Luitzen Egbertus Jan Brouwer. Über Abbildung von Mannigfaltigkeiten. *Mathematische Annalen*, 71(1):97–115, 1911.

8 Paz Carmi, Sariel Har-Peled, and Matthew J. Katz. On the Fermat-Weber center of a convex object. *Computational Geometry*, 32(3):188–195, 2005.

9 Hui Han Chin, Aleksander Madry, Gary L. Miller, and Richard Peng. Runtime guarantees for regression problems. In *Proc. Innovations in Theoretical Computer Science*, pages 269–282. ACM, 2013.

10 Ernest J. Cockayne and Zdzislaw A. Melzak. Euclidean constructibility in graph-minimization problems. *Mathematics Magazine*, 42(4):206–208, 1969.

11 Michael B. Cohen, Yin Tat Lee, Gary L. Miller, Jakub Pachocki, and Aaron Sidford. Geometric median in nearly linear time. In *Proc. Symposium on Theory of Computing*, pages 9–21. ACM, 2016.

12 Theodorus Jozef Dekker. Finding a zero by means of successive linear interpolation. In *Constructive Aspects of the Fundamental Theorem of Algebra*, pages 37–48. Wiley Interscience, 1967.

13 Adrian Dumitrescu, Minghui Jiang, and Csaba D. Tóth. New bounds on the average distance from the Fermat-Weber center of a planar convex body. *Discrete Optimization*, 8(3):417–427, 2011.

14 Sándor P Fekete, Joseph SB Mitchell, and Karin Beurer. On the continuous Fermat-Weber problem. *Operations Research*, 53(1):61–76, 2005.

15 Dan Feldman and Michael Langberg. A unified framework for approximating and clustering data. In *Proc. Symposium on Theory of Computing*, pages 569–578. ACM, 2011.

16 Alexandre Goldsztejn. Comparison of the Hansen-Sengupta and the Frommer-Lang-Schnurr existence tests. *Computing*, 79(1):53–60, 2007.

17 Eldon R. Hansen and R.I. Greenberg. An interval Newton method. *Applied Math. and Computation*, 12:89–98, 1983.

18 Sariel Har-Peled and Akash Kushal. Smaller coresets for k-median and k-means clustering. *Discrete & Computational Geometry*, 37(1):3–19, 2007.

19 Rudolf Krawczyk. Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken. *Computing*, 4(3):187–201, 1969.

20 Harold W. Kuhn. A note on Fermat's problem. *Mathematical programming*, 4(1):98–107, 1973.

21 Long Lin and Chee Yap. Adaptive isotopic approximation of nonsingular curves: the parameterizability and nonlocal isotopy approach. *Discrete & Computational Geometry*, 45(4):760–795, 2011.

22 Luis Fernando Mello and Lucas Ruiz dos Santos. On the location of the minimum point in the Euclidean distance sum problem. *São Paulo Journal of Mathematical Sciences*, 12:108–120, 2018.

23 Ramon E Moore. *Interval Analysis*, volume 4. Prentice-Hall Englewood Cliffs, NJ, 1966.

24 Kent E Morrison. The fedex problem. *The College Mathematics Journal*, 41(3):222–232, 2010.

25 Gyula Sz Nagy. Tschirnhaus'sche Eiflächen und Eikurven. *Acta Mathematica Academiae Scientiarum Hungarica*, 1(1):36–45, 1950.

26 Karl Nickel. Triplex-algol and applications. *Interner Bericht des Instituts für Informatik der Universität Karlsruhe*, 1969.

27 Karl Nickel. On the Newton method in interval analysis. Technical report, Wisconsin University-Madison Mathematics Research Center, 1971.

28 Jiawang Nie, Pablo A. Parrilo, and Bernd Sturmfels. Semidefinite representation of the k-ellipse. In *Algorithms in algebraic geometry*, pages 117–132. Springer, 2008.

29 Lawrence M Ostresh Jr. Convergence and descent in the Fermat location problem. *Transportation Science*, 12(2):153–164, 1978.

30 Pablo A. Parrilo and Bernd Sturmfels. Minimizing polynomial functions. *Algorithmic and quantitative real algebraic geometry, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 60:83–99, 2003.

31 Helmut Ratschek and Jon Rokne. *Computer methods for the range of functions*. Horwood, 1984.

32 Hanan Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, 1990.

33 Junpei Sekino. n-ellipses and the minimum distance sum problem. *The American mathematical monthly*, 106(3):193–202, 1999.

34 Sergey P Shary. Krawczyk operator revised. *Novosibirsk, Institute of Computational Technologies, Rússia*, 2004.

35 Rudolf Sturm. Über den Punkt kleinster Entfernungssumme von gegebenen Punkten. *Journal für die reine und angewandte Mathematik*, 97:49–61, 1884.

36 Cong Wang, Yi-Jen Chiang, and Chee Yap. On soft predicates in subdivision motion planning. *Computational Geometry: Theory and Applications.*, 48(8):589–605, September 2015.

37 Endre Weiszfeld. Sur le point pour lequel la somme des distances de n points donnés est minimum. *Tohoku Mathematical Journal, First Series*, 43:355–386, 1937.

38 Guoliang Xue and Yinyu Ye. An efficient algorithm for minimizing a sum of Euclidean norms with applications. *SIAM Journal on Optimization*, 7(4):1017–1036, 1997.

Repulsion Region in a Simple Polygon*

Arthur van Goethem[1], Irina Kostitsyna[1], Kevin Verbeek[1], and Jules Wulms[1]

1 **Department of Mathematics and Computer Science, TU Eindhoven, the Netherlands**
{a.i.v.goethem|i.kostitsyna|k.a.b.verbeek|j.j.h.m.wulms}@tue.nl

Abstract

We study a motion planning problem for point objects controlled by an external repulsive force. Given a point object ζ inside a simple polygon P, a *repulsor* r (also represented by a point in P) moves ζ by always pushing it away from r. When ζ hits the boundary of P, then ζ slides along the boundary continuously increasing the distance to r until this is no longer possible. For a fixed polygon P and starting position s of ζ inside P, we define the *repulsion region* as the set of points in P that can be reached by ζ by placing a repulsor r anywhere inside P (and keeping the position of r fixed throughout the motion). In this paper we show that, for a specific class of polygons, the worst case complexity of the repulsion region is $\Theta(n^2)$ and the repulsion region can be computed in $O(n^2 \log n)$ time, where n is the complexity of the polygon.

1 Introduction

Algorithmic path planning and motion control questions are usually studied in the context of the actuating abilities of (autonomous or externally controlled) mobile objects. The object is put in motion by the forces produced from within, by its actuators. A very different approach is to consider a static object (or a set of objects), with no means of producing any kind of motion itself, controlled by external global forces. There has been a number of papers exploring motion planning of point objects by external control. Becker et al. [3] study the question of controlling a swarm of particles in a geometric environment by applying a sequence of global forces, such as gravity or a homogeneous magnetic field. Aloupis et al. [2] consider the problem of rolling a ball out of a polygon (or equivalently, draining a polygon filled with water) by tilting it. Akitaya et al. [1] consider the trash compaction problem, where a set of trash particles are pushed together with a sweep line to form a compact shape.

One commonality among these papers is that, at any given moment, the force applied to an object at any possible location has the same direction. In contrast, in the *beacon attraction* model [4, 5], a point magnet, called beacon, is placed inside the polygon which exerts a magnetic pull towards itself on all the points of the polygon. The forces applied to the objects at different locations are no longer parallel, which leads to rather particular geometric properties. Specifically, a *beacon attraction region*, i.e., all the points of the polygon that eventually reach a given beacon and not get stuck along the way, has very different properties than an *inverse beacon attraction region*, i.e., all the beacon locations that a given point can eventually reach and not get stuck along the way [4, 5, 8, 9].

Inspired by the beacon attraction model, Bose and Shermer [6] introduce the model of repulsion, where a repulsion actuator exerts a magnetic repulsive force on the points of a

* Kevin Verbeek is supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 639.021.541. Jules Wulms is supported by the Netherlands eScience Center (NLeSC) under project no. 027.015.G02. Research on the topic of this abstract was initiated at the 4th Workshop on Applied Geometric Algorithms (AGA 2018) in Langbroek, The Netherlands, supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 639.023.208.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

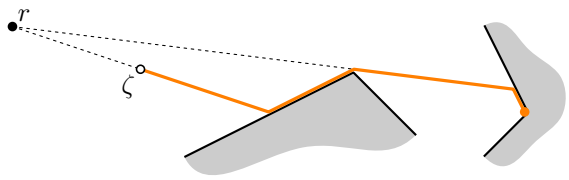


Figure 1 Repulsor r acts on the particle ζ. If the direction away from the repulsor is interior to the polygon, ζ moves away from the repulsor along a straight line. Otherwise, if possible, ζ slides along a boundary segment in the direction of increasing Euclidean distance.

polygon. They consider the question whether it is possible to gather all the points of a convex polygon in one point with one repulsion actuator. Mozafari and Shermer [10] study the problem of finding a set of *acceptable* points for a given target location t, i.e., the set of points that can be pushed to t with one repulsion actuator.

We continue to explore the repulsion model and study the complementary problem: given a starting position s, find all the points *reachable* from s with one repulsion actuator.

Problem statement and contributions. Consider a repulsion activator (or *repulsor*) r and a particle ζ inside a polygon P. Under the repulsion of r, the particle ζ moves away from r along a straight line as long as ζ is interior to P, even if r is not visible to ζ (refer to Figure 1). When ζ reaches the boundary of P, it slides along it in the direction of increasing Euclidean distance from r, if such a direction exists. Otherwise, the motion of the particle terminates.

Given a polygon P with n vertices and a point s in it, we define the *repulsion region* $R(s)$ as the set of points $p \in P$ for which there exists a repulsor $r \in P$ such that a particle placed at s will eventually reach the point p under the influence of the repulsor r. In this abstract we restrict the attention to a specific class of polygons, which we define in Section 2, along with some basic notation and properties. In Section 3 we show that the repulsion region has worst-case complexity $\Theta(n^2)$. Finally, we show in Section 4 that the repulsion region can be computed in $O(n^2 \log n)$ time. Although we believe these results hold for all simple polygons, proving so is significantly harder, and therefore left for the full version of this paper.

2 Preliminaries

Consider a particle ζ, currently at position p_ζ, that is moved by a repulsor r inside a simple polygon P. By definition, the Euclidean distance between r and ζ is monotonically increasing throughout the motion. For an edge e of P not containing p_ζ, consider the line ℓ perpendicular to e passing through p_ζ. If this line intersects e in a point h, then we call h the *split point* of e with respect to the current position of ζ (see Fig. 2 (left)). Note that, if r is strictly on one side of ℓ and pushes ζ onto e, then ζ will hit e on the opposite side of ℓ and continue moving away from h.

Furthermore, consider a reflex vertex v of P with two incident edges e_1 and e_2. Extend the edges e_1 and e_2 beyond v. When ζ lies inside the resulting cone, then a repulsor may push it towards either of the edges. If the segment $\overline{p_\zeta v}$ splits the interior angle at v into two obtuse angles, then we call v a *split vertex* with respect to the current position of ζ.

Now consider a particle ζ sliding along an edge e_1 towards a convex vertex v (see Fig. 2 (right)). Let e_2 be the other edge incident to v, and let ℓ be the line perpendicular to e_2 going through v. Let H_1 and H_2 be the half planes defined by ℓ containing e_1 and e_2, respectively.

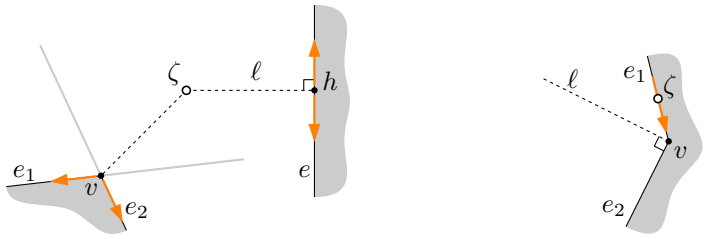


Figure 2 Left: point h is the split point of e, and reflex vertex v is a split vertex, with respect to the current location of particle ζ. Right: ζ is sliding along e_1 towards v, ℓ is perpendicular to e_2. If the repulsor is above ℓ, then ζ will continue sliding along e_2, otherwise it will remain in v.

If the repulsor pushing ζ towards v lies in H_1 then ζ will continue sliding along e_2 after it passes v. However, if the repulsor is in H_2 then the motion of ζ will terminate at v.

In the (degenerate) case that a repulsor is located exactly on the line through ζ and a split vertex/point with respect to the current position of ζ, the particle moves onto either of the two incident edges. This choice does not significantly impact the repulsion region.

Finally, we denote a path followed by the particle ζ (starting at s) repulsed by a repulsor r by π_r. Observe that π_r must be simple. Let π_1 and π_2 be two paths generated by repulsors r_1 and r_2, respectively. We say that π_1 and π_2 have a *convergence point* c, if $c \in \pi_1$ and $c \in \pi_2$. Note that the starting position s is a convergence point of all paths. If two paths have a convergence point other than s, then we say that they are *converging*. Furthermore, we say that π_1 and π_2 *properly* intersect in convergence point c if c is interior to P.

Regular polygons. We call a polygon *regular* if (1) looking at the split points/vertices with respect to s, the angle between consecutive split points/vertices around s is at most π, and (2) there does not exist a repulsor r such that we can identify two points p_1, p_2 on π_r for which $r \in \overline{p_1p_2}$ and $\overline{p_1p_2}$ is completely contained in P; otherwise the polygon is *irregular* (see Fig. 3(a) and (b)). Intuitively, the path π_r of a repulsor r cannot spiral around r in regular polygons. Contrary to regular polygons (see Section 3), paths of repulsors may properly intersect in irregular polygons, making the analysis significantly harder. Furthermore, we obtain the following useful property in regular polygons, which again, does not hold generally for irregular polygons.

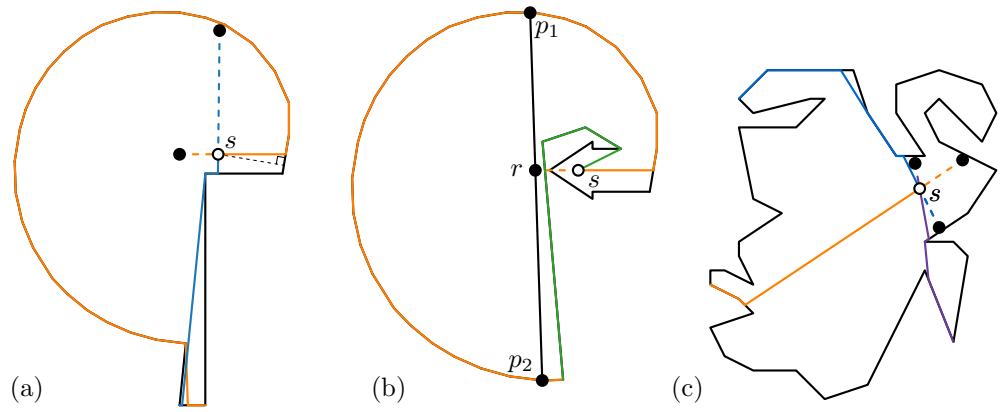


Figure 3 Regular vs irregular polygons: (a) condition (1) is violated and paths cross; (b) condition (2) is violated and geodesic distance to s decreases; (c) a regular polygon.

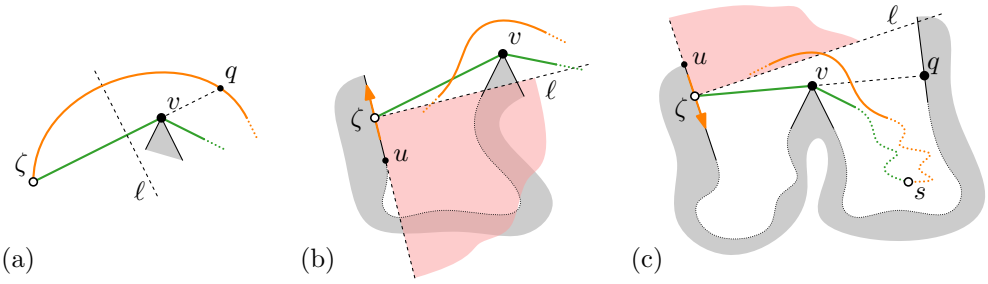


Figure 4 Illustration of Lemma 1. In a regular polygon the geodesic distance from ζ to s is monotonically increasing. Geodesic path is shown in green, path π_r in orange.

▶ **Lemma 1.** *In a regular polygon P, the geodesic distance inside P from a particle ζ to its starting position s is monotonically increasing.*

Proof sketch. Consider the geodesic path γ from s to a point p_ζ, the current position of ζ. Let v be the last vertex on γ before p_ζ. We consider several cases (see Fig. 4). If p_ζ is interior to P, then the result follows from the fact that the distance from ζ to r is monotonically increasing. If p_ζ is on the boundary of P, then let $\overline{vp_\zeta}$ split the polygon into two subpolygons P_1 and P_2, where P_1 contains s. If ζ is pushed into P_1, then the geodesic distance between ζ and s cannot decrease. Otherwise, the geodesic distance between ζ and s may only decrease if π_r spirals around r, but then P is irregular. ◀

3 Complexity of the repulsion region

To prove an upper bound on the complexity of the repulsion region, we first show that the paths of two repulsors cannot properly intersect in a regular polygon P.

▶ **Lemma 2.** *In a regular polygon P no two repulsion paths have a proper intersection.*

Proof sketch. Let π_1 and π_2 be the paths generated by two repulsors r_1 and r_2, and let ℓ be the line through r_1 and r_2. Assume that c and d are two consecutive convergence points of π_1 and π_2. We can show that c and d must be on the same side of ℓ, for otherwise there must exist another convergence point between c and d at ℓ. Additionally, π_1 and π_2 cannot cross ℓ between c and d. Then, due to the relative angles from d to r_1 and r_2, there cannot be a proper intersection at d. By repeating this argument, we conclude that π_1 and π_2 do not have a proper intersection. ◀

Now let v be a (reflex) vertex of P and consider two repulsors r_1 and r_2 for which the generated paths π_1 and π_2 contain v. Consider a repulsor r_3 that lies on the segment $\overline{r_1r_2}$ (potentially outside P) and its repulsion path π_3. Since two repulsion paths cannot properly intersect, π_3 is essentially contained between π_1 and π_2, and hence must also contain v. We obtain the following result.

▶ **Corollary 3.** *The set of repulsors $S(v)$ (possibly outside of the polygon) that push the particle ζ to a reflex vertex v is convex.*

Consequently, the set of repulsors $S'(v)$ inside P that push the particle to a (reflex) vertex v consists of at most $O(n)$ connected components. To construct part of the repulsion region starting from v, we can simply combine the at most $O(n)$ cones emanating from v coming

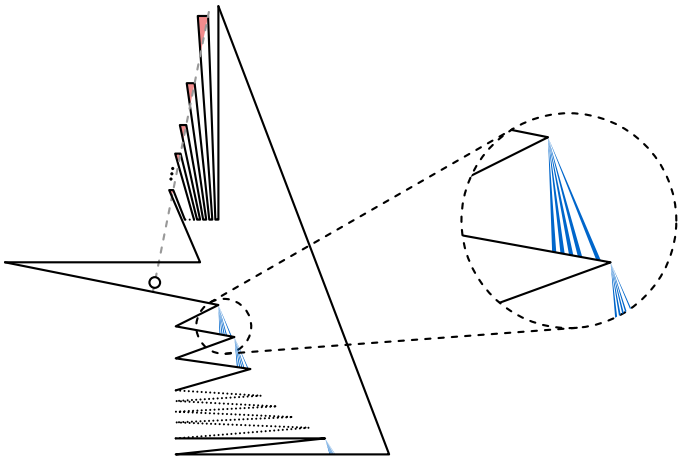


Figure 5 Polygon with $O(n)$ towers and $O(n)$ cliffs. The repulsion region we get by placing repulsors in the pink areas at the top, results in blue $O(n)$ cones per reflex vertex at the bottom.

from the $O(n)$ connected components of $S'(v)$. The complete repulsion region is then the union of all cones over all reflex vertices of P, as the path of the particle either follows the boundary, or moves internally through P starting from a reflex vertex. Since the cones cannot intersect, we get the following upper bound. This bound is tight, as shown by the example in Fig. 5.

▶ **Theorem 4.** *The repulsion region of a regular polygon P has worst-case complexity $\Theta(n^2)$, where n is the number of vertices of P.*

4 Computing the repulsion region

To compute the repulsion region, we must explicitly compute the corresponding convex sets for each (reflex) vertex of the regular polygon P. To compute the convex sets in the right order, we heavily rely on Lemma 1. Specifically, we construct the *shortest path map* (SPM) in P with source s (following the approach of [8]). This SPM is a subdivision of P which captures all possible shortest paths starting from s. For every region R_i of the SPM, there is a reflex vertex v_i (or s) that is the last vertex on the shortest path from s to a point $p \in R_i$. We call v_i the base of R_i. We now add the following vertices to P. For every reflex vertex v_i, we extend the last segment of the shortest path from s to v_i until it hits the boundary of P at w_i. The segment $\overline{v_i w_i}$ is called the *window* of R_i and lies on the boundary of R_i. We add the window vertex w_i to P. Furthermore, for every edge e in R_i, we add the split point of e with respect to v_i to P, if it exists. Finally, we construct the directed *repulsion graph* $G = (V, E)$, where V consists of all vertices, window vertices, and split points of P. The edges of G include all edges of the shortest path tree in P with source s, and the edges on the boundary of P directed towards larger geodesic distance from s (which is well defined, since we split P at split points). See Figure 6 for an example.

We can argue the following properties for any path π_r of a repulsor r: (1) π_r can enter a region R_i only via its base v_i or its window vertex w_i , and (2) if π_r reaches the base v_i of a region R_i, then it must proceed in R_i.As a result, the repulsion graph G represents the combinatorial structures of all possible paths π_r from s through P. Now, to construct the convex set for a particular vertex v of P, we can simply consider the incoming edges in G.

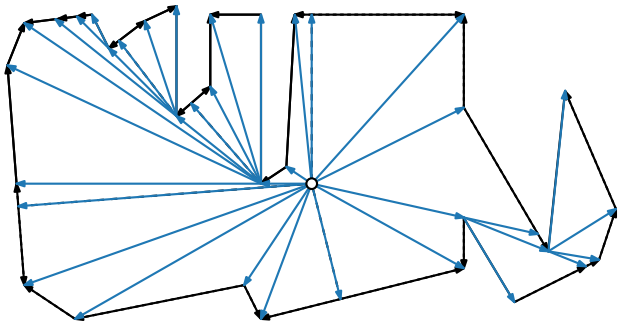


Figure 6 Repulsion graph G consisting of black edges along the boundary of P, and blue shortest path tree edges.

Typically, sliding along some incident edge of P towards v induces an additional half-plane constraint, as described in Section 2. Finally, we may need to combine the resulting convex sets from various incoming edges. However, since the union of these convex sets must again be convex, this simply means that some half-plane constraints can be eliminated, which is easy to compute.We can then obtain the following result.

▶ **Theorem 5.** *The repulsion region $R(s)$ of a regular polygon P can be computed in $O(n^2 \log n)$ time, where n is the number of vertices of P.*

Proof sketch. We first compute the SPM with source s in $O(n)$ time [7]. From the SPM we can easily compute the repulsion graph G in $O(n)$ time. Note that G has $O(n)$ complexity.

We then compute the convex set of repulsors for each vertex v of G, in order of increasing geodesic distance from s. For each incoming edge of v in G, we add the corresponding half-plane constraint to the convex set of repulsors reaching that edge and ending in v. Next, we combine the resulting convex sets for all incoming edges, thus constructing the convex set for v. Since each convex set may have $O(n)$ complexity, computing the convex set for v takes $O(nd_v)$ time, where d_v is the in-degree of v in G. Thus, we can compute the convex sets for all vertices in $\sum_v O(nd_v) = O(n^2)$ time.

Finally, we intersect the convex set of each vertex v with P in $O(n \log n)$ time using the algorithm in [11]. We can then easily compute the repulsion region $R(s)$ by extending the corresponding cones from each reflex vertex v. Thus, the repulsion region $R(s)$ can be computed in $O(n^2 \log n)$ time. ◀

References

1 Hugo Akitaya, Greg Aloupis, Maarten Löffler, and Anika Rounds. Trash compaction. In *Proc. 32nd European Workshop on Computational Geometry*, 2016.

2 Greg Aloupis, Jean Cardinal, Sébastien Collette, Ferran Hurtado, Stefan Langerman, and Joseph O'Rourke. Draining a polygon—or—rolling a ball out of a polygon. *Computational Geometry*, 47(2):316–328, 2014.

3 Aaron Becker, Erik Demaine, Sándor Fekete, Jarrett Lonsford, and Rose Morris-Wright. Particle computation: complexity, algorithms, and logic. *Natural Computing*, 18(1):181–201, 2019.

4 Michael Biro. *Beacon-based routing and guarding.* PhD thesis, Stony Brook University, 2013.

5 Michael Biro, Justin Iwerks, Irina Kostitsyna, and Joseph Mitchell. Beacon-based algorithms for geometric routing. In *Proc. 13th Algorithms and Data Structures Symposium (WADS)*, 2013.

6 Prosenjit Bose and Thomas Shermer. Gathering by repulsion. In *Proc. 16th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, pages 13:1–13:12, 2018.

7 Leonidas Guibas, John Hershberger, Daniel Leven, Micha Sharir, and Robert Endre Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2:209–233, 1987.

8 Irina Kostitsyna, Bahram Kouhestani, Stefan Langerman, and David Rappaport. An optimal algorithm to compute the inverse beacon attraction region. In *Proc. 34th International Symposium on Computational Geometry (SoCG)*, pages 55:1–55:14, 2018.

9 Bahram Kouhestani, David Rappaport, and Kai Salomaa. On the inverse beacon attraction region of a point. In *Proc. 27th Canadian Conference on Computational Geometry (CCCG)*, 2015.

10 Amirhossein Mozafari and Thomas Shermer. Transmitting particles in a polygonal domain by repulsion. In *Proc. 12th International Conference Combinatorial Optimization and Applications (COCOA)*, pages 495–508, 2018.

11 Ari Rappoport. An efficient algorithm for line and polygon clipping. *The Visual Computer*, 7(1):19–28, 1991.

The angular blowing-a-kiss problem

Kevin Buchin, Irina Kostitsyna, Roel Lambers, and Martijn Struijs

Department of Mathematics and Computing Science, TU Eindhoven, The Netherlands
k.a.buchin@tue.nl, i.kostitsyna@tue.nl, r.lambers@tue.nl, m.a.c.struijs@tue.nl

Abstract

Given a set of agents that have fixed locations but can rotate at unit speed, we aim to find an efficient schedule such that every pair of agents has looked at each other. We present schedules and lower bounds for different geometric settings.

1 Introduction

Given n people in a rectangular room, the *kissing problem* asks for the most efficient way for each pair of people to kiss each other goodbye [1]. We consider the variant of the problem in which the people blow kisses instead. Rather than changing locations, people now only need to turn to face each other.

Our motivation to study this problem comes from the research performed at NASA ARC on probing the magnetosphere of the Earth by a swarm of satellites with the use of directional antennas [2]. To perform probing, two satellites need to orient the antennas towards each other to be able to send and receive data. In this context we are interested in the most efficient schedule that allows every pair of satellites to perform probing. Independently, Fekete et al. [4] have studied this setting, focusing on the case in which only a subset of the satellite pairs need to communicate.

Problem Statement. In this paper, an *agent* has a fixed location in the plane and a heading direction that it can change over time at unit speed. We will refer to an agent by its location.

The input is a set of n agent locations $p_1, \dots, p_n$ in the plane. Note that we allow agents to choose their initial direction. A pair of agents p_i, p_j can *scan* each other at time t if p_j is in the direction of p_i, and vice versa. We also say, the pair of agents is scanned. The goal is to define valid schedules for the agents as to minimize the time to scan all pairs of agents.

We define a schedule for all agents by defining a schedule per agent. A schedule for agent p_i is an ordering of the other agents $\Pi^i = \langle \pi^i_1, \dots, \pi^i_{n-1} \rangle$ with a time t^i_j associated with each of the agents in the order. A schedule for the agents is valid if

1. $t^i_j \le t^i_{j+1}$, for all i and for $0 \le j < n$,
2. $\measuredangle \pi^i_j, p_i, \pi^i_{j+1} \le t^i_{j+1} - t^i_j$, for all i and for $0 \le j < n$, where $\measuredangle BAC$ denotes the smaller angle between B and C at A,
3. if $p_j = \pi^i_k$ and $p_i = \pi^j_\ell$, then $t^i_k = t^j_\ell$.

The objective of the *blowing-a-kiss problem* is to find a valid schedule S that minimizes $t(S) = \max_{1 \le i \le n} t^i_{n-1}$, i.e., the time until all pairs of agents have looked at each other.

We distinguish two models. In the *asynchronous model*, looking at each other can happen at any time. In the *synchronous model* scans need to be synchronized with $\lfloor n/2 \rfloor$ disjoint pairs being scanned at the same time. More specifically, the schedule has *rounds* $r_1, \dots, r_N$, where $N = n$ if n is odd, and $N = n - 1$ if n is even. In each round an agent can scan one other agent. If n is odd, in every round one agent does not scan, we say this agent has a *bye* or is a *bye agent*. Each round r_i has a timestamp t_i associated with it with $t_i \le t_{i+1}$. We define the *distance* between two rounds as $d(r_i, r_j) = |t_j - t_i|$.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

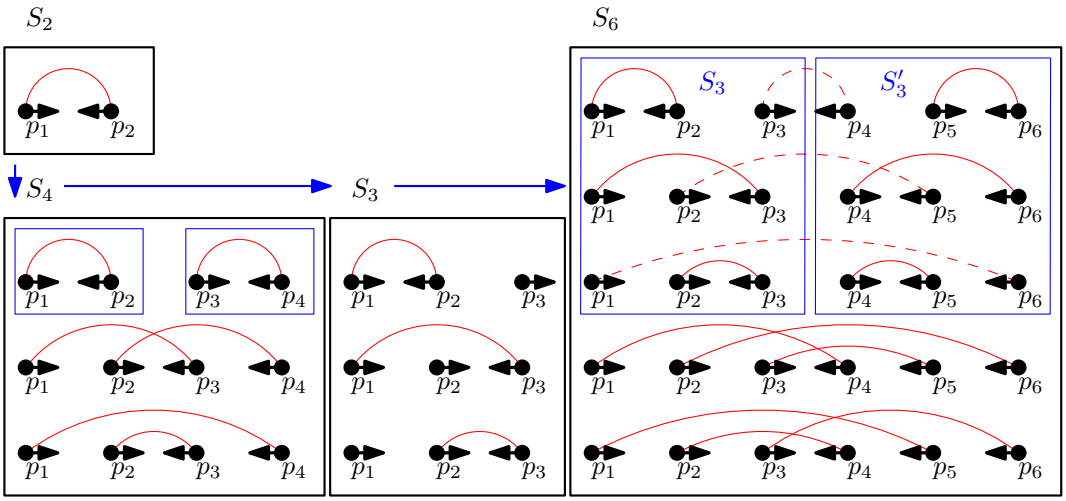


Figure 1 S_6 is constructed via S_2, S_4, S_3. In S_6 the byes from S_3 scan those from S'_3 (dashed)

Results. For the case that all agents are on a line, we present a schedule in both models that takes $\pi(\lceil \log n \rceil - 1)$ time, which we prove is optimal. For agents regularly spaced on a circle we present a schedule in the asynchronous model that takes $\pi(\lceil \log n \rceil - 1) + o(1)$ time, which we prove is near optimal. In the synchronous model and for n being a power of 2, we present a schedule that takes at most $2\pi \log n$ time. For the general two-dimensional case we present a schedule in the asynchronous model that takes $\frac{3\pi}{2}\lceil \log n \rceil - \frac{\pi}{2}$ time.

Related work. For the general two-dimensional case, Fekete et al. [4] independently obtained a schedule with the same cost, and additionally prove a lower bound for this case. A related geometric problem is the angular freeze-tag problem [3].

2 Schedules

2.1 Line

In this case the agents $p_1, \ldots, p_n$ are on one line, given from left to right. When facing other agents, an agent has an *orientation*, and it is either oriented to the left or to the right. The time it takes to change the orientation is π.

In the following we construct a schedule S in the synchronous model with $t(S) = \pi(\lceil \log_2 n \rceil - 1)$.The strategy relies on the following lemmata.

▶ **Lemma 2.1.** *Given a schedule S_n for some even n with $t(S_n) = \pi(\lceil \log n \rceil - 1)$, we can construct a schedule S_{n-1} for $n-1$ agents with time $t(S_{n-1}) = \pi(\lceil \log(n-1) \rceil - 1)$ where all bye agents are oriented to the right.*

Proof. Given S_n, remove agent p_n and give a bye to an agent when they would have scanned p_n. This is a valid schedule for $n-1$ agents with time $\pi(\lceil \log n \rceil - 1) = \pi(\lceil \log(n-1) \rceil - 1)$ (since n is even). Since p_n is rightmost, all by agents are oriented to the right. ◀

▶ **Lemma 2.2.** *Given a schedule S_n with $t(S_n) = \pi(\lceil \log n \rceil - 1)$ with all bye agents oriented to the right, we can construct a schedule S_{2n} for $2n$ agents with $t(S_{2n}) = \pi(\lceil \log 2n \rceil - 1)$.*

Proof. Let S'_n be the mirrored schedule of S_n, i.e. the orientation and scans of p_i are swapped with p_{n-i}, and all left orientations change to right and vice versa. First, agents $p_1, \ldots, p_n$ follow the schedule S_n, and agents $p_{n+1}, \ldots, p_{2n}$ follow schedule S'_n simultaneously. If an agent p_i has a bye in S_n, agent p_{2n-i} has a bye in S'_n, and those agents are directed towards each other, and therefore can (and do) scan each other. This removes any byes from the first n rounds, and all unscanned pairs that remain are pairs with one agent in the left half and the other in the right half of $p_1, \ldots, p_{2n}$.

For the remaining rounds, orient the agents in the left half to the right, and in the right half to the left. Consider the graph $G = (P, E)$ on the set P of agents with an edge between any pair of agents that still needs to scan each other. For all $(p, p') \in E$ the agents p and p' are directed towards each other. If n is even, G is a regular bipartite graph of degree n. If n is odd, then each agent had a bye exactly once in S_n, so G is a regular bipartite graph of degree $n-1$. Since a regular bipartite graph has a 1-factorisation, the final rounds can be scheduled. The first rounds take the same time as S_n, and the final rounds need a single rotation of π, resulting in the claimed time. ◀

Strategy 1. For $n = 2$, the two agents directly scan each other. This is S_2. For $n > 2$, construct S_n recursively from $S_{n/2}$ for even n and from S_{n+1} for odd n. See Fig. 1.

▶ **Theorem 2.3.** *For n agents on a line Strategy 1 constructs a schedule S_n with $t(S_n) = \pi(\lceil \log_2 n \rceil - 1)$ in the synchronous (and asynchronous) model.*

Proof. If $n = 2$, $t(S_n) = 0 = \pi(\lceil \log n \rceil - 1)$ time. If $n > 2$, $t(S_n) = \pi(\lceil \log n \rceil - 1)$ follows inductively by Lemmas 2.1 and 2.2. S_n is also a valid schedule in the asynchronous model, since any schedule in the synchronous model is valid in the asynchronous model. ◀

2.2 Regularly-spaced on circle

In the following the agents $p_1, \ldots, p_n$ are regularly spaced on a circle, given in counter-clockwise order. Consequently, for any p_i, the angular distance between two other consecutive agents is identical (i.e. $\frac{\pi}{n}$). We call this time interval a *step*.

2.2.1 Regularly-spaced, synchronous model with $n = 2^k$.

Strategy 2. Define for every agent p_j the value b_j^i as the value of the i-th bit of j, where j has binary representation $b_j^k \cdots b_j^1$ (not including b_j^{k+1}). The strategy works in phases $1, \ldots, k$, where in phase ℓ agent p_j scans all agents $p_{j'}$ with $b_j^\ell \neq b_{j'}^\ell$ that it has not scanned in an earlier phase in time less than 2π.

Phase ℓ has $2^{k-\ell}$ rounds. In the first round of the phase, every agent p_j is oriented towards and scans agent $p_{j'}$, with

$$j' = \begin{cases} j - 2^{\ell-1} \mod n & \text{if } b_j^\ell = 1 \\ j + 2^{\ell-1} \mod n & \text{if } b_j^\ell = 0 \end{cases}.$$

For the next rounds $i = 1, \ldots, 2^{k-\ell} - 1$, the agents with $b_j^\ell = 1$ rotate clockwise and the agents with $b_j^\ell = 0$ rotate counter-clockwise to scan $p_{j'}$, with

$$j' = \begin{cases} j - 2^{\ell-1}(2i+1) \mod n & \text{if } b_j^\ell = 1 \\ j + 2^{\ell-1}(2i+1) \mod n & \text{if } b_j^\ell = 0 \end{cases}.$$

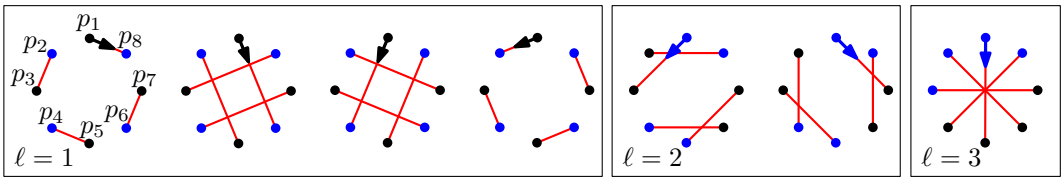


Figure 2 Synchronous schedule for $n = 8$. Agents are blue if the ℓ-th bit is 0, and black otherwise. For $\ell = 1$, p_1 rotates 2 steps clockwise (since $b_1^1 = 1$) in each round. For $\ell = 2$, it rotates 4 steps counter-clockwise (since $b_1^2 = 0$)

Since $j - j' \mod n$ is a multiple of $2^{\ell-1}$ and not a multiple of 2^ℓ, we have $b_j^\ell \neq b_{j'}^\ell$ and so $p_{j'}$ is oriented towards p_j when p_j is oriented towards $p_{j'}$. The angular movement between every two consecutive rounds is equal to $2^\ell \frac{\pi}{n}$, with total movement equal to $\frac{(n-2)\pi}{n}$.

Note that in phase ℓ, every agent p_j scans all other agents $p_{j'}$ where the ℓ-th bit is the smallest bit such that $b_j^\ell \neq b_{j'}^\ell$. So, after k phases, every agent has scanned all other agents.

▶ **Theorem 2.4.** *For $n = 2^k$ agents regularly spaced on a circle Strategy 2 constructs a schedule S with $t(S) \leq 2\pi \log n$ in the synchronous model.*

Proof. For every phase ℓ, the angular movement to complete it is upper bounded by π. The angular movement needed to go from the end state of a step, to the initial state of the next step, is bounded by π as well. Since there are k phases, $t(S) \leq 2\pi k = 2\pi \log n$. ◀

2.2.2 Regularly-spaced, asynchronous model

Strategy 3. The strategy in the asynchronous model works in phases. In each phase, each subset of agents $\{p_i, p_{i+1} \ldots, p_{i+s}\}$ of P that hasn't scanned each other yet is split evenly into a left half $\{p_i, \ldots, p_{i+\lfloor (s-1)/2 \rfloor}\}$ and a right half $\{p_{i+\lfloor (s+1)/2 \rfloor}, \ldots, p_{i+s}\}$. We will scan all pairs between the halves in parallel as follows. See Fig. 3 for an example of a phase.

First, split the left and right halves evenly again into a top and bottom part. W.l.o.g., we assume the subset starts at p_1 and that s is a multiple of 4. Initially, each agent p_j is directed towards p_{s+1-j}. First, scan all pairs between the top left and top right, as follows: for any agent p_j on the top left, rotate towards p_s if p_j and p_s still need to scan each other. Otherwise, rotate towards $p_{3s/4+1}$ until p_j points at $p_{3s/4+1}$. The top right rotates symmetrically. Note that for any $0 \leq j \leq k \leq s/4$, p_{j+1} points to p_{s-k} after $j + k$ steps and p_{s-k} points to p_{j+1} after $k + j$ steps. So, after $s/2 - 1$ steps, all pairs in the top part have been scanned, all agents in the top left point to $p_{3s/4+1}$, and all agents in the top right point to $p_{s/4}$. Symmetrically and in parallel, we can scan all agents in the bottom part after $s/2$ steps, such that all agents in the bottom left point to $p_{3s/4}$, and all agents in the bottom right point to $p_{s/4+1}$.

Next, we rotate each agent in the top left to $p_{3s/4}$ and each agent in the top right to $p_{s/4+1}$, the other agents rotate symmetrically. Then, each agent p_j in the top left waits until it has scanned $p_{3s/4}$, and then rotates towards $p_{s/2-j}$. Note that when p_j and $p_{s/2+1}$ scan each other, p_j is finished for the phase. Additionally, p_j points to $p_{s/2-j}$, the initial orientation for the next phase, after $s/2$ steps. The agents in the bottom half reach the initial orientation for the next phase after $s/2 + n - s$ steps. Now we take each half separately as input for the next phase. After $\lceil \log n \rceil$ phases, all pairs of agents have been scanned.

▶ **Theorem 2.5.** *For n agents regularly spaced on a circle, Strategy 3 constructs a schedule S with $t(S) \leq \pi \frac{n+2}{n}(\lceil \log n \rceil - 1)$ in the asynchronous model.*

Proof. Each phase of Strategy 3, except the last, takes $2(2\lceil\frac{s}{4}\rceil - 1) + n - s \leq n + 2$ steps, and the last 0 steps, so Strategy 3 uses $(n + 2)(\lceil\log n\rceil - 1)$ steps, taking π/n time each. ◀

2.3 General agents in the plane, asynchronous model

Analogously to kd-tree, the strategy subdivides the set of agents iteratively by vertical and horizontal lines passing through median x- and y-coordinates. In each iteration all agents on one side of the line scan all agents on the other side of the line, resulting in $\lceil\log n\rceil$ iterations.

Strategy 4. We describe one iteration i in which we split the agents along a vertical line, that is when i is odd (refer to Figure 4). The case of even i is analogous. Let P' be the subset that is split, and let A and B be the resulting subsets of P' to the left and right, respectively. Each iteration consists of two stages. In the first stage, all the agents in A rotate by $\pi/2$ to point downwards (direction $3\pi/2$), and all agents in B rotate by $\pi/2$ to point upwards (direction $\pi/2$). In the second, main stage, the points in A and B rotate counter-clockwise to the directions $\pi/2$ and $3\pi/2$ respectively. During this phase all pairs of agents (p, q) with $p \in A$ and $q \in B$ scan each other.

▶ **Theorem 2.6.** *For n agents in the plane, Strategy 4 constructs a schedule S with $t(S) = \frac{3\pi}{2}\lceil\log n\rceil - \frac{\pi}{2}$.*

3 Lower bounds

If a subset of size n' of the agents is nearly collinear, the 1D-analysis gives us that at least $\log n'$ orientation changes taking close to π time must occur, for all agents to see each other. We formalise this idea in the following lemma. It makes use of the fact that at least $\lceil\log n'\rceil$ bipartite graphs are needed to cover the complete graph on n' vertices [5].

▶ **Lemma 3.1.** *Let $r \in \mathbb{R}$, and $Q = \{q_1, \ldots, q_{n'}\}$ be a set of agents in the plane. Suppose Q is ordered such that for all $i \in [n']$ and all $1 \leq j < i < k \leq n'$, $\measuredangle q_j, q_i, q_k \geq r$. Then, any schedule that scans all pairs of agents in Q takes at least $r \cdot (\lceil\log n'\rceil - 1)$ time.*

Proof. Suppose some schedule scans all pairs of agents in Q within time T. Partition the time interval $[0, T]$ into the intervals $[(i-1)r, ir)$ for $i = 1, \ldots, \lfloor\frac{T}{r}\rfloor$ and the interval $[r\lfloor\frac{T}{r}\rfloor, T]$. Given an interval, consider the graph $G = (Q, E)$ with agents as vertices and an edge between the pairs of agents scanned in the interval. We will show that this graph is bipartite.

Given an interval, call an agent q_i *positive* (resp. negative) in that interval if there is an q_j with $j > i$ (resp. $j < i$) with $(q_i, q_j) \in E$. Since the length of each interval is strictly smaller than r, an agent cannot be both positive and negative in the same interval. Additionally, if $(q_i, q_j) \in E$, exactly one of those agents is positive and the other is negative. So, each edge has an end in the set of positive agents and another end in the set of negative agents. These sets are disjoint, so the graph G is bipartite.

Since all pairs of agents in Q need to be scanned, the union of the bipartite graphs scanned in each interval must be equal to the complete graph. At least $\lceil\log n'\rceil$ bipartite graphs are needed to cover the complete graph on n' vertices, so there must be at least $\lceil\log n'\rceil$ intervals. Since we partitioned $[0, T]$ in $\lfloor\frac{T}{r}\rfloor + 1$ intervals, we have $\frac{T}{r} \geq \lfloor\frac{T}{r}\rfloor \geq \lceil\log n'\rceil - 1$. ◀

▶ **Theorem 3.2.** *Any schedule for the angular blowing-a-kiss problem on a line takes at least $\pi(\lceil\log n\rceil - 1)$ time. The schedule constructed by Strategy 1 is optimal.*

Proof. $Q = P$ and $r = \pi$ satisfy the conditions of Lemma 3.1. ◀

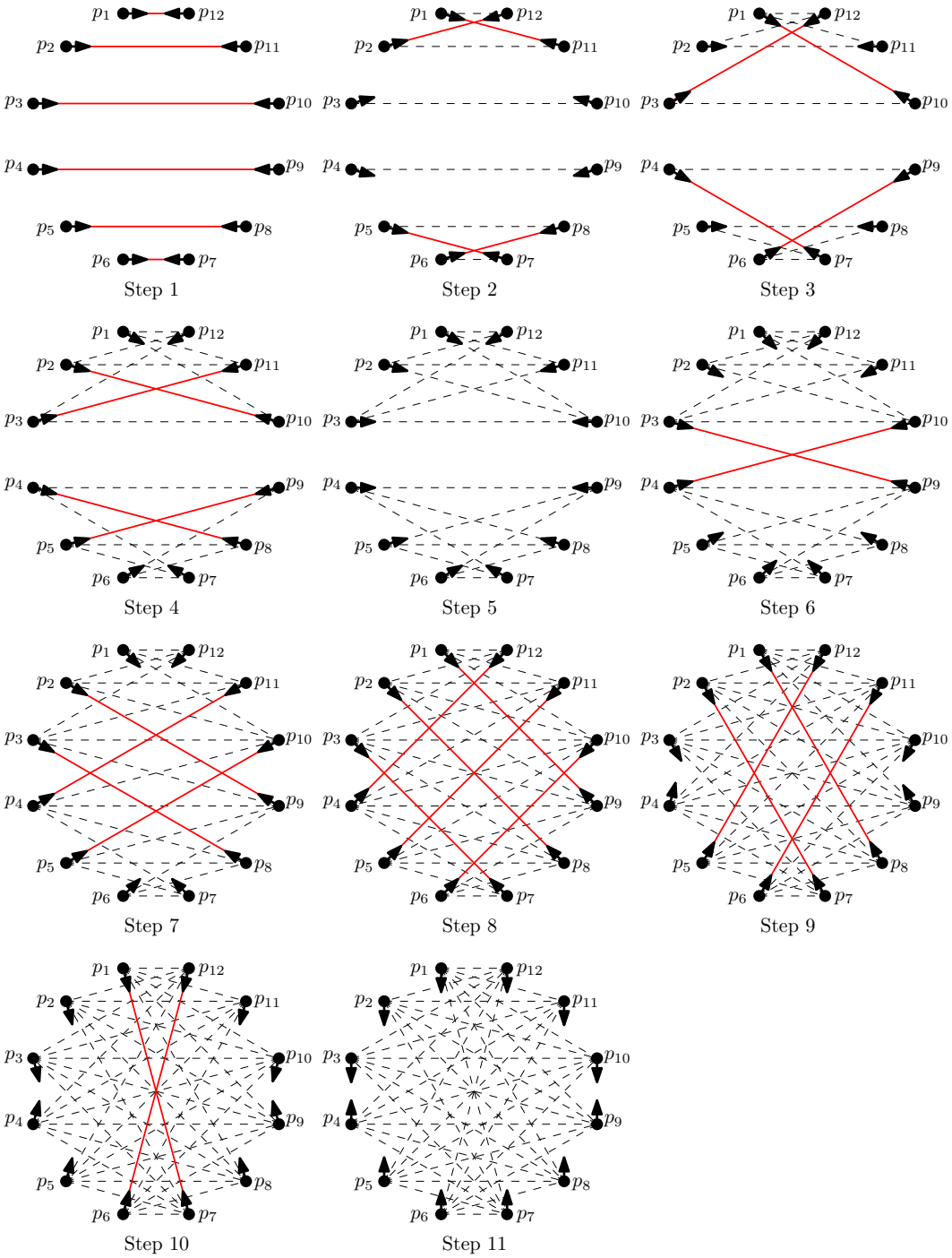


Figure 3 The first phase of Strategy 3 for $n = 12$ has a component of size 12 that is scanned in 10 steps. In step 11, the agents have the correct heading for the next phase. For each step, the edges scanned in that step are colored red, and the edges scanned in an earlier step are dashed.

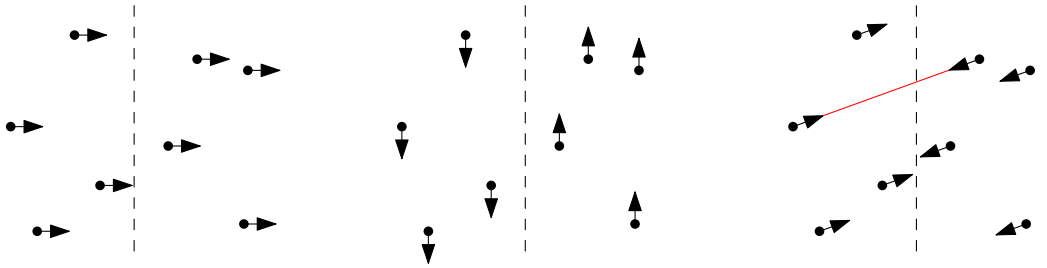


Figure 4 Left: the starting orientation of the agents in iteration i. Middle: the orientation of the agents before the main phase. Right: a snapshot during the main phase when two agents scan each other.

▶ **Theorem 3.3.** *Any schedule for the angular blowing-a-kiss problem with agents regularly-spaced on a circle takes at least* $\pi(1-\frac{1}{\log n})(\log n - \log\log n - 1)$ *time. The schedule constructed by Strategy 3 is asymptotically optimal, i.e. the approximation ratio goes to* 1 *as* $n \to \infty$*. The approximation ratio of the schedule constructed by Strategy 2 goes to* 2 *as* $n \to \infty$*.*

Proof. In this configuration, we can take $Q = \{p_1, \ldots, p_k\}$ and $r = (n-k+1)\frac{\pi}{n}$ to satisfy Lemma 3.1, and get a lower bound of $(n-k+1)\frac{\pi}{n}(\lceil \log k \rceil - 1)$ for any $k \in [n]$. Setting $k = \lceil n/\log n \rceil$, we get a lower bound of $\frac{\pi}{n}(n - \lceil n/\log n \rceil + 1)(\lceil \log \lceil n/\log n \rceil \rceil - 1) \geq \frac{\pi}{n}(n - n/\log n)(\log(n/\log n) - 1) = \pi(1 - \frac{1}{\log n})(\log n - \log\log n - 1) \sim \pi \log n$. Strategy 3 takes at most $\pi\frac{n+2}{n}(\lceil \log n \rceil - 1)$ time, so the ratio goes to 1 as $n \to \infty$. The approximation ratio for Strategy 2 is derived analogously. ◀

Acknowledgements. We thank Daniel Cellucci for proposing the problem to us. We also thank Sándor Fekete and Joe Mitchell for fruitful discussions, and Sándor Fekete, Linda Kleist and Dominik Krupke for sharing their paper [4] with us and helpful comments.

References

1 Michael A Bender, Ritwik Bose, Rezaul Chowdhury, and Samuel McCauley. The kissing problem: how to end a gathering when everyone kisses everyone else goodbye. *Theory of Computing Systems*, 54(4):715–730, 2014.

2 Daniel Cellucci. Personal communication, 2018. Distributed Spacecraft Autonomy Project, NASA Ames Research Center, USA.

3 Sándor Fekete and Dominik Krupke. Beam it up, Scotty: Angular freeze-tag with directional antennas. In *Extended Abstracts of 34th European Workshop on Computational Geometry*, 2018.

4 Sándor P. Fekete, Linda Kleist, and Dominik Krupke. Minimum scan cover with angular transition costs. In *Proc. 36th International Symposium on Computational Geometry (SoCG 2020)*, 2020. To appear.

5 Peter C Fishburn and Peter L Hammer. Bipartite dimensions and bipartite degrees of graphs. *Discrete Mathematics*, 160(1-3):127–148, 1996.

On Generating Polygons: Introducing the Salzburg Database*

Günther Eder[1], Martin Held[1], Steinþór Jasonarson[1], Philipp Mayer[1], and Peter Palfrader[1]

1 **Universität Salzburg, FB Computerwissenschaften, Salzburg, Austria,**
{geder,held,sjas,pmayer,palfrader}@cs.sbg.ac.at

Abstract

The Salzburg Database is a repository of polygonal areas of various classes and sizes, with and without holes. Positive weights are assigned to all edges of all polygons. We introduce this collection and briefly describe the generators that produced its polygons. The source codes for all generators as well as the polygons generated are publicly available.

1 Introduction

An important part of software development is testing the correctness and evaluating the performance of an algorithm implementation. Ideally, one would run the code on data of practical relevance. However, it often is next to impossible to obtain enough practically relevant inputs. Then the second-best choice is to run an algorithm for a reasonably large number of "random" inputs. Subjecting the code to inputs of different characteristics is important since this may help to trigger different execution paths. Similarly, a large range of input sizes is needed to obtain insights in the actual runtime and memory consumption. This allows for comparing different implementations in a meaningful way.

Our goal with the Salzburg Database is to provide a repository of data for such testing purposes. The initial content of the Salzburg Database is purely polygonal, containing simply-connected and multiply-connected polygonal areas in two dimensions.

Every polygon has positive weights assigned to its edges. These weights can be used to test codes that operate on weighted polygonal input, such as for computing weighted straight skeletons. The file format is extensible, so we can also add vertex-weights and other information such as edge or vertex colorings in the future.

We note that this is work in progress. In particular, we are still evaluating the generators and the characteristics of the polygons generated by them. Hence, we expect to see some fine tuning of the generators in the near future. In the sequel we describe the database and its generators.

2 Generators

Generating simple polygons is not a new problem. For convex and x-monotone polygons, Zhu et al. [9] propose a solution to generate them uniformly at random. Tomás and Bajuelos [7] introduce a quadratic-time algorithm to generate random polygons on a grid. Dailey and Whitfield [3] describe a heuristic that takes $\mathcal{O}(n \log n)$ time to compute a simple polygon. They start from a randomly chosen triangle followed by repetitive edge subdivision. Sedhu et al. [5] introduce a different heuristic, which constructs a random polygon starting from the convex hull of a given point set. They randomly select a vertex inside the hull and add

* Work supported by Austrian Science Fund (FWF): Grants ORD 53-VO and P31013-N31.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

it to the polygon while maintaining the simplicity using visibility checks. Later, Sedhu et al. [6] introduce a different approach that uses the convex layers of a given point set and constructs a simple polygon in $\mathcal{O}(n \log n)$ time.

As this brief overview of the literature demonstrates, clearly some research has been devoted to this topic. Here, we present our generators and their actual implementations, some of which, like RPG (Section 2.4) or FPG (Section 2.1), implement algorithms from, or inspired by, literature.

2.1 Triangulation Perturbation

Our implementation FPG is motivated by an approach originally proposed by O'Rourke and Virmani [4]: They start with a regular polygon $\mathcal{P}$ and then translate its vertices while maintaining the polygon's simplicity. A direction and speed are chosen at random and assigned to each vertex of $\mathcal{P}$. Then, the vertices of $\mathcal{P}$ are processed consecutively. A single vertex is moved one "time unit" as long as $\mathcal{P}$ remains simple, otherwise that move is omitted and a new random velocity is chosen for the next round. O'Rourke and Virmani [4] suggest to use several hundred translations per vertex.

As vertices can also move in an outward direction, a domain is defined which has to contain $\mathcal{P}$. We use a large rectangle to limit the outward movement of the vertices.

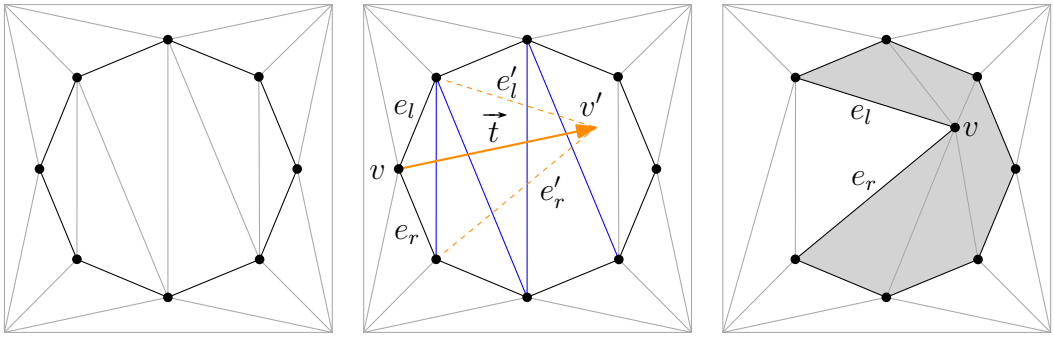


Figure 1 (a) Triangulation of the start polygon and its domain; (b) Translation of vertex v by the vector $\vec{t}$; (c) The polygon after the translation.

Maintaining the simplicity of $\mathcal{P}$ during the vertex translations can be an expensive task if carried out naively. We utilize a triangulation of the interior and the exterior of $\mathcal{P}$ to simplify intersection tests while moving a polygon vertex; cf. Figure 1a. Let v denote a boundary vertex of $\mathcal{P}$ that we want to translate and let e_l and e_r denote its two incident edges. In practice, a randomly chosen translation vector $\vec{t}$ tends to violate the simplicity of $\mathcal{P}$, with high probability, which leads to a bad performance. Therefore, we choose a random direction for $\vec{t}$ first. Then, the length of $\vec{t}$ is generated from a normal distribution using parameters suitable to the local environment around v, in the chosen direction. Experiments show that such an approach for choosing translation vectors produces only few invalid translations.

After translating v by $\vec{t}$, we obtain v' and the edges e_l' and e_r', respectively. Our intersection test involves checking all triangles pierced by e_l' or e_r'. In case all triangle edges intersected by e_l' and e_r' are interior or exterior diagonals, we change v into v' in $\mathcal{P}$. Additionally, we may have to modify the triangulation by checking the triangles intersected by the modified edges as well as the triangles incident at v. If we cross a polygon edge,

we reject $\vec{t}$ as translation vector and restart the process. See an example of this process illustrated in Figures 1b and 1c.

FPG starts from a regular polygon where a triangulation, in- and outside, is trivially obtained. To speed up the generation of large polygons, instead of starting with a large regular polygon, FPG can start with a smaller one, and then "grow" this polygon by repeatedly splitting random edges. The additional vertex introduced by the split is then translated to avoid collinearities.

If we pick edges uniformly at random, we see clusters of many short edges and a few very long edges. This presumably is due to the fact that areas with short edges are more likely to get extra vertices than areas of the same size which contain (fewer) long edges; cf. Figure 2. To avoid this clustering, we instead pick edges randomly weighted by their length.

Furthermore, FPG is capable of generating polygons with holes. Since $\mathcal{P}$ is regular at the beginning, we can trivially place regular holes inside $\mathcal{P}$ as well. The described process works also for this setting, as the intersection tests hinge on the triangulation. In Figure 3 we illustrate the evolution of a polygon computed by FPG, the polygon has 10 vertices, with a triangular hole formed by three additional vertices. The first two images in Figure 4 are the result of FPG using edge-subdivision; the second image depicts a polygon with holes.

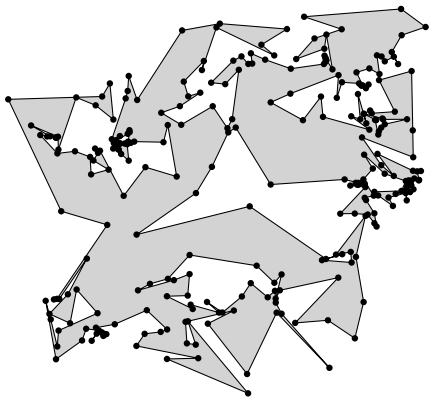


Figure 2 Polygon exhibiting clustering due to the selection of edges uniformly at random in the subdivision step.

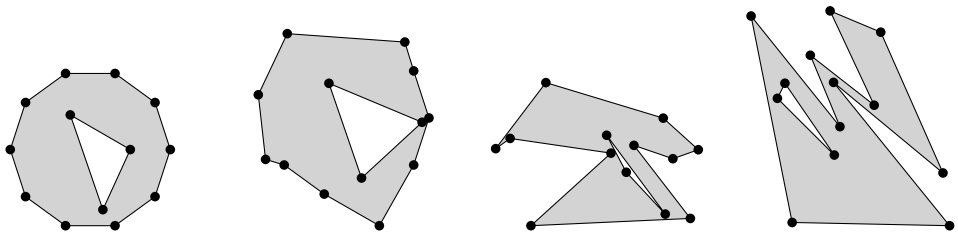


Figure 3 Polygon generated by FPG after 1, 8, 50, and 500 iterations without edge-subdivision.

2.2 Combining Line Sweep and 2-Opt Moves

Our generator SPG constructs a simple polygon $\mathcal{P}$ on a given point set S in the plane. (Such a point set can be generated randomly or specified by a user.) Initially, SPG creates a polygon by choosing a random permutation of the input vertices. This start-polygon contains, with high probability, self-intersections. Therefore, a line sweep is applied to identify intersecting pairs of edges, followed by local modifications which remove these intersections.

To identify pairs of edges that intersect we use the classic Bentley-Ottmann algorithm [2]. We sweep from left to right, thereby maintaining a sorted set of edges that intersect the sweep-line. The input vertices comprise the event points of the line sweep. During the sweep, at vertex v_i, we have to modify the sweep-line status by removing and/or adding the edges

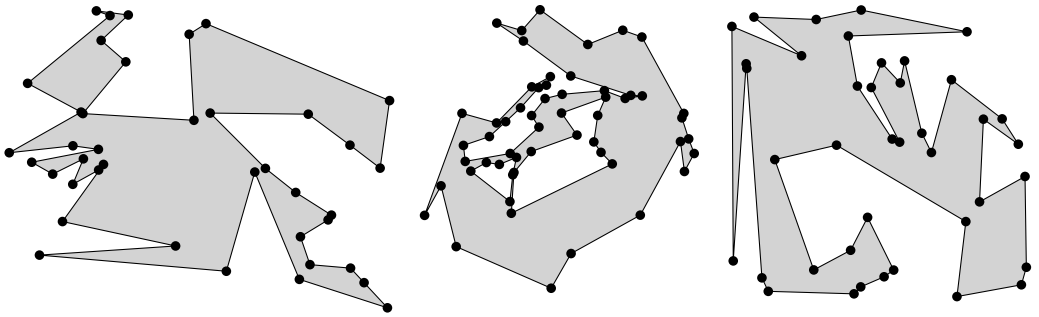


Figure 4 Left-to-right: A polygon and a polygon with holes computed by FPG, and a polygon generated by SPG.

incident at v_i. Additionally, at every event point, we have to verify that any newly added edge is not intersecting its neighbors in the status. In case a pair of edges does intersect, we have to resolve that intersection before we carry on with the sweep.

We resolve intersections by applying so-called *2-opt* moves. A 2-opt move replaces the edges $e_1 = \overline{v_1v_2}$ and $e_2 = \overline{v_3v_4}$ by the edges $e'_1 = \overline{v_1v_3}, e'_2 = \overline{v_2v_4}$. (Note that the polygon boundary becomes disconnected if the 2-opt move connects the wrong vertex pairs.) As we apply 2-opt moves during the line sweep to resolve intersections, we may introduce new intersections. However, a key property of the 2-opt move is that it decreases the length of the polygon (if not all points are collinear). This guarantees that we will eventually arrive at a polygon that is simple if we apply 2-opt moves repeatedly to resolve intersections. A result by van Leeuwen and Schoone [8] tells us that we need at most $\mathcal{O}(n^3)$ 2-opt moves.

We implemented and tested three variants of the line sweep. They differ mainly in how they proceed after finding and resolving an intersection: (a) After a 2-opt move is carried out, we simply continue with the line sweep. After arriving at the right-most vertex we restart the line sweep at the left-most vertex. The sweep is repeated until all intersections are resolved. (b) After a 2-opt move, we test and resolve all intersections at the current sweep-line position, before carrying on. Again, at the right-most vertex we restart until all intersections are resolved. (c) After a 2-opt move, we reverse the sweep direction to deal with possibly new edge intersections. We resume our rightwards sweep at the left-most vertex affected by the 2-opt move. The last image in Figure 4 was generated by SPG on a point set of 40 vertices using sweep-variant (a).

Note that collinear edges need special care because a 2-opt move will not always result in a shortening of the perimeter of the polygon. If intersecting collinear edges are detected, then we remove these edges and sort the respective collinear vertices. Then, we connect the vertices by edges in consecutive order, i.e., form a chain of non-overlapping collinear edges. This guarantees that the perimeter of the polygon decreases also in the case of collinear vertices.

2.3 SRPG

SRPG generates simply-connected and multiply-connected polygonal areas by means of a regular grid that consists of square cells. Given two integer values, a and b, SRPG generates a grid of size a times b. By default SRPG then generates orthogonal polygons on this grid. An additional parameter p, between zero and one, leads to a smaller or larger number of vertices in the produced polygon. SRPG is able to produce octagonal polygons by cutting off

corners with $\pm 45°$ diagonals during the construction. Cutting corners repeatedly, without the diagonal restriction, yields an approximation of a smooth free-form curve. Additionally, SRPG can apply perturbations in order to generate polygons with axes-parallel edges whose vertices do not lie on a grid, or to generate polygons whose edges (in general) are not parallel to the coordinate axes. See Figure 5 for some sample polygons.

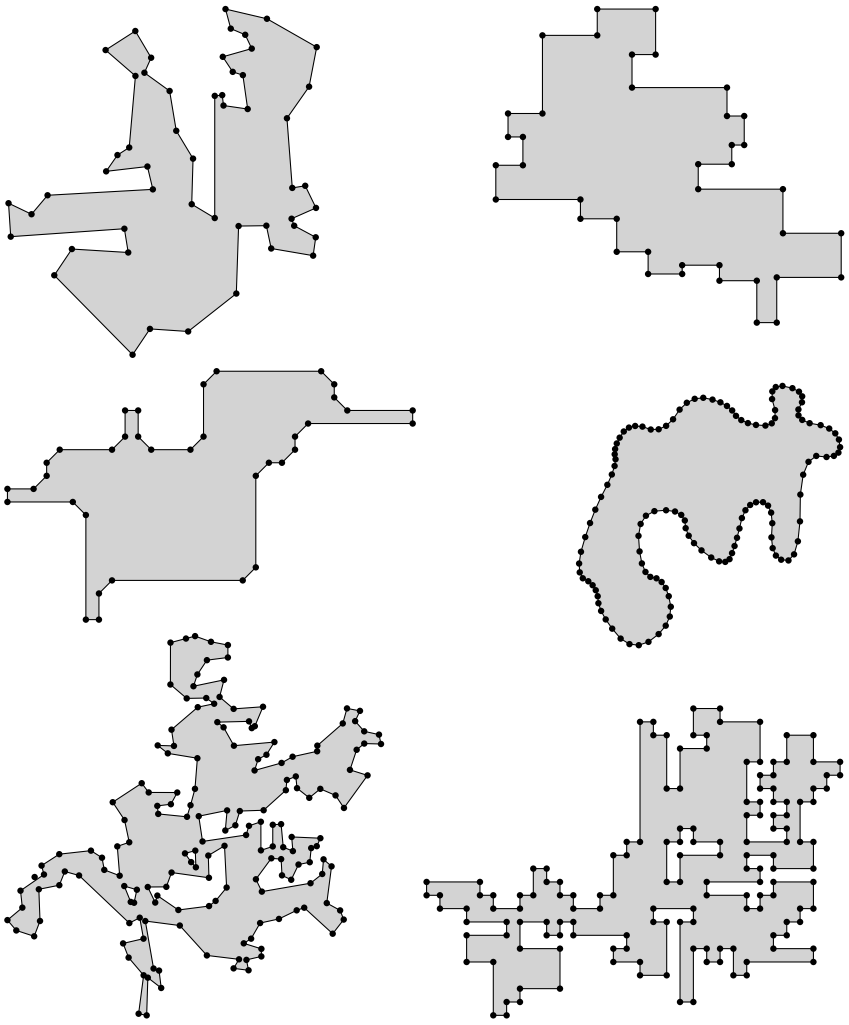


Figure 5 Samples of a random, an orthogonal, an octagonal, and a smoothed polygon generated by SRPG, as well as a random and a grid-aligned orthogonal polygon with holes.

2.4 RPG

Auer and Held [1] first described RPG more than twenty years ago. RPG supports various heuristics to generate "random" polygons for a given set of vertices. In particular, it is able to produce star-shaped polygons uniformly at random. Furthermore, it generates x-monotone polygons uniformly at random, based on the algorithm by Zhu et al. [9]. We have resurrected this code and updated it to compile on modern platforms, thus meeting requests voiced by several colleagues. A recent extension of RPG also supports the generation of polygons with holes. See Figure 6 for examples of some polygons generated by RPG.

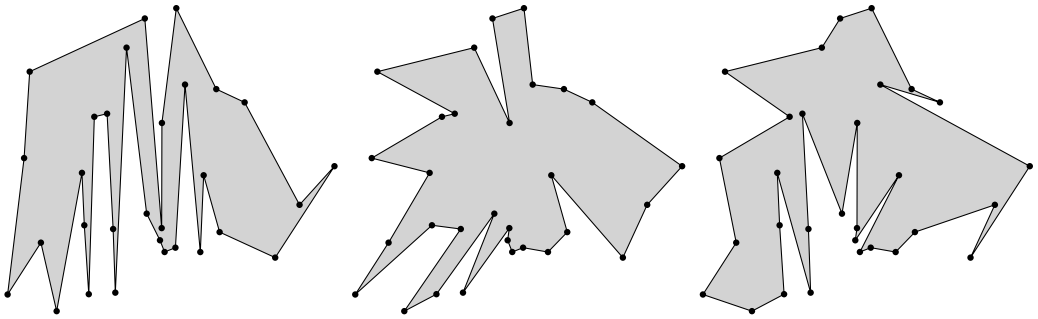


Figure 6 In left-to-right order, an x-monotone, a star-shaped, and a simple polygon computed by RPG on 30 vertices.

2.5 Additional Generators

Our repository also contains codes to produce well-known polygons such as the Koch snowflake (also in a nested variant), the Sierpinski curve, and closed variants of the Hilbert and Lebesgue curves; see Figure 7.

3 Salzburg Database

The Salzburg Database is available at `https://sbgdb.cs.sbg.ac.at/`. Since this is work-in-progress, we expect to add additional data-sets and generators in the near future. The database can be used freely and is provided via direct download or git.

Currently, all our generators are written in C++ or plain C. However, we are not averse to adding code written in other languages such as Python. All source code available on GitHub (`https://github.com/cgalab`) and can be used freely under the GPL(v3) license.

We conclude this survey of the Salzburg Database with a call for participation. If you have "interesting" polygons or data-sets you like to have included then, please, send them to us. You are also welcome to to contact us if you have an interest in a specific class of polygons that is missing.

References

1 T. Auer and M. Held. Heuristics for the Generation of Random Polygons. In *Proceedings of the 8th Canadian Conference on Computational Geometry (CCCG)*, pages 38–44, 1996.

2 J. L. Bentley and T. A. Ottmann. Algorithms for Reporting and Counting Geometric Intersections. *IEEE Transactions on Computers*, 28(9):643–647, 1979.

3 D. Dailey and D. Whitfield. Constructing Random Polygons. In *Proceedings of the 9th ACM SIG-Information Technology Education Conference (SIGITE)*, pages 119–124, 2008.

4 J. O'Rourke and M. Virmani. Generating random polygons. Technical report, Smith College, Northampton, MA 01063, USA, 1991.

5 S. Sadhu, S. Hazarika, K. Jain, S. Basu, and T. De. GRP_CH Heuristic for Generating Random Simple Polygon. In *International Workshop on Combinatorial Algorithms (IWOCA)*, 2012.

6 S. Sadhu, N. Kumar, and B. Kumar. Random Polygon Generation through Convex Layers. *Procedia Technology*, 10:356–364, 2013.

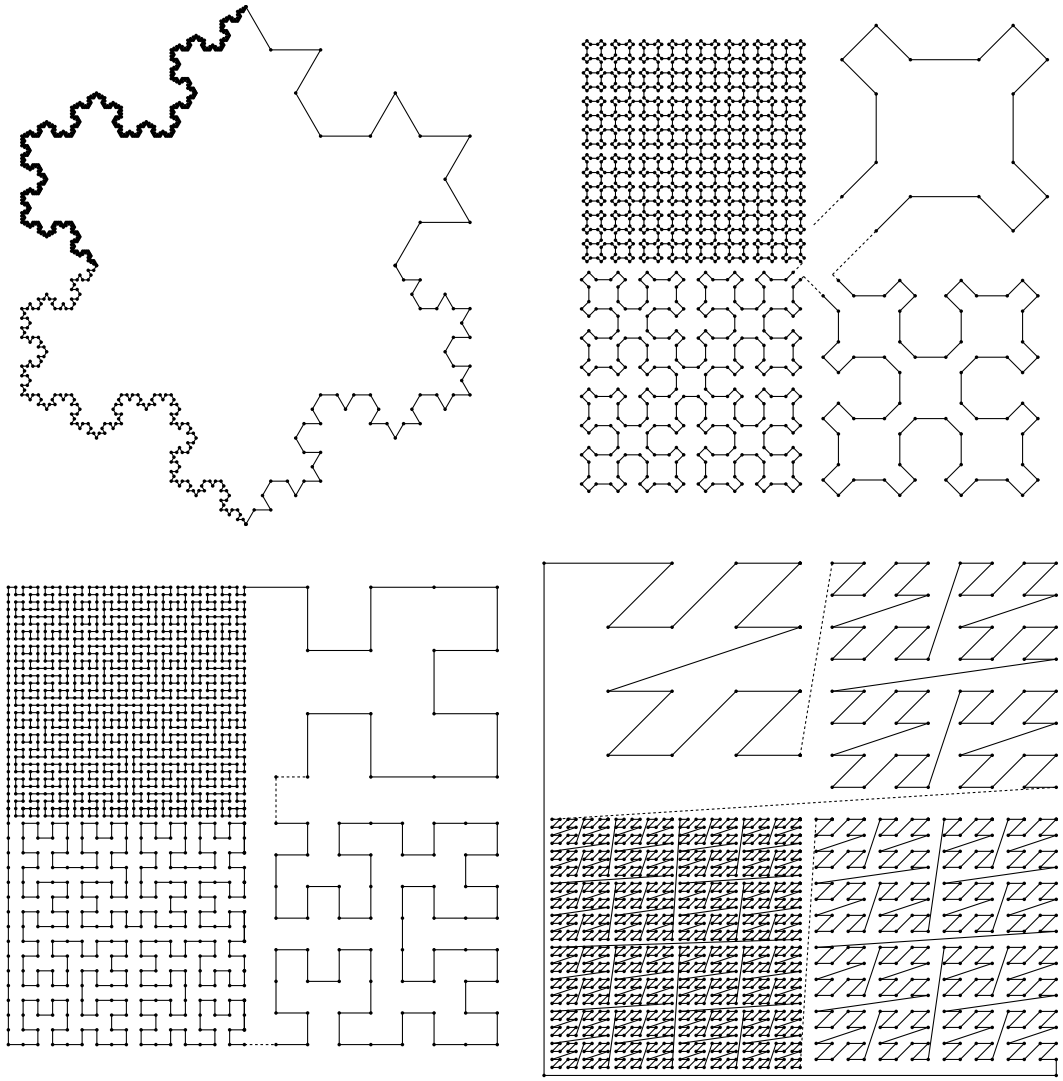


Figure 7 The curves of Koch, Sierpinski, Hilbert, and Lebesgue, in reading order. Each figure is partitioned into four quadrants which portions of the curve at different orders.

7 A. Tomás and A. Bajuelos. Quadratic-Time Linear-Space Algorithms for Generating Orthogonal Polygons with a given Number of Vertices. In *Computational Science and Its Applications (ICCSA)*, pages 117–126, 2004.

8 J. van Leeuwen and A. A. Schoone. Untangling a Travelling Salesman Tour in the Plane. In J. Mühlbacher, editor, *Proc. 7th Conference Graph-theoretic Concepts in Computer Science (WG'81)*, pages 87–98, 1982.

9 C. Zhu, G. Sundaram, J. Snoeyink, and J. Mitchell. Generating Random Polygons with Given Vertices. *Computational Geometry: Theory and Applications*, 6(5):277–290, 1996.

Local Routing in a Tree Metric 1-Spanner

Milutin Brankovic[1], Joachim Gudmundsson[2], and André van Renssen[3]

1 **University of Sydney, Australia**
`milutin.brankovic3@gmail.com`
2 **University of Sydney, Australia**
`joachim.gudmundsson@sydney.edu.au`
3 **University of Sydney, Australia**
`andre.vanrenssen@sydney.edu.au`

Abstract

Solomon and Elkin [5] constructed a shortcutting scheme for weighted trees which results in a 1-spanner for the tree metric induced by the input tree. The spanner has logarithmic lightness, logarithmic diameter, a linear number of edges and bounded degree (provided the input tree has bounded degree). This spanner has been applied in a series of papers devoted to designing bounded degree, low-diameter, low-weight $(1+\epsilon)$-spanners in Euclidean and doubling metrics. In this paper, we present a simple local routing algorithm for this tree metric spanner. The algorithm has a routing ratio of 1, is guaranteed to terminate after $O(\log n)$ hops and requires $O(\Delta \log n)$ bits of storage per vertex where Δ is the maximum degree of the tree on which the spanner is constructed.

1 Introduction

Let T be a weighted tree. The tree metric induced by T, denoted M_T, is the complete graph on the vertices of T where the weight of each edge (u, v) is the weight of the path connecting u and v in T. For $t \geq 1$, a t-spanner for a set of points V with a distance function d is a subgraph H of the complete graph on V such that every pair of distinct points $u, v \in V$ is connected by a path in H of total weight at most $t \cdot d(u, v)$. We refer to such paths as t-spanner paths. A t-spanner has diameter Λ if every pair of points is connected by a t-spanner path consisting of at most Λ edges. Typically, t-spanners are designed to be sparse, often with a linear number of edges. The lightness of a graph is the ratio of its weight to the weight of its minimum spanning tree. Solomon and Elkin [5] define a 1-spanner for tree metrics. Given an n vertex weighted tree of maximum degree Δ, the 1-spanner has $O(n)$ edges, $O(\log n)$ diameter, $O(\log n)$ lightness and maximum degree bounded by $\Delta + k$ (k is an adjustable parameter considered to be a constant for our purposes). While being an interesting construction in its own right, this tree metric 1-spanner has been used in a series of papers as a tool for reducing the diameter of various Euclidean and doubling metric spanner constructions [1, 2, 4, 5].

A local routing algorithm for a weighted graph G is a method by which a message can be sent from any vertex in G to a given destination vertex. The successor to each vertex u on the path traversed by the routing algorithm must be determined using only knowledge of the destination vertex, the neighbourhood of u and possibly some extra information stored at u. In some settings, the routing algorithm may modify the message header to provide extra information for future routing decisions. However, the routing algorithm presented in this paper does not require a modifiable header. Given two vertices u, v, we define $d_{route}(u, v)$ to be the length of the routing path traversed when routing from u to v. The routing ratio of the routing algorithm is defined to be $\max_{u,v \in V(G)} \frac{d_{route}(u,v)}{d_G(u,v)}$ where $d_G(u, v)$ denotes the

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

length of the shortest path from u to v in G. We define the diameter of a local routing algorithm to be an upper bound on the number of edges traversed when routing between any two vertices. In this paper, we present a simple local routing algorithm for the tree metric spanner from [5] with routing ratio 1 and diameter $O(\log n)$.

2 Shortcutting Scheme

In this section we describe the tree metric 1-spanner for which our routing algorithm is defined. The spanner is due to Solomon and Elkin [5]. For brevity, we only give a high level overview of the construction. Full details can be found in [5] and will also appear in the full version of this paper.

Given a rooted weighted tree T on n vertices, an integer k, $k \geq 1$, is chosen. While the construction is defined for any k in the range $1 \leq k \leq n-2$, we choose $k = O(1)$. Next, at most $k+1$ vertices are selected from $V(T)$. Let us denote this set by C_T. The method by which these vertices are selected is deterministic. Denote by $T \setminus C_T$ the forest resulting from the removal of the vertices C_T (and their incident edges) from T. Next, the procedure adds the edges of the complete graph on C_T to the spanner. If the forest $T \setminus C_T$ is non-empty, the procedure is recursively applied to each tree in $T \setminus C_T$. We define *canonical subtrees* to be the trees on which the recursive procedure is called during the course of the construction of the spanner. For a canonical subtree T', we say $C_{T'}$ is a set of cut vertices. Note that for every vertex v, there is a canonical subtree T^v for which $v \in C_{T^v}$. We say T^v is the canonical subtree of v.

We note that the spanner defined by Solomon and Elkin [5] actually differs slightly from what is presented here in that rather than including the edges of the complete graph on sets of cut vertices, a certain spanner with $O(k)$ edges and $O(\alpha(k))$ diameter (α denotes the inverse Ackermann function) is used instead. However, the spanner resulting from the use of the complete graph has higher weight and degree only by a factor of k while being far easier to work with for the purpose of routing.

Let G denote the graph resulting from the construction described above. The following is established by Solomon and Elkin [5] for the version of the spanner defined in the original paper. It is easy to see the properties also hold for the version of the spanner described here.

▶ **Theorem 1.** *The graph G satisfies the following:*

1. *G is a $O(\log n)$ diameter 1-spanner for M_T.*
2. *$wt(G) = O(\log n) \cdot wt(T)$.*
3. *For any canonical subtree T', each tree in $T' \setminus C_{T'}$ has at most $2 \cdot |T'|/k$ vertices.*
4. *The maximum degree of G is at most $O(1) + \Delta$ where Δ is the maximum degree of T.*

Note that property 3 implies that the recursion depth of the spanner construction algorithm is $O(\log n)$.

3 Routing Algorithm

In this section, we describe a local routing algorithm for the spanner described above. The routing algorithm presented in this section requires that the vertices of the spanner store certain information which we specify below. We make use of the labelling scheme of Santoro and Khatib [3].

Let $rank(v)$ denote the rank of v in a post-order traversal of T. We define

$$L(v) := \min\{rank(w) : w \text{ is a descendant of } v\}.$$

Let N_v be the set of neighbours of v in G. Each vertex v of G stores the following information:

1. $rank(v)$ and $L(v)$.
2. The depth of v, i.e, the hop distance of v from the root of T.
3. $rank(w)$ and $L(w)$ for each $w \in N_v$.
4. The depth of w for each $w \in N_v$.

▶ **Lemma 2.** *In the labelling scheme outlined above, each vertex of G stores $O((\Delta+k)\log n)$ bits of information.*

We note that this labelling scheme enables us to determine if a given vertex is an ancestor or descendant of another. Indeed, a vertex u is an ancestor of a vertex v if and only if $L(u) \leq rank(v) \leq rank(u)$ and u is a descendant of v if and only if $L(v) \leq rank(u) \leq rank(v)$.

This test is also used in the tree routing algorithm of Santoro and Khatib [3]. In our routing algorithm, we use this test to determine the neighbours of the current vertex in the tree spanner which are actually on the original path to the destination. We must limit our routing steps to these vertices to ensure a routing ratio of 1. When then use additional criteria to make the best choice from the feasible routing steps to ensure the diameter of the routing algorithm is $O(\log n)$.

Given a current vertex u and a destination vertex v, the algorithm executes the routing steps of one of the cases defined below. We assume that at each stage of the algorithm, the integers $rank(v)$ and $L(v)$ are known.

For convenience of analysis, in each case we specify two routing steps. For ease of exposition, we consider a vertex u to be both a descendant and ancestor of itself.

Case 0: u and v are joined by an edge. Route to v.

Case 1: u is an ancestor of v in T. Let X be the set of vertices in C_{T^u} which are ancestors of v. Let x be the deepest element of X. Route first to x and then to the child of x which is an ancestor of v.

Case 2: u is a descendant of v in T. Let X be the set of vertices in C_{T^u} which are descendants of v and ancestors of u. Let x be the highest vertex in X. Route first to x and then to its parent.

Case 3: u is not an ancestor or descendant of v. Let X be the set of vertices in C_{T^u} which are ancestors of v and not ancestors of u. If $X \neq \emptyset$, we define x to be the deepest vertex in X and define x' to be the child of x which is an ancestor of v. Let Y be the set of vertices in C_{T^u} which are ancestors of u but not ancestors of v. We define y to be the highest vertex in Y.

Case 3 a): X is empty. Route first to y and then to the parent of y.

Case 3 b): X is non-empty. Route first to x and then to x'.

▶ **Theorem 3.** *Let u and v be vertices of G. Let $\delta_T(u,v)$ denote the length of the path from u to v in T. The routing algorithm described above is guaranteed to terminate after a finite number of steps and the length of the path traversed is exactly $\delta_T(u,v)$.*

Next we show that routing paths consist of $O(\log n)$ edges. In order to do this, we must define *canonical sequences*. First, we assign an integer sequence $S_{T'}$ to each canonical subtree T'. These sequences are defined inductively as follows. The original tree T is assigned the empty sequence. Let T' be a canonical subtree and suppose T' has already been assigned the sequence $S_{T'}$. Each canonical subtree $T_j \in T' \setminus C_{T'} = \{T_1, ..., T_p\}$ is assigned the sequence obtained by appending j to $S_{T'}$. Given a vertex v of G, we define its canonical sequence to be $S_v = S_{T^v}$. Note that if for two vertices u and v, S_u is a prefix of S_v, then T^u contains T^v. Note also that $S_u = S_v$ if and only if $T^u = T^v$ and so u and v are joined by an edge if $S_u = S_v$, by definition of the spanner.

▶ **Lemma 4.** *Let u and v be vertices of T such that u is either an ancestor or a descendant of v. Let u' be the vertex reached after executing the routing steps of either Case 1 or Case 2 when routing from u to v. Then the following statements hold:*

1. *If S_u is a prefix of S_v, then $|S_{u'}| > |S_u|$. Moreover, either $S_{u'} = S_v$ or S'_u is a prefix of S_v.*
2. *If S_v is a prefix of S_u, then $|S_{u'}| < |S_u|$. Moreover, either $S_{u'} = S_v$ or S_v is a prefix of $S_{u'}$.*
3. *Suppose S_u and S_v share a common prefix S of length $m < \min\{|S_u|, |S_v|\}$. Then $|S_{u'}| < |S_u|$. Moreover, either $S_{u'} = S$ or S is a prefix of $S_{u'}$*

Since the spanner construction algorithm has logarithmic depth, we see that the length of a canonical sequence is at most $O(\log n)$. Using this observation and Lemma 4, it is not difficult to show the following.

▶ **Lemma 5.** *Suppose u and v in G are such that u is an ancestor or descendant of v in T. Then, when routing from u to v, the routing algorithm reaches v after traversing $O(\log n)$ edges.*

Consider the case where u is neither an ancestor nor a descendant of v. The following lemma shows that in this case, the algorithm either routes to a vertex on the path from $lca(u,v)$ to v, where $lca(u,v)$ is the lowest common ancestor of u and v, or it executes the routing steps that would be executed if the algorithm were routing from u to $lca(u,v)$.

▶ **Lemma 6.** *Let u and v be vertices of G such that $lca(u,v) \notin \{u,v\}$. Suppose that the set X as defined in Case 3 is empty so that the algorithm executes the routing steps of Case 3 a) when routing from u to v. Then the algorithm performs the routing steps which would be performed if the destination was $lca(u,v)$ rather than v.*

Lemmas 5 and 6 imply the following:

▶ **Theorem 7.** *Let u and v be vertices in G. The routing algorithm reaches v when routing from u after traversing at most $O(\log n)$ edges.*

4 Concluding Remarks

We have demonstrated that a slightly modified version of the tree metric 1-spanner of Solomon and Elkin [5] supports a $O(\log n)$ diameter local routing algorithm with routing

ratio 1. The tree metric spanner has been used in the literature as a tool to reduce the diameter of various spanner constructions while either preserving or incurring minimal penalties in other desirable properties of the spanner such as number of edges, degree, diameter and weight. We leave it as future work to use this local routing algorithm as a basis for local routing algorithms on some of the aforementioned Euclidean and doubling metric spanners.

References

1 S. Arya, G. Das, D. M. Mount, J. S. Salowe, and M. Smid. Euclidean spanners: Short, thin, and lanky. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, STOC '95, pages 489–498, 1995.

2 T. Chan, M. Li, L. Ning, and S. Solomon. New doubling spanners: Better and simpler. *SIAM Journal on Computing*, 44(1):37–53, 2015.

3 N. Santoro and R. Khatib. Labelling and implicit routing in networks. *The Computer Journal*, 28(1):5–8, 1985.

4 S. Solomon. From hierarchical partitions to hierarchical covers: Optimal fault-tolerant spanners for doubling metrics. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, STOC '14, page 363–372, 2014.

5 S. Solomon and M. Elkin. Balancing degree, diameter, and weight in Euclidean spanners. *SIAM Journal on Discrete Mathematics*, 28(3):1173–1198, 2014.

A better approximation for longest noncrossing spanning trees*

Sergio Cabello[1], Aruni Choudhary[†2], Michael Hoffmann[‡3], Katharina Klost[2], Meghana M. Reddy[‡3], Wolfgang Mulzer[2], Felix Schröder[4], and Josef Tkadlec[5]

1 **University of Ljubljana, Slovenia**
sergio.cabello@fmf.uni-lj.si
2 **Institut für Informatik, Freie Universität Berlin, Germany**
{arunich,kathklost,mulzer}@inf.fu-berlin.de
3 **Department of Computer Science, ETH Zürich, Switzerland**
{hoffmann,meghana.mreddy}@inf.ethz.ch
4 **Technische Universität Berlin, Germany**
fschroed@math.tu-berlin.de
5 **Institute of Science and Technology, Austria**
jtkadlec@ist.ac.at

Abstract

Let P be a finite set of points in the plane. For any spanning tree T on P, we denote by $|T|$ the Euclidean length of T. Let T_{OPT} be a noncrossing spanning tree of maximum length for P. We show how to construct a noncrossing spanning tree T_{ALG} with $|T_{\text{ALG}}| \geq \delta \cdot |T_{\text{OPT}}|$ with $\delta = 0.512$. We also show how to improve this bound when the points lie in a thin rectangle.

1 Introduction

In this paper we address the problem of finding a longest noncrossing spanning tree. The closely related problems of finding both a shortest (noncrossing) and a longest (possibly crossing) spanning tree are computationally easy. The minimization version is simply the classical minimum spanning tree problem, and the noncrossing property follows from the triangle inequality. Similarly, the longest spanning tree can be computed in a greedy fashion. In contrast, finding the longest noncrossing spanning tree is conjectured to be NP-hard [1].

As obtaining an efficient exact algorithm seems to be difficult, we focus on polynomial-time approximation algorithms for the longest noncrossing spanning tree. One of the first results is due to Alon et al. [1] who gave an 0.5-approximation. Dumitrescu and Tóth [3] refined this algorithm and achieved an approximation factor of 0.502. In their analysis, they compare the output of their algorithm to a longest, possibly crossing, spanning tree. With a modification of this algorithm, Biniaz et al. improved this factor slightly to 0.503 [2]. They also compare their result to the longest crossing spanning tree. While such a tree provides a safe upper bound, it is not a valid solution for the problem and may be up to $\pi/2 > 1.5$ times longer than a longest noncrossing spanning tree [1].

* This research was started at the 3rd DACH Workshop on Arrangements and Drawings, August 19–23, 2019, in Wergenstein (GR), Switzerland, and continued at the 16th European Research Week on Geometric Graphs, November 18–22, 2019, in Strobl, Austria. We thank all the participants of the workshop for valuable discussions and for creating a conducive research atmosphere.
† Supported in part by ERC StG 757609.
‡ Supported by the Swiss National Science Foundation within the collaborative DACH project *Arrangements and Drawings* as SNSF Project 200021E-171681.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

In this paper, we aim to design a better approximation algorithm by making use of the noncrossing property. In this way, we obtain a significant improvement on the approximation factor to 0.512. Our algorithm uses similar ideas and constructions as the previous algorithms.

Moreover, we can show an even better approximation for "thin" point sets. In particular, we show that when the point set lies in a thin rectangular strip, then there is always a noncrossing spanning tree of length at least 2/3 the length of the longest (possibly crossing) spanning tree, and that this bound is tight.

2 Preliminaries

Let $P \subset \mathbb{R}^2$ be the given point set. Without loss of generality we assume that $\operatorname{diam}(P) = 1$. Similar to the existing algorithms [1, 2, 3], we make extensive use of stars. The *star* S_p rooted at some point $p \in P$ is the tree that connects p to all other points of P (see Figure 1).

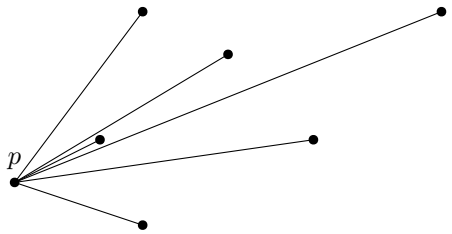


Figure 1 A star S_p.

The following slight generalization of Lemma 3 in Dumitrescu and Tóth [3] will be very useful throughout the paper.

▶ **Lemma 2.1.** *Let $p, q \in P$. Then $\max\{|S_p|, |S_q|\} \geq \frac{n}{2}\|pq\|$.*

Proof. First we note that $\max\{|S_p|, |S_q|\} \geq \frac{1}{2}(|S_p| + |S_q|)$. The triangle inequality yields:

$$|S_p| + |S_q| = \sum_{r \in P} \|pr\| + \|rq\| \geq \sum_{r \in P} \|pq\| = n \cdot \|pq\|. \qquad \blacktriangleleft$$

▶ **Observation 2.2.** *Let ab be a longest edge of T_{OPT}. As $\|ab\| \leq 1$ by assumption, we have*

$$|T_{\text{OPT}}| \leq \|ab\|(n-1) < \|ab\|n \leq n.$$

3 The 0.512-approximation

We show how to compute a spanning tree T_{ALG} with $|T_{\text{ALG}}| \geq \delta \cdot |T_{\text{OPT}}| = 0.512 \cdot |T_{\text{OPT}}|$. Our approach is the following: we guess a longest edge ab of T_{OPT}. If $\|ab\| < d := \frac{1}{2\delta}$ then it is straightforward to give a good approximation, as shown below in Lemma 3.1. Otherwise, we describe six different noncrossing spanning trees for the set P and show that at least one of them gives an approximation ratio of at least δ.

We use the noncrossing property of the optimal tree T_{OPT} in Lemma 3.2, which also is the bottleneck case in our construction.

From now on, we assume that ab is a longest edge in T_{OPT} and that p, q is a pair of vertices that realizes the diameter, that is, $\|pq\| = 1 \geq \|ab\|$.

▶ **Lemma 3.1.** *Let T_{DIAM} the longer of S_p and S_q. If $\|ab\| < d$, then $|T_{\text{DIAM}}| \geq \delta \cdot |T_{\text{OPT}}|$.*

Proof. From Lemma 2.1 it follows that $\max\{|S_p|, |S_q|\} \geq \frac{n}{2}$. As we observed above, we have $|T_{\text{OPT}}| \leq \|ab\|n < dn$. Thus, we get an approximation ratio of

$$\frac{|T_{\text{DIAM}}|}{|T_{\text{OPT}}|} \geq \frac{n/2}{dn} = \frac{1}{2d} = \delta.$$ ◀

Now we only consider the case where $\|ab\| \geq d$. Additionally for ease of presentation, we will assume that $a = (0, 0)$ and $b = (\|ab\|, 0)$ without loss of generality.

First, we define $F = D(a, 1) \cap D(b, 1)$ to be the region with distance at most 1 from a and b. Since the diameter of the point set is 1, we can be sure that $P \subset F$. Let $\hat{\alpha}$ be a constant to be determined later. Set $\gamma = \frac{2\cdot\delta-1+\hat{\alpha}}{\hat{\alpha}}$ and let $E = \{x \in \mathbb{R}^2 \mid \|ax\| + \|xb\| \leq \gamma\}$.

Lastly, we subdivide $E \cap F$ into three vertical strips. We fix a parameter $\omega = 0.1$. Let ℓ_1, ℓ_2 be the vertical lines at $\omega\|ab\|$ and $(1-\omega)\|ab\|$, respectively. Let L be the part of $E \cap F$ to the left of ℓ_1, let M be the part between ℓ_1 and ℓ_2, and let R be the part to the right of ℓ_2. See Figure 2 for a schematic.

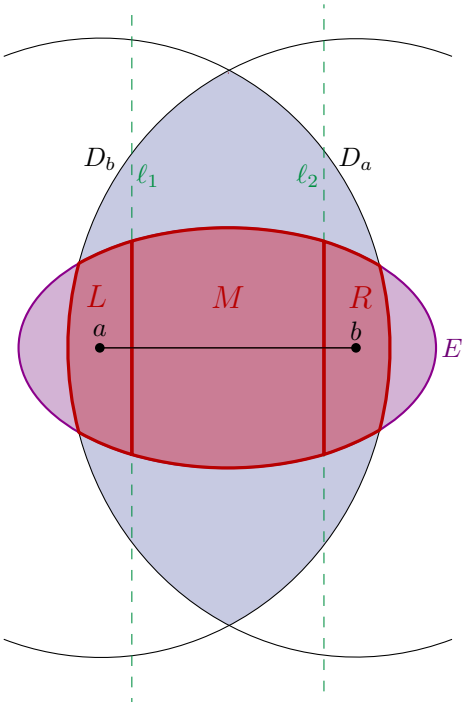


Figure 2 Subdivision of the plane into regions with respect to a longest edge ab of T_{OPT}.

We denote by α the fraction of points in $F \setminus E$, and by β_L, β_M and β_R the fraction of points in L, M and R respectively. Note that $\alpha + \beta_L + \beta_M + \beta_R = 1$. Now we are equipped to consider the next two cases:

▶ **Lemma 3.2.** *Assume*

$$\beta_M \geq \hat{\beta} = \frac{\delta - 0.5}{\delta \cdot \left(1 - \sqrt{1 - d^2(\omega - \omega^2)}\right)}$$

and recall that T_{DIAM} is the larger of the stars at the diameter. Then $|T_{\text{DIAM}}| \geq \delta \cdot |T_{\text{OPT}}|$.

Proof. The main insight in this case is that we can find a tighter bound on T_{OPT} by exploiting that ab is an edge of T_{OPT} and so no other edge of T_{OPT} can cross ab. Let $\overline{M}$ be the region of

F between ℓ_1 and ℓ_2 and above ab. Refer to Figure 3 for illustration. We will argue that every edge with an endpoint in M has length at most $\operatorname{diam}(\overline{M})$.

Let $c_1 = \ell_1 \cap \partial(E) \cap \overline{M}$ and $c_2 = \ell_1 \cap ab$. Disregarding symmetry, it follows from convexity that the longest possible edge starting in M has either c_1 or c_2 as an endpoint. If the endpoint is c_1, then the edge may reach below the line through ab. A maximum length edge starting from c_1 ends at the intersection z of the line through c_1 and b with the boundary of F. If the endpoint is c_2, the length of this edge is $\operatorname{diam}(\overline{M})$. Both cases are shown in Figure 3.

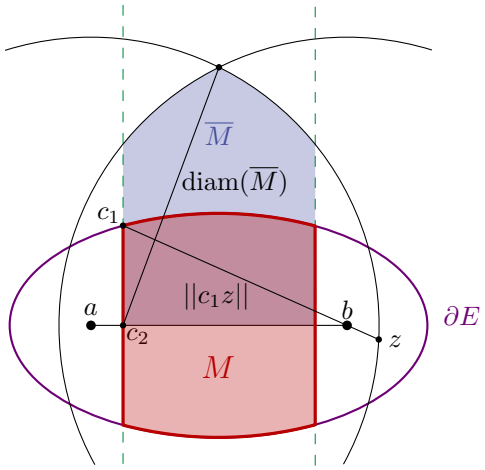


Figure 3 The starting points c_1 and c_2 of longest edges in T_{OPT}.

Now we consider how these lengths change for $d \le \|ab\| \le 1$. By basic trigonometry, we can give expressions for $\operatorname{diam}(\overline{M})$ and $\|c_1z\|$ that only depend on $\|ab\|$:

$$\operatorname{diam}(\overline{M}) = \sqrt{1 - \|ab\|^2(\omega - \omega^2)}$$

$$\|c_1b\| = \sqrt{((1-\omega)\|ab\|)^2 + \left(\frac{\sqrt{\gamma^2 - \|ab\|^2} \cdot \sqrt{(\gamma/2)^2 - (\|ab\|/2 - \omega\|ab\|)^2}}{\gamma}\right)^2}$$

$$\|c_1z\| \le \|c_1b\| + \frac{\|c_1b\|(1 - \|ab\|)}{(1-\omega)\|ab\|}$$

The last bound is tight for $\|ab\| = 1$.

When considering $\operatorname{diam}(\overline{M})$ and $\|c_1z\|$ as functions of $\|ab\|$, by considering the plots (Figure 4) it follows that

$$\begin{aligned} \sqrt{1 - d^2(\omega - \omega^2)} = \operatorname{diam}(\overline{M})_d \ge \quad & \operatorname{diam}(\overline{M}) \quad \ge \operatorname{diam}(\overline{M})_1 \text{ and} \\ \|c_1z\|_d \ge \quad & \|c_1z\| \quad \ge \|c_1z\|_1 , \end{aligned} \tag{1}$$

where the subscripted versions denote the values at $\|ab\| = d$ and $\|ab\| = 1$, respectively.

With the chosen constants we get $\operatorname{diam}(\overline{M})_1 \ge \|c_1z\|_d$ (again refer to Figure 4). Thus, $\operatorname{diam}(\overline{M})$ is a valid upper bound for the length of the edge starting in M.

Using (1) and the definition of $\hat{\beta}$ we can bound the size of T_{OPT} and the approximation ratio:

$$\begin{aligned} |T_{\text{OPT}}| &\le n \cdot (\beta_M \cdot \operatorname{diam}(\overline{M}) + (1 - \beta_M)) \le n \cdot (\beta_M \cdot \operatorname{diam}(\overline{M})_d + (1 - \beta_M)) \\ &= n \cdot (1 - \beta_M \cdot (1 - \operatorname{diam}(\overline{M})_d)) \end{aligned}$$

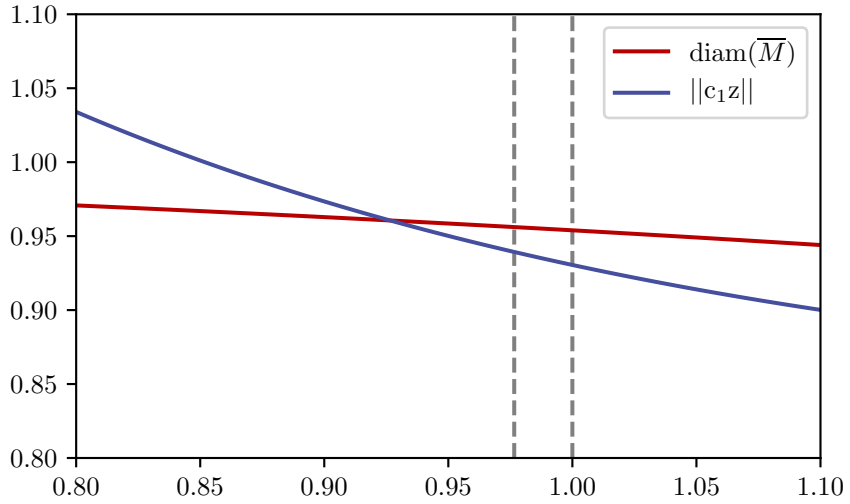


Figure 4 Plot of $\operatorname{diam}(\overline{M})$ and $\|c_1 z\|$ over the length of ab. The vertical lines are at d and 1.

$$\frac{|T_{\text{DIAM}}|}{|T_{\text{OPT}}|} \geq \frac{0.5n}{(1-\beta_M \cdot (1-\operatorname{diam}(\overline{M})_d))n} \geq \frac{0.5}{1-\hat{\beta}\cdot(1-\operatorname{diam}(\overline{M})_d)} = \delta. \qquad \blacktriangleleft$$

In the next case, we assume that $\alpha \geq \hat{\alpha}$ and also show that there is a good star.

▶ **Lemma 3.3.** *If*

$$\alpha \geq \hat{\alpha} = 1 - \frac{2\delta + \hat{\beta}(1-\omega)}{2-3\omega},$$

then $\max\{|S_a|, |S_b|\} \geq \delta \cdot |T_{\text{OPT}}|$.

Proof. As before we bound $\max\{|S_a|, |S_b|\} \geq \frac{1}{2}(|S_a| + |S_b|)$. This time we get:

$$\begin{aligned} |S_a| + |S_b| &\geq n(\alpha \cdot \gamma + (1-\alpha)\|ab\|) \\ &= n(\|ab\| + \alpha(\gamma - \|ab\|)) \\ &\geq n \cdot (\|ab\| + \alpha(\gamma - 1)). \end{aligned}$$

With Observation 2.2 ($|T_{\text{OPT}}| \leq \|ab\|n$) we get

$$\frac{\max\{|S_a|, |S_b|\}}{|T_{\text{OPT}}|} \geq \frac{n(\|ab\| + \alpha(\gamma-1))}{2\|ab\|n} \geq \frac{1}{2} + \frac{\alpha}{2}(\gamma-1) \geq \frac{1}{2} + \frac{\hat{\alpha}}{2}(\gamma-1) = \delta. \qquad \blacktriangleleft$$

Last but not least we consider the case where α and β_M are both small. Intuitively, this means that almost all points are located left or right in E.

▶ **Lemma 3.4.** *If $\alpha < \hat{\alpha}$ and $\beta_M < \hat{\beta}$, then there is a tree which gives a δ-approximation.*

Proof. In this case we do not use a star but trees B_{ab}, B_{ba} of diameter at most five. We will describe the structure B_{ab} with regard to a. The structure B_{ba} with regard to b is symmetric. See Figure 5 for an example of the construction.

We start by connecting all points in R to a (blue edges). This gives a star with length at least $\beta_R(1-\omega)\|ab\|$. The edges of this star subdivide L into wedges. We define the upper

wedge to be the region above both the highest edge and the x-axis. The lowest wedge is defined accordingly. For each such wedge W (except the last) we take the lower point of R defining W and connect it to all points in $L \cap W$. The lowest point in R also connects to the points in the lowest wedge of L (green edges). Each of these new edges has weight at least $(1-2\omega)\|ab\|$.

Now we connect the points in M. The edges of the tree so far subdivide M into quadrilateral regions, which are defined by two edges of the tree. We again want to connect the vertices in such a subregion in a star like fashion. From the interior of such a subregion at least one boundary edge between a point from L and a point from R is fully visible. For every subregion we pick the better of the two stars centered at the two endpoints of such an edge (red edges). By Lemma 2.1 this yields a total additional weight of at least $0.5 \cdot \beta_M(1-2\omega)\|ab\|$.

Recall that $\alpha + \beta_L + \beta_M + \beta_R = 1$. By bounding the maximum by the average, we get

$$\begin{aligned}
\max\{|B_{ab}|, |B_{ba}|\} &\geq \frac{n\|ab\|}{2}((\beta_L + \beta_R)(2-3\omega) + \beta_M(1-2\omega)) \\
&= \frac{n\|ab\|}{2}((1-\alpha)(2-3\omega) - \beta_M(1-\omega)) \\
&\geq \frac{n\|ab\|}{2}((1-\hat{\alpha})(2-3\omega) - \hat{\beta}(1-\omega)). \\
\frac{\max\{|B_{ab}|, |B_{ba}|\}}{|T_{\text{OPT}}|} &\geq \frac{\frac{n\|ab\|}{2}((1-\hat{\alpha})(2-3\omega) - \hat{\beta}(1-\omega))}{\|ab\|n} = \delta.
\end{aligned}$$

◀

▶ **Theorem 3.5.** *A $\delta = 0.512$-approximation for the longest noncrossing Euclidean spanning tree can be computed in polynomial time.*

Proof. We compute S_p for each $p \in P$. Additionally, for each pair a, b with $\|ab\| > d = 1/(2\delta)$, we compute B_{ab} and B_{ba}. Let T_{ALG} be the largest of these structures.

By the exhaustive case distinction in Lemmas 3.1 to 3.4, for the pair a, b which leads to the longest edge in T_{OPT} this leads to a $\delta = 0.512$-approximation. ◀

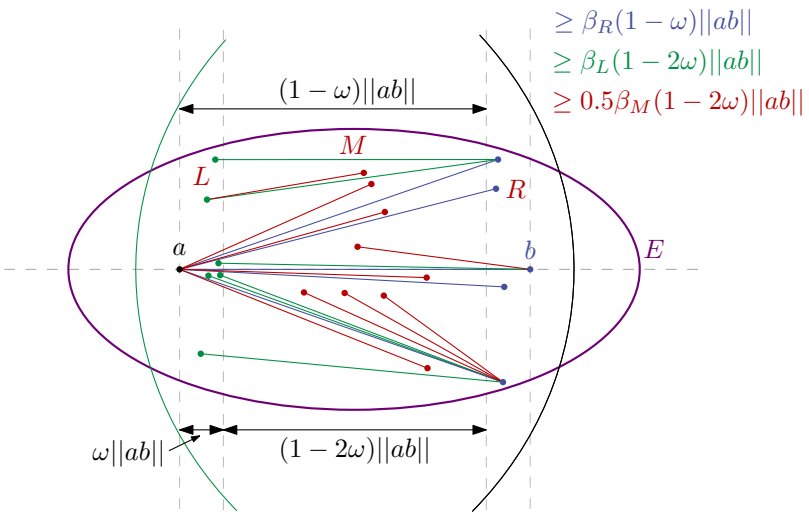


Figure 5 Structure B_{ab}. The edges of each stage of the construction have a different color.

4 Improved approximation factor for thin point sets

In this section we present stronger bounds for thin point sets. Given $\sigma > 0$, we say that P is (at most) *σ-thick* if there exists a diameter of P such that all points in P have distance at most σ from this diameter. Moreover, let T_{CR} be the longest (possibly crossing) tree on P.

▶ **Theorem 4.1.** *There is a polynomial-time algorithm that, given a σ-thick point set P with $\sigma \leq \frac{1}{3}$, constructs a planar spanning tree T_{ALG} with*

$$|T_{\text{ALG}}| \geq f(\sigma) \cdot |T_{\text{CR}}| \geq f(\sigma) \cdot |T_{\text{OPT}}|,$$

where $f(\sigma)$ is given by

$$f(\sigma) = \frac{2}{3} \cdot \sqrt{\frac{1+4\sigma^2}{5-4\sqrt{1-\sigma^2}+4\sigma^2}}.$$

Inspecting the function $f(\sigma)$, we get, e.g., $f(0.3) \geq 0.516$ and $f(0.1) \geq 0.636$. Also, in the limit $d \to 0$ we get $f(\sigma) \to 2/3$. The constant $2/3$ here is tight: There exist perturbations of point sets lying on a segment for which the longest planar trees have length arbitrarily close to $2/3$ of the length of the longest (possibly crossing) tree (see Figure 6).

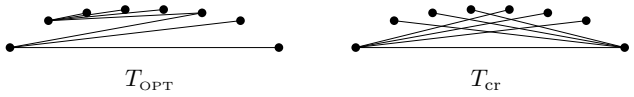


Figure 6 A thin convex set consisting of $n+1$ points with equally spaced x-coordinates $0, 1, \ldots, n$. For large n, the length of any longest planar tree is $1+2+\cdots+n \approx \frac{1}{2}n^2$, whereas the length of the longest (possibly crossing) tree is roughly $2 \cdot (n/2 + \cdots + n) \approx \frac{3}{4}n^2$. Thus, as $n \to \infty$, we obtain $|T_{\text{OPT}}|/|T_{\text{CR}}| \to \frac{2}{3}$.

Proof. [of Theorem 4.1] Fix P and $\sigma \leq \frac{1}{3}$. Denote the relevant diameter of P by pq, and without loss of generality place it as $p = (0,0)$, $q = (1,0)$. Divide $P \setminus \{p, q\}$ by a vertical line ℓ into a set P_p of points closer to p and a set P_q of points closer to q (see Figure 7(a)).

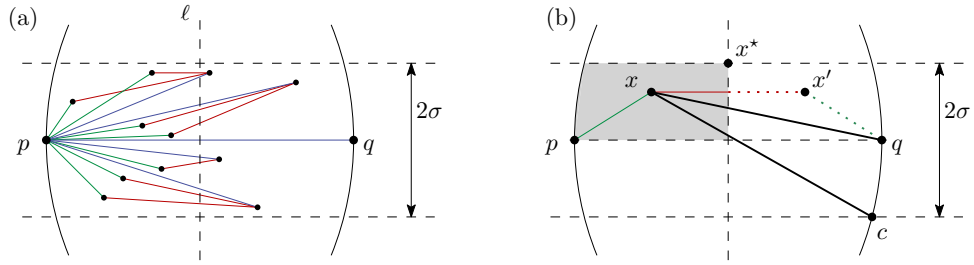


Figure 7 (a) We star the points in the right half from p (blue) and then either star the points in the left half from p too (yielding S_p, blue and green) or connect them to points in the right half (yielding T_{pq}, blue and red). (b) With the shown notation we have $f(\sigma) = 2\|x^\star q\|/(3\|x^\star c\|)$.

We construct a tree T_{pq} as follows: Connect p to all points in $P_q \cup \{q\}$. This splits P_p into wedges with apex p. For each wedge, connect all its points in P_p to the endpoint of its upper side in P_q (use the lower side for the uppermost wedge). Note that T_{pq} is planar. We construct T_{qp} in a symmetric fashion and set T_{ALG} to be the longest of T_{pq}, T_{qp}, S_p, S_q.

Next we argue that T_{ALG} satisfies $|T_{\text{ALG}}| \geq f(\sigma) \cdot |T_{\text{CR}}|$. It suffices to show

$$\frac{|S_p| + 2|T_{pq}| + 2|T_{qp}| + |S_q|}{6} \geq f(\sigma) \cdot |T_{\text{CR}}|.$$

Note that all four trees on the left-hand side include edge pq and since pq is a diameter, we can without loss of generality assume that T_{CR} contains it too. Direct all other edges of those five trees towards pq. Fix a point $x \in P \setminus \{p, q\}$ and let $x_{\text{CR}}, x_{pq}, x_{qp}$ be the other endpoints of the edges pointing from x in $T_{\text{CR}}, T_{pq}, T_{qp}$, respectively. (Note that in S_p all edges point towards p, similarly for S_q and q.) It suffices to prove that

$$\frac{\|xp\| + 2\|xx_{pq}\| + 2\|xx_{qp}\| + \|xq\|}{6 \cdot \|xx_{\text{CR}}\|} \geq f(\sigma)$$

Without loss of generality, suppose that x belongs to P_p and lies above pq. Let x' be the reflection of x about ℓ and c the furthest point from x within the intersection of unit disks centered at p and q. Using the triangle inequality in $\triangle pxx'$, the left-hand side is at least

$$\frac{\|xp\| + \|xx'\| + 2\|xq\| + \|xq\|}{6\|xc\|} \geq \frac{\|px'\| + 3\|xq\|}{6\|xc\|} = \frac{2}{3} \cdot \frac{\|xq\|}{\|xc\|}.$$

Since $\sigma \leq \frac{1}{3}$, the ratio $\|xq\|/\|xc\|$ is minimized when $x = x^\star$ lies on ℓ with distance σ from pq (see Figure 7(b)). Since $\|pc\| = 1$, using the Pythagorean theorem, we easily compute

$$\|x^\star c\| = \sqrt{\left(\sqrt{1-\sigma^2} - 1/2\right)^2 + (2\sigma)^2} \quad \text{and} \quad \|x^\star q\| = \sqrt{(1/2)^2 + \sigma^2},$$

which matches the desired expression $f(\sigma)$. ◀

5 Conclusion

We showed that it is possible to significantly increase the approximation factor from 0.503 to 0.512 in the general case and even towards 2/3, when the point set is σ-thick for $\sigma \to 0$.

The improvement in the approximation factor relies in one case on the planarity of the optimum tree. Without further analysis this does not yield a better approximation factor with regard to the longest crossing tree.

In future work, we aim to further reduce the running time and the approximation factor. For the latter we plan to build on the fact that T_{OPT} is noncrossing, which can lead to further advances. The last open problem would be to settle the question of NP-hardness.

References

1 Noga Alon, Sridhar Rajagopalan, and Subhash Suri. Long non-crossing configurations in the plane. *Fundam. Inform.*, 22(4):385–394, 1995. doi:10.3233/FI-1995-2245.

2 Ahmad Biniaz, Prosenjit Bose, Kimberly Crosbie, Jean-Lou De Carufel, David Eppstein, Anil Maheshwari, and Michiel Smid. Maximum Plane Trees in Multipartite Geometric Graphs. *Algorithmica*, 81(4):1512–1534, 2019. URL: http://link.springer.com/10.1007/s00453-018-0482-x, doi:10.1007/s00453-018-0482-x.

3 Adrian Dumitrescu and Csaba D. Tóth. Long non-crossing configurations in the plane. *Discrete Comput. Geom.*, 44(4):727–752, 2010. doi:10.1007/s00454-010-9277-9.

The Tree Stabbing Number is not Monotone

Wolfgang Mulzer[1] and Johannes Obenaus[*2]

1 **Freie Universität Berlin**
`mulzer@inf.fu-berlin.de`
2 **Freie Universität Berlin**
`johannes.obenaus@fu-berlin.de`

Abstract

Let $P \subseteq \mathbb{R}^2$ be a set of points and T be a spanning tree of P. The *stabbing number* of T is the maximum number of intersections any line in the plane determines with the edges of T. The *tree stabbing number* of P is the minimum stabbing number of any spanning tree of P. We prove that the tree stabbing number is not a monotone parameter, i.e., there exist point sets $P \subsetneq P'$ such that $\text{TREE-STAB}(P) > \text{TREE-STAB}(P')$, answering a question by Eppstein [4, Open Problem 17.5].

1 Introduction

Let $P \subseteq \mathbb{R}^2$ be a set of points in general position, i.e., no three points lie on a common line. A *geometric graph* $G = (P, E)$ is a graph equipped with a drawing where edges are realized as straight-line segments. The *stabbing number* of G is the maximum number of proper intersections that any line in the plane determines with the edges of G. Let $\mathcal{G}$ be a graph class (e.g., trees, paths, triangulations, perfect matchings etc.). The *$\mathcal{G}$-stabbing number* of P is the minimum stabbing number of any geometric graph $G = (P, E)$ belonging to $\mathcal{G}$ (as a function of P).

Stabbing numbers are a classic topic in computational geometry and received a lot of attention both from an algorithmic as well as from a combinatorial perspective. We mainly focus on the stabbing number of spanning trees (see, e.g., [11] for more information), which has numerous applications. For instance, Welzl [10] used spanning trees with low stabbing number to efficiently answer triangle range searching queries, Agarwal [1] used them in the context of ray shooting (also see [2,3] for more examples). Furthermore, Fekete, Lübbecke and Meijer [5] proved $\mathcal{NP}$-hardness of stabbing numbers for several graph classes, namely for spanning trees, triangulations and matchings, though for paths this question remains open.

It is natural to ask whether stabbing numbers are monotone, i.e., does it hold for any pointset $P \subseteq \mathbb{R}^2$ that the $\mathcal{G}$-stabbing number of P is not smaller than the $\mathcal{G}$-stabbing number of any proper subset $P' \subsetneq P$. Recently, Eppstein [4] gave a detailed analysis of several parameters that are monotone and depend only on the point set's order type. Clearly, stabbing numbers depend only on the order type. Eppstein observed that the path stabbing number is monotone [4, Observation 17.4] and asked whether this is also the case for the tree stabbing number [4, Open Problem 17.5]. We prove that neither the tree stabbing number (Corollary 3.4) nor the triangulation stabbing number (Corollary 4.2) nor the matching stabbing number (Corollary 5.2) are monotone. A more detailed analysis can also be found in the second author's Master thesis [9]. Each of the following sections is dedicated to one graph class.

* Partially supported by ERC StG 757609

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

2 Path Stabbing Number

For completeness we repeat the main argument that the path stabbing number, denoted by PATH-STAB$(\cdot)$, is monotone, which can be found in [4, Observation 17.4] for example.

▶ **Lemma 2.1.** *Let G be a geometric graph. The following two operations do not increase the stabbing number of G:*

1. *Removing a vertex of degree 1.*
2. *Replacing a vertex v of degree 2 with the segment connecting its two neighbours w_1, w_2.*

Proof. Clearly, the first operation cannot increase the stabbing number, since it does not add any new segments.

For the second part, let G' be the geometric graph obtained from G by performing operation 2 and let ℓ be an arbitrary line. If ℓ has strictly less than STABBING-NUMBER(G) intersections in G, it has at most STABBING-NUMBER(G) intersections in G', since we added only one segment. Otherwise, if ℓ has STABBING-NUMBER(G) intersections in G, it clearly does not pass through any vertex of G and if ℓ intersects the newly inserted segment $\overline{w_1w_2}$ it must have also intersected either $\overline{w_1v}$ or $\overline{vw_2}$. ◀

▶ **Corollary 2.2.** PATH-STAB$(\cdot)$ *is monotone.*

3 Tree Stabbing Number

We construct point sets $P_1 \subsetneq P_2$ of size n and $n+1$ such that TREE-STAB$(P_1) >$ TREE-STAB(P_2). The point $p \in P_2 \setminus P_1$ we want to remove, must, of course, have degree at least 3 in any spanning tree of minimum stabbing of P_2, since otherwise the arguments of Lemma 2.1 apply.

Our construction, which is depicted in Figure 1 (a), is as follows. Start with a unit circle around the origin O and place 3 evenly distributed points x_1, x_2, x_3 on this circle (in counterclockwise order). Next, add an "arm" consisting of 2 points y_i, z_i $(i = 1, 2, 3)$ at each of the x_i (outside the circle) such that the points O, x_i, y_i, z_i form a convex chain for $i = 1, 2, 3$ (which are all three oriented the same way). These arms need to be flat enough, i.e., the line supporting the segment $\overline{x_iy_i}$ must intersect the interior of the segment $\overline{Ox_{i+2}}$ (indices are taken modulo 3), but also curved enough, i.e., the line supporting the segment $\overline{y_iz_i}$ must have the remaining 8 points on the same side. In particular, there are lines intersecting the segments $\overline{x_iy_i}$, $\overline{y_iz_i}$ and also $\overline{Ox_{i+2}}$ on the one hand and $\overline{y_{i+2}z_{i+2}}$ on the other hand (the red lines in Figure 1 (a)). If there is no danger of confusion, we might omit that indices are taken modulo 3 (as in the previous sentence).

Define the two point sets P_1, P_2 (which are both in general position) to be

$$P_1 = \{x_1, y_1, z_1, x_2, y_2, z_2, x_3, y_3, z_3\}, \qquad P_2 = P_1 \cup \{O\}.$$

▶ **Lemma 3.1.** *It holds that* TREE-STAB$(P_1) = 4$ *and* TREE-STAB$(P_2) \leq 3$.

Proof. This result was obtained by a computer-aided brute-force search (the source code is available on github [8]). In order to compute the stabbing number of a given geometric graph spanning some point set, it is enough to consider a *representative set* H_P of lines. For any line ℓ that partitions the point set into two non-empty subsets, there is a line in the representative set inducing the same partitioning. For an n-point set in general position, the size of a representative set is $\binom{n}{2}$ (see the full version of this paper [7]). Hence, we

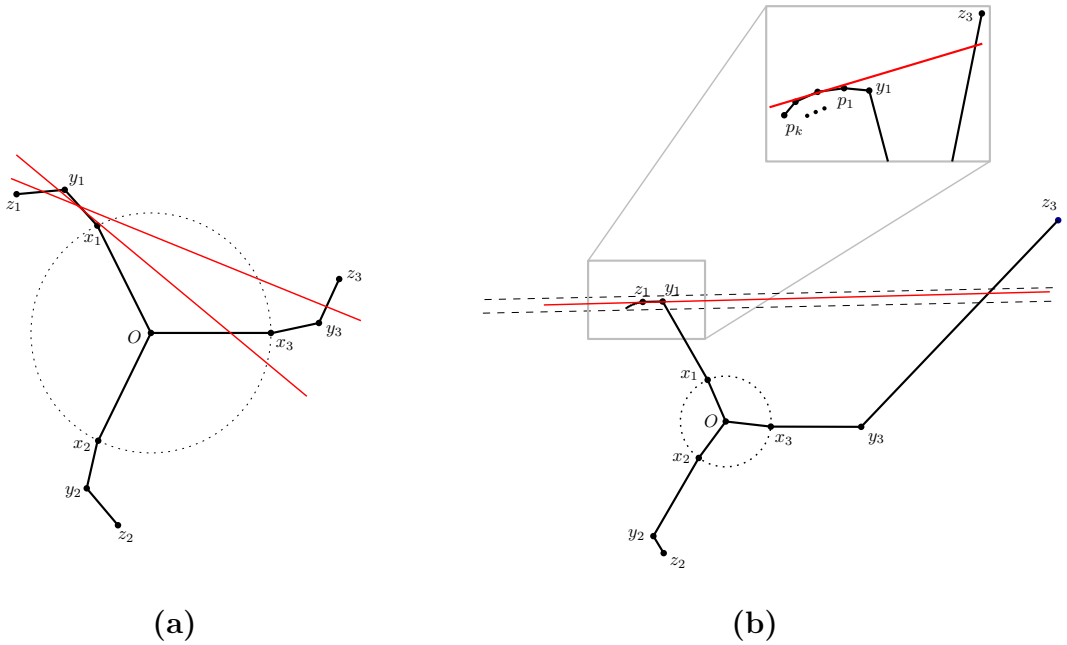


Figure 1 Illustration of a set of (a) 9 points and (b) n points such that removing the point O increases the tree stabbing number.

have $|H_{P_1}| = 36$ and $|H_{P_2}| = 45$. The sets H_{P_1} and H_{P_2} were also obtained by computer assistance. Any pair of points induces four distinct representative lines, computing these and removing duplicates yields H_{P_1} and H_{P_2} (as in [6] for example).

Now, it is enough to compute – for all $9^7 = 4782969$ possible spanning trees on P_1 – their intersections with the lines in H_{P_1}, yielding $\text{TREE-STAB}(P_1) = 4$.

On the other hand, for P_2 the spanning tree depicted in Figure 1 has stabbing number 3 (again by computing all intersections with lines in H_{P_2}) implying $\text{TREE-STAB}(P_2) \leq 3$. ◀

Next, we generalize this construction to arbitrarily large point sets. We simply replace one of the z_i (say z_1) by a convex chain C consisting of k points $p_1, \ldots, p_k$ (see Figure 1 (b)). Denote the convex chains x_1y_1C, $x_2y_2z_2$ and $x_3y_3z_3$ by C_1, C_2 and C_3.

Our goal will be to remove all but two points of $C \cup \{y_1\}$ to get back to our 9-point setting. Of course, it is crucial to keep the relative position of the points as it is in the 9-point set. Thus, place the points $p_1, \ldots, p_k$ such that:

1. $O, x_1, y_1, p_1, \ldots, p_k$ forms a convex chain.
2. close enough to y_1, so that the order type of the resulting point set is the same no matter which $k-1$ of the points in $C \cup \{y_1\}$ we remove. In particular, no line through any two points not belonging to $y_1, p_1, \ldots, p_k$ may separate these points.
3. for any two segments formed by any triple of points in C_1 (consecutively along the convex chain) there is a line intersecting these two segments and also $\overline{y_3z_3}$. To achieve this, C needs to be sufficiently flat and z_3 needs to be pushed further away.

Note that Lemma 3.1 has been verified to still hold after the modification of pushing z_3 further out. Before proving that this construction fulfills the desired properties, we need one more preliminary lemma (see Figure 2).

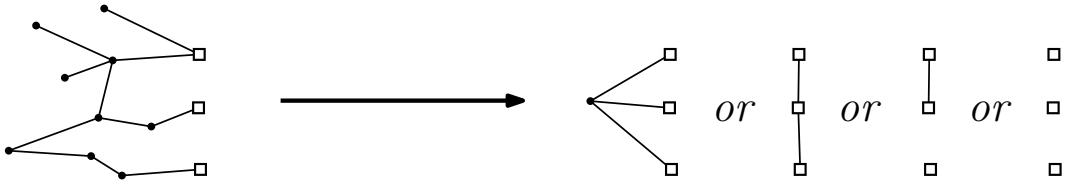


Figure 2 Illustration of Lemma 3.2. Special vertices are depicted as squares. Other vertices of degree 1 or 2 are successively removed.

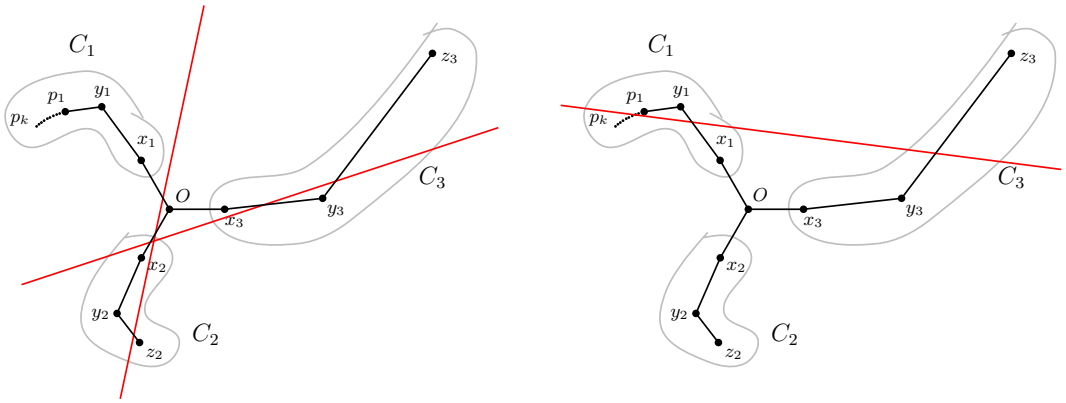


Figure 3 There is no line that intersects more than 3 segments in this spanning tree.

▶ **Lemma 3.2.** *Let $G = (V, E)$ be a forest with c connected components and $|V| \geq 4$. Mark three of the vertices as special (call them v_1, v_2, v_3) and iteratively remove/replace vertices of degree 1 and 2 (as in Lemma 2.1) until no non-special vertex of degree ≤ 2 remains. Then the resulting graph is a forest and consists of the three special vertices and at most one non-special vertex.*

The proof is straightforward and can be found in the full version of this paper [7]. Now, we are prepared to prove our main lemma.

▶ **Lemma 3.3.** *For any integer $n \geq 9$, there exist (planar) point sets $P'_1 \subsetneq P'_2$ of size $|P'_1| = n$ and $|P'_2| = n + 1$ such that* TREE-STAB(P'_1) > TREE-STAB(P'_2).

Proof. Let $k = n - 8$ and define P'_1 and P'_2 as above (Figure 1 (b)), replacing z_1 by $p_1, \ldots, p_k$:

$$P'_1 = \{x_1, y_1, p_1, \ldots, p_k, x_2, y_2, z_2, x_3, y_3, z_3\}, \qquad P'_2 = P'_1 \cup \{O\}.$$

On the one hand, it is straightforward to see that the spanning tree depicted in Figure 1 (b) has stabbing number 3 (see Figure 3 for an illustration) and hence TREE-STAB$(P'_2) \leq 3$.

On the other hand, we show TREE-STAB$(P'_1) \geq 4$ next. Assume for the sake of contradiction that there is a spanning tree T of P'_1 with stabbing number at most 3. Our goal will be to carefully remove points from P_1 such that the stabbing number of T cannot increase until there are only 9 points left in exactly the same relative position as in Lemma 3.1. Clearly, this would be a contradiction.

Consider the set of edges of T with at least one endpoint among the points in C_1. There are at most 3 edges having only one endpoint in C_1 (we call them *bridges*). If there would be more than 3 bridges, there is a line that intersects at least 4 line segments, namely a line that separates C_1 from the rest. Because of the same reason, not all three bridges can go to the same other component (C_2 or C_3).

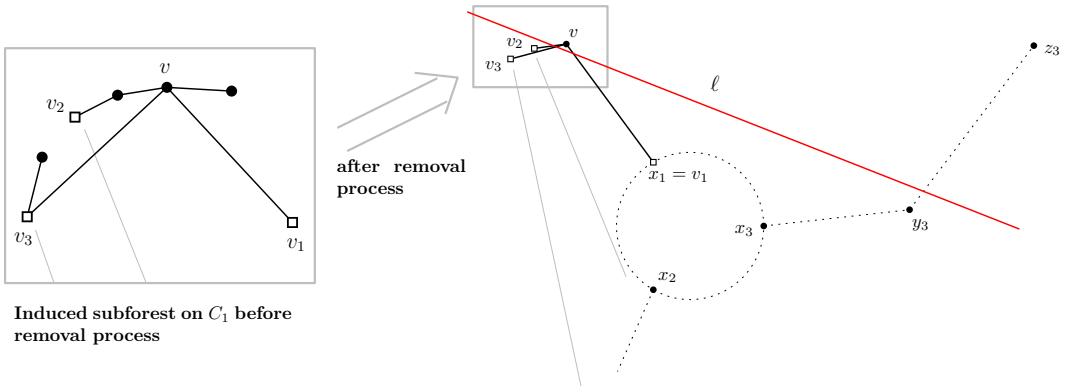


Figure 4 Illustration of Case 2. If a non-special vertex v survives the removal process, the red line has too many intersections.

There are at most 3 points in C_1 that are incident to a bridge and if they are distinct, one of them needs to be x_1, otherwise the line separating x_1 from the rest of C_1 has 4 intersections. Pick three vertices v_1, v_2, v_3 in C_1 such that x_1 and any point incident to a bridge is among them and mark them as special.

Next, we apply Lemma 3.2 to the subforest induced by C_1:

Case 1: No non-special vertex in C_1 survives the removal process.

Then 9 points with the same order type as in Lemma 3.1 and a spanning tree with stabbing number 3 remain, which is a contradiction to Lemma 3.1.

Case 2: One non-special vertex v in C_1 survives the removal process.

Then v is incident to all special vertices v_1, v_2, v_3. If v is the last vertex along C_1, there is obviously a line having more than three intersections. Otherwise, by construction, there is a line ℓ that separates v from v_1, v_2, v_3 and at the same time z_3 from the rest of the point set (see Figure 4). In particular, ℓ has only z_3 and v on one side and all other points on the other. z_3 cannot be adjacent to v, since v is not incident to a bridge and therefore contributes another intersection to ℓ. This is a contradiction to the assumption that T was a spanning tree of stabbing number 3. ◀

▶ **Corollary 3.4.** TREE-STAB$(\cdot)$ *is not monotone.*

4 Triangulation Stabbing Number

We denote the triangulation stabbing number by TRI-STAB$(\cdot)$. Proving non-monotonicity of TRI-STAB$(\cdot)$ is much simpler, only exploiting the additional structure enforced by triangulations. Consider two symmetric convex chains $C_1 = \{p_1, \ldots, p_n\}$ and $C_2 = \{p'_1, \ldots, p'_n\}$ (sufficiently flat) each consisting of n points and facing each other as depicted in Figure 5 (a). These points constitute the point set P. P' consists of the same $2n$ points and two more (slightly perturbed) points added on the line segment connecting the two middle points of C_1 and C_2 (as in Figure 5 (b)). Then the following holds:

▶ **Lemma 4.1.** TRI-STAB$(P) \geq 2n - 1$ *and* TRI-STAB$(P') \leq n + 4 \log n + 3$.

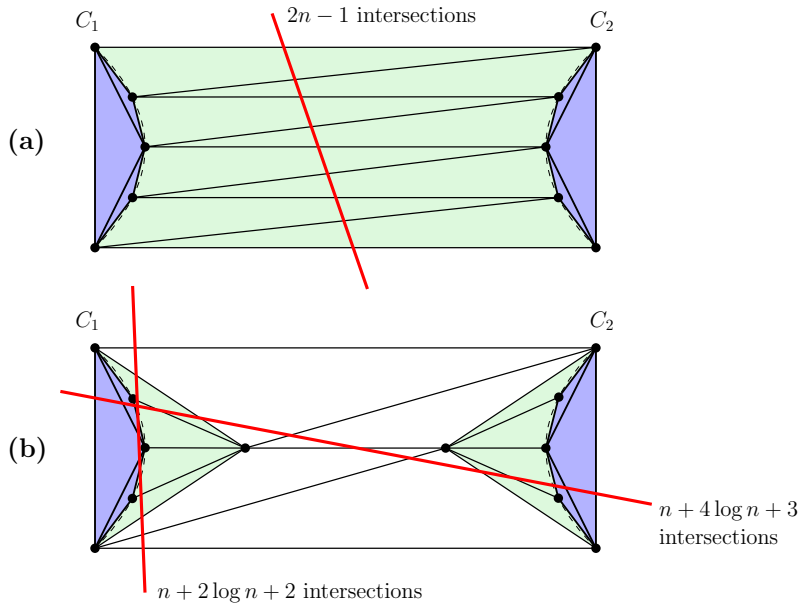


Figure 5 Two symmetric chains in (a) might have a larger triangulation stabbing number compared to the same point set with additional points inbetween (b).

The proof of Lemma 4.1 is straightforward and can be found in the full version of this paper [7].

▶ **Corollary 4.2.** TRI-STAB$(\cdot)$ *is not monotone.*

5 Matching Stabbing Number

First note that the point sets in the case of matchings have to be of even size and all matchings are perfect. Again, we only illustrate the construction, which simply exploits the structure of matchings (again, the proof can be found in the full version [7]).

Take k points $p_1, \ldots, p_k$ in convex position and one point x inside such that any segment $\overline{xp_i}$ is intersected by some $\overline{p_jp_k}$. Next, double all points within a small enough ε-radius (preserving general position) and for a point p name the partner point p' (see Figure 6).

Define the point sets P_1 and P_2 to be:

$$P_2 = \{x, x', p_1, \ldots, p_k, p'_1, \ldots, p'_k, \}, \qquad P_1 = P_2 \setminus \{x', p'_1\}.$$

▶ **Lemma 5.1.** *It holds that* MAT-STAB$(P_1) \geq 3$ *and* MAT-STAB$(P_2) \leq 2$.

▶ **Corollary 5.2.** *The matching stabbing number,* MAT-STAB$(\cdot)$*, is not monotone.*

6 Conclusion

Our proof of Lemma 3.1 relies on computer assistance and of course it would be interesting to turn this into a pen-and-paper proof.

Furthermore, it is easy to generalize stabbing numbers to the context of range spaces $(X, \mathcal{R})$, where X is a set and $\mathcal{R}$ a set of subsets of X, called *ranges*. A spanning path then

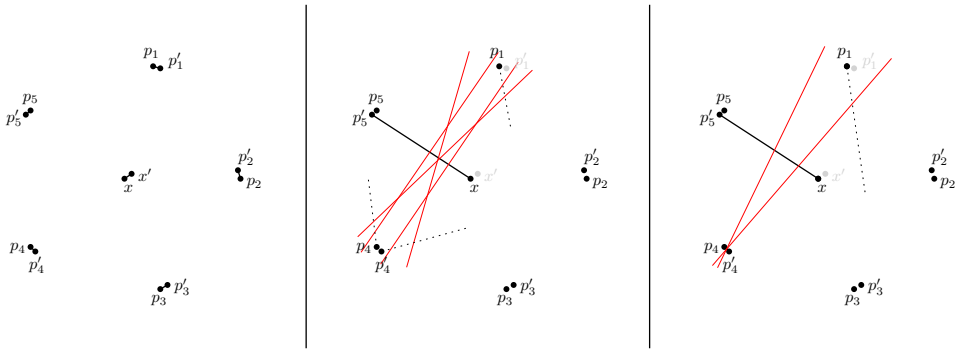


Figure 6 A point set with matching stabbing number 2 in (a) and removing p_1 and x' results in a point set with larger matching stabbing number, illustrated in (b) and (c).

corresponds to a permutation of X and a set $A \subseteq X$ is *stabbed* by a range $r \in \mathcal{R}$ if there are $x, y \in A$ such that $x \in r$ and $y \notin r$. It is straightforward to prove Corollary 2.2 in this context, but we don't know how to apply this for other graph classes.

References

1 Pankaj K. Agarwal. Ray shooting and other applications of spanning trees with low stabbing number. *SIAM J. Comput.*, 21(3):540–570, June 1992. URL: `http://dx.doi.org/10.1137/0221035`, `doi:10.1137/0221035`.

2 P.K. Agarwal, M. Vankreveld, and M. Overmars. Intersection queries in curved objects. *Journal of Algorithms*, 15(2):229 – 266, 1993. URL: `http://www.sciencedirect.com/science/article/pii/S0196677483710400`, `doi:https://doi.org/10.1006/jagm.1993.1040`.

3 Herbert Edelsbrunner, Leonidas Guibas, John Hershberger, Raimund Seidel, Micha Sharir, Jack Snoeyink, and Emo Welzl. Implicitly representing arrangements of lines or segments. *Discrete & Computational Geometry*, 4(5):433–466, Oct 1989. URL: `https://doi.org/10.1007/BF02187742`, `doi:10.1007/BF02187742`.

4 David Eppstein. *Forbidden Configurations in Discrete Geometry*. Cambridge University Press, 2018. `doi:10.1017/9781108539180`.

5 Sándor P. Fekete, Marco E. Lübbecke, and Henk Meijer. Minimizing the stabbing number of matchings, trees, and triangulations. *Discrete & Computational Geometry*, 40(4):595, Oct 2008. URL: `https://doi.org/10.1007/s00454-008-9114-6`, `doi:10.1007/s00454-008-9114-6`.

6 Panos Giannopoulos, Maximilian Konzack, and Wolfgang Mulzer. Low-crossing spanning trees: an alternative proof and experiments. 2014.

7 Wolfgang Mulzer and Johannes Obenaus. The tree stabbing number is not monotone, 2020. full version. `arXiv:2002.08198`.

8 Johannes Obenaus. source code. 2019. URL: `https://github.com/jogo23/stabbing_number_thesis`.

9 Johannes Obenaus. Spanning trees with low (shallow) stabbing number. Master's thesis, ETH Zurich, 9 2019. URL: `https://www.mi.fu-berlin.de/inf/groups/ag-ti/theses/master_finished/obenaus_johannes/index.html`.

10 Emo Welzl. Partition trees for triangle counting and other range searching problems. In *Symposium on Computational Geometry*, 1988.

11 Emo Welzl. *On spanning trees with low crossing numbers*, pages 233–249. Springer Berlin Heidelberg, Berlin, Heidelberg, 1992. URL: https://doi.org/10.1007/3-540-55488-2_30, doi:10.1007/3-540-55488-2_30.

On the maximum number of crossings in star-simple drawings of K_n with no empty lens*

Stefan Felsner[1], Michael Hoffmann[2], Kristin Knorr[3], and Irene Parada[4]

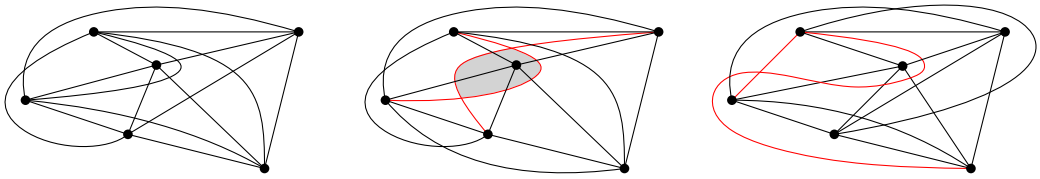
1 **Institute of Mathematics, Technische Universität Berlin, Germany.**
felsner@math.tu-berlin.de
2 **Department of Computer Science, ETH Zürich, Switzerland.**
hoffmann@inf.ethz.ch
3 **Department of Computer Science, Freie Universität Berlin, Germany.**
knorrkri@inf.fu-berlin.de
4 **Institute of Software Technology, Graz University of Technology, Austria.**
iparada@ist.tugraz.at

Abstract

A star-simple drawing of a graph is a drawing in which adjacent edges do not cross and edges are not self-intersecting. In contrast, there is no restriction on the number of crossings between two independent edges. When allowing empty lenses (a face in the arrangement induced by two edges that is bounded by a 2-cycle), two independent edges may cross arbitrarily many times in a star-simple drawing. We consider star-simple drawings of K_n without empty lens. In this setting we prove an upper bound of $3((n-4)!)$ on the maximum number of crossings between any pair of edges. It follows that the total number of crossings is finite and upper bounded by $n!$.

1 Introduction

A *topological drawing* of a graph G is a drawing in the plane where vertices are represented by pairwise distinct points, and edges are represented by Jordan arcs with their vertices as endpoints. Additionally, edges do not contain any other vertices, every common point of two edges is either a proper crossing or a common endpoint, and no three edges cross at a single point. A *simple drawing* is a topological drawing in which adjacent edges do not cross, and independent edges cross at most once.

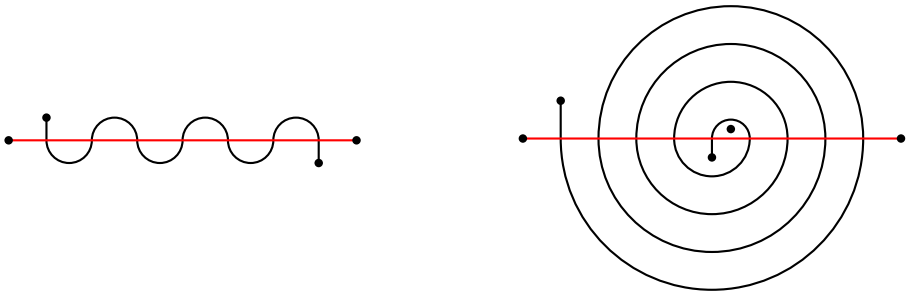


(a) simple **(b)** star-simple but not simple **(c)** not star-simple

Figure 1 Examples for topological drawings of K_6 and a (nonempty) lens (shaded in (b)).

* This research started at the 3rd Workshop within the collaborative DACH project *Arrangements and Drawings*, August 19–23, 2019, in Wergenstein (GR), Switzerland, supported by the German Research Foundation (DFG), the Austrian Science Fund (FWF), and the Swiss National Science Foundation (SNSF). We thank the participants for stimulating discussions. S.F. is supported by DFG Project FE 340/12-1. M.H. is supported by SNSF Project 200021E-171681. K.K. is supported by DFG Project MU 3501/3-1 and within the Research Training Group GRK 2434 *Facets of Complexity*. I.P. is partially supported by FWF project I 3340-N35.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.



(a) twisting **(b)** spiraling

Figure 2 Constructions to achieve an unbounded number of crossings.

We study the broader class of *star-simple* drawings, where adjacent edges do not cross, but independent edges may cross any number of times; see Figure 1 for illustration. In such a drawing, for every vertex v the induced substar centered at v is simple, that is, the drawing restricted to the edges incident to v forms a plane drawing. In the literature (e.g., [1, 2]) these drawings also appear under the name *semi-simple*, but we prefer star-simple because the name is more descriptive.

Star-simple drawings can have regions (not necessarily faces) whose boundary consists of two continuous pieces of (two) edges, we call such a region a *lens*; see Figure 1b. More precisely, a lens is a face in the arrangement induced by two edges and it is bounded by a 2-cycle. A lens is *empty* if it has no vertex in its interior. If empty lenses are allowed, the number of crossings in star-simple drawings of graphs with at least two edges is unbounded (twisting), see Figure 2a. We restrict our attention to star-simple drawings with no empty lens. This restriction is—in general—not sufficient to guarantee a bounded number of crossings (spiraling), as illustrated in Figure 2b. However, we will show that star-simple drawings *of the complete graph* K_n with no empty lens have a bounded number of crossings.

Empty lenses also play a role in the context of the crossing lemma for multigraphs [4]. This is because a group of arbitrarily many parallel edges can be drawn without a single crossing. Hence, for general multigraphs there is no hope to get a lower bound on the number of crossings as a function of the number of edges. However, parallel edges create empty lenses, and disallowing these is enough to make a difference [4].

Kynčl [3, Section 5 Picture hanging without crossings] proposed a construction of two edges in a graph on n vertices with an exponential number (2^{n-4}) of crossings and no empty lens; see Figure 3. This configuration can be completed to a star-simple drawing of K_n,

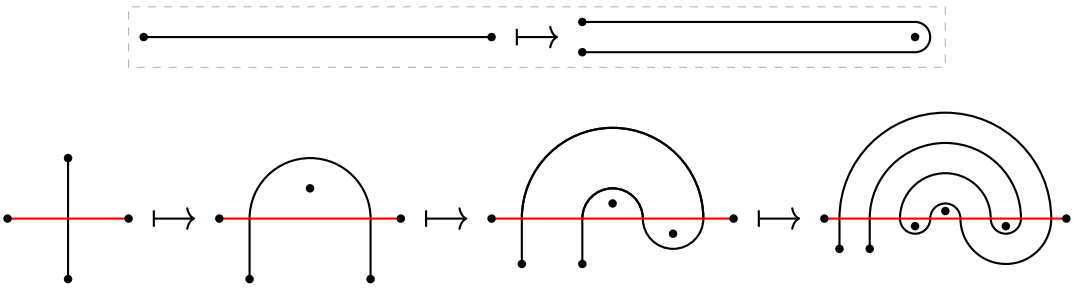
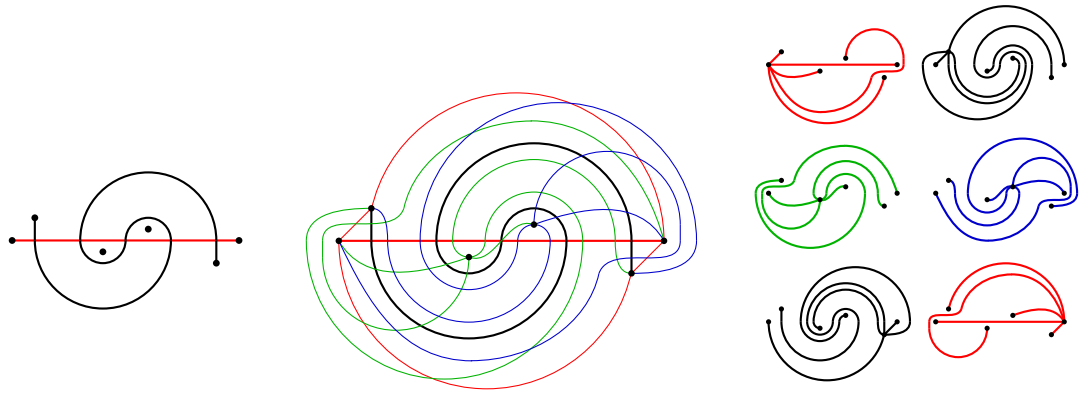


Figure 3 The doubling construction yields an exponential number of crossings.



(a) 2 edges, 5 crossings. **(b)** A star-simple completion of (a). **(c)** The stars of the drawing.

Figure 4 Two edges with $2^{n-4} + 2^{n-6}$ crossings in a star-simple drawing of K_n, for $n = 6$.

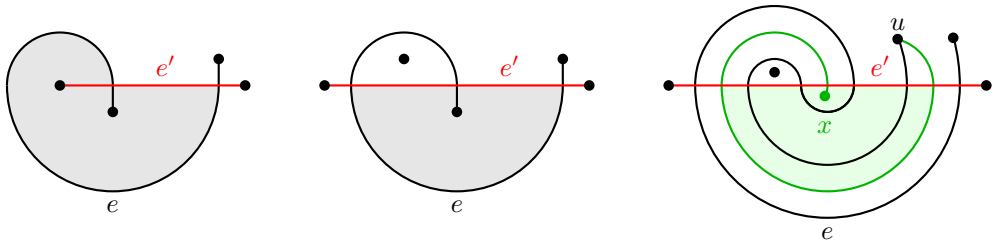
cf. [5]. For $n = 6$ it is possible to have one more crossing while maintaining the property that the drawing can be completed to a star-simple drawing of K_6; see Figure 4. Repeated application of the doubling construction of Figure 3 leads to two edges with $2^{n-4} + 2^{n-6}$ crossings in a graph on n vertices. This configuration can be completed to a star-simple drawing of K_n. We suspect that this is the maximum number of crossings of two edges in a star-simple drawing of K_n.

2 Crossing patterns

In this section we study the induced drawing $D(e, e')$ of two independent edges e and e' in a star-simple drawing D of the complete graph.

▶ **Lemma 2.1.** *The four vertices incident to e and e' belong to the same region of $D(e, e')$.*

Proof. Assuming that the two edges cross at least two times, the drawing $D(e, e')$ has at least two regions. Otherwise, the statement is trivial. If the four vertices do not belong to the same region of $D(e, e')$, then there is a vertex u of e and a vertex v of e' which belong to different regions. Now consider the edge uv in the drawing D of the complete graph. This edge has ends in different regions of $D(e, e')$, whence it has a crossing with at least one of e and e'. This, however, makes a crossing in the star of u or v. This contradicts the assumption that D is a star-simple drawing. ◀



(a) deadlocks **(b)** spiral

Figure 5 Forbidden patterns.

Lemma 2.1 implies that the deadlock configurations as shown in Figure 5(a) do not occur in star-simple drawings of complete graphs. Formally, a *deadlock* is a pair e, e' of edges such that not all incident vertices lie in the same region of the drawing $D(e, e')$.

Now suppose that D is a star-simple drawing of a complete graph with no empty lens. In this case we can argue that e and e' do not form a *spiral* as the black edge e and the red edge e' in Figure 5(b). Indeed, a spiral has an interior lens L and by assumption this lens is non-empty, i.e., L contains a vertex x. Let e and e' be the black and the red edge in Figure 5(b), respectively, and let u be a vertex of e. The edge xu (the green edge in the figure) has no crossing with e, hence, it has to follow the spiral. This yields a deadlock configuration of the edges xu and e'. Note that if in Figure 5(b) instead of drawing the green edge xu we connect x with an edge f to one of the vertices of the red edge e' such that f and the red edge have no crossing, then f and the black edge e form a deadlock.

We use this intuition to formally define a spiral. Two edges e, e' form a *spiral* if they form a lens L such that if we place a vertex x in L and draw a curve γ connecting x to a vertex u of e so that γ does not cross e, then γ and e' form a deadlock.

3 Crossings of pairs of edges

In this section we derive an upper bound for the number of crossings of two edges in a star-simple drawing of K_n with no empty lens. Actually, we show the following.

▶ **Theorem 3.1.** *Consider a star-simple drawing of K_n with no empty lens. If $C(k)$ is the maximum number of crossings of a pair of edges that (a) form no deadlock and no spiral and such that (b) all lenses formed by the two edges can be hit by k points, then $C(k) \leq \mathrm{e} \cdot k!$.*

Proof. Due to Lemma 2.1 we can assume that all four vertices of e and e' are on the outer face of the drawing $D(e, e')$. We think of e' as being drawn red and horizontally and of e as being a black meander edge. Let $p_1, \ldots, p_k$ be points hitting all the lenses of the drawing $D(e, e')$. Let u be one of the endpoints of e. For each $i = 1, \ldots, k$ we draw an edge e_i connecting p_i to u such that e_i has no crossing with e and, subject to this, the number of crossings with e' is minimized. Figure 6 shows an example.

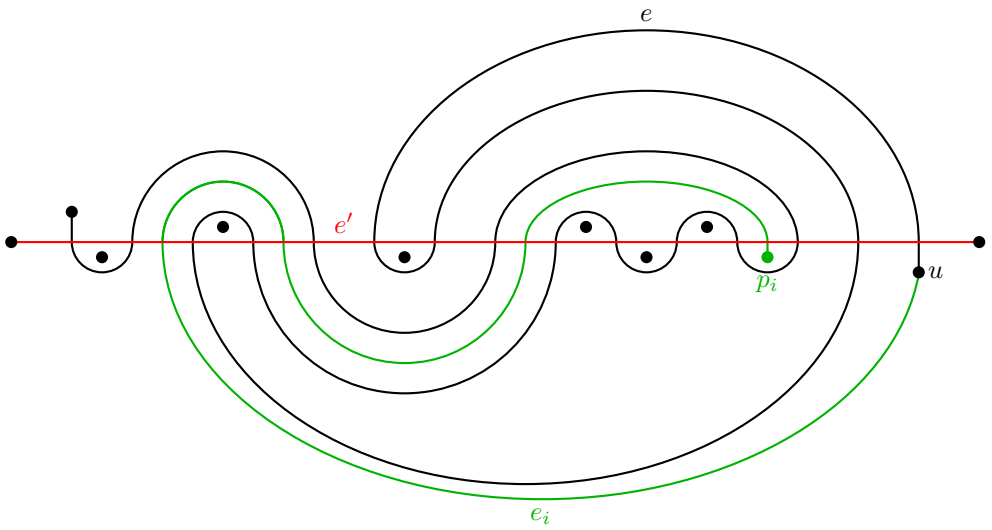


Figure 6 The drawing $D(e, e')$ and an edge e_i connecting p_i to u.

We claim the following three properties:

(P1) The edges e_i and e' form no deadlock and no spiral.
(P2) All the lenses of e_i and e' are hit by the $k-1$ points $p_1, \ldots, p_{i-1}, p_{i+1}, \ldots, p_k$.
(P3) Between any two crossings of e and e' from left to right, i.e., in the order along e', there is at least one crossing of e' with one of the edges e_i.

Before proving the properties we show that they imply the statement of the theorem. From (P1) and (P2) we see that the number X_i of crossings of e_i and e' is upper bounded by $C(k-1)$. From (P3) we obtain that $C(k) \leq 1 + \sum_i X_i$. Combining these we get

$$C(k) \leq k \cdot C(k-1) + 1 \leq k! \cdot \sum_{s=0}^{k} \frac{1}{s!} \leq k! \cdot \mathrm{e}. \qquad \blacktriangleleft$$

For the proof of the three claims we need some notation. Let $\xi_1, \xi_2, \ldots, \xi_N$ be the crossings of e and e' indexed according to the left to right order along the horizontal edge e'. Let g_i and h_i be the pieces of e' and e, respectively, between crossings ξ_i and ξ_{i+1}. The bounded region enclosed by $g_i \cup h_i$ is the *bag* B_i and g_i is the *gap* of the bag. The minimal lenses formed by e and e' are exactly the bags B_i where h_i is a crossing free piece of e. From now on when referring to a *lens* we always mean such a minimal lens. The following is crucial:

▶ **Observation 3.2.** *For two bags B_i and B_j the open interiors are either disjoint or one is contained in the other.*

The observation implies that the containment order on the bags is a downwards branching forest. The minimal elements in the containment order are the lenses. Consider a lens L and the point p_i inside L. Since the vertex u of e is in the outer face of $D(e, e')$ the edge e_i has to leave each bag which contains L and by definition it has to leave a bag B containing L through the gap g of B.

We now reformulate and prove the third claim.

(P3') For each pair ξ_i, ξ_{i+1} of consecutive crossings on e' there is a lens L and a point $p_j \in L$ such that e_j crosses e' between ξ_i and ξ_{i+1}.

Proof of (P3'). The pair ξ_i, ξ_{i+1} is associated with the bag B_i. In the containment order of bags a minimal bag below B_i is a lens, let L be any of the minimal elements below B_i. By assumption, L contains a point p_j. Since $L \in B_i$, we have that also $p_j \in B_i$. Thus, it follows that e_j has a crossing with the gap g_i, i.e., e_j has a crossing with e' between ξ_i and ξ_{i+1}. $\blacktriangleleft$

Proof of (P1). We have to show that e_i and e' form no deadlock and no spiral. The minimality condition in the definition of e_i implies that if $L = B_{i_1} \subset B_{i_2} \subset \ldots \subset B_{i_t}$ is the maximal chain of bags with minimal element L then e_i is crossing the gaps of these bags in the given order and has no further crossings with e'. If γ is a curve from L to u which avoids e then in the ordered sequence of gaps crossed by γ we find a subsequence which is identical to the ordered sequence of gaps crossed by e_i. Since e and e' form no spiral there is such a curve γ which forms no deadlock with e' but then e_i forms no deadlock with e'.

Now assume that e_i and e' form a spiral. Let B be the largest bag containing p_i. Think of B as a drawing of e_i with a broad pen which may also have some extra branches which have no correspondence in e_i, see Figure 7. The formalization of this picture is that for every bag β formed by e_i with e' there is a bag $B(\beta)$ formed by e and e' with $B(\beta) \subset \beta$. Now, if there is a lens λ formed by e_i with e' such that every e_i-avoiding curve to u is a deadlock

with e', then there is a lens $L(\lambda)$ formed by e and e' with $L(\lambda) \subset \lambda$ and every e-avoiding curve to u is also B-avoiding and hence e_i-avoiding. This show that every such curve has a deadlock with e', whence e and e' form a spiral, contradiction. ◀

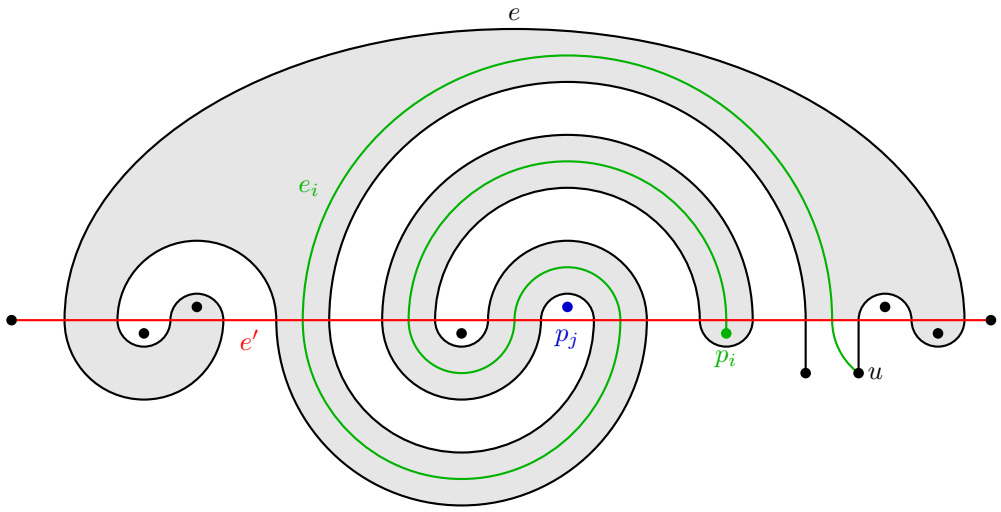


Figure 7 An edge e_i(green) forming a spiral with e'. The bag B in gray and the lens $L(\lambda)$ marked with the vertex p_j(blue).

Proof of (P2). We already know that e_i and e' form no deadlock. Therefore, by Lemma 2.1, the vertices of e_i and e' belong to the same region of $D(e_i, e')$. All crossings of e_i with e' correspond to bags of e and e', therefore the vertices of e and e' are in the outer face of $D(e_i, e')$. Together this shows that p_i is also in the outer face of $D(e_i, e')$. Since every lens of $D(e_i, e')$ contains a lens of $D(e, e')$, it also contains one of the points hitting all lenses of $D(e, e')$. Hence, all lenses of $D(e_i, e')$ are hit by the $k-1$ points $p_1, \ldots, p_{i-1}, p_{i+1}, \ldots, p_k$. ◀

4 Crossings in complete drawings

Accounting for the four endpoints of the two crossing edges we have $k \leq n-4$ in Theorem 3.1. Therefore, we obtain that the number of crossings of a pair of edges in a star-simple drawing of K_n without empty lens is upper bounded by $\mathrm{e}(n-4)!$. This directly implies that the drawing of K_n has at most $n!$ crossings. We know drawings of K_n in this drawing mode with an exponential number of crossings. It would be interesting to reduce the huge gap between upper and lower bound.

References

1 Oswin Aichholzer, Florian Ebenführer, Irene Parada, Alexander Pilz, and Birgit Vogtenhuber. On semi-simple drawings of the complete graph. In *Abstracts of the XVII Spanish Meeting on Computational Geometry (EGC'17)*, pages 25–28, 2017.

2 Martin Balko, Radoslav Fulek, and Jan Kynčl. Crossing numbers and combinatorial characterization of monotone drawings of K_n. *Discrete & Computational Geometry*, 53(1):107–143, 2015. doi:10.1007/s00454-014-9644-z.

3 Jan Kynčl. Simple realizability of complete abstract topological graphs simplified, 2016. arxiv.org/abs/1608.05867v1.

4 János Pach and Géza Tóth. A crossing lemma for multigraphs. In *Proc. SoCG 2018*, volume 99 of *LIPIcs*, pages 65:1–65:13, 2018. doi:10.4230/LIPIcs.SoCG.2018.65.

5 Irene Parada. *On straight-line and topological drawings of graphs in the plane*. PhD thesis, Graz University of Technology, 2019.

Simple Topological Drawings of k-Planar Graphs*

Chih-Hung Liu[1], Meghana M. Reddy[†1], and Csaba D. Tóth[2,3]

1 **Department of Computer Science, ETHZ, Zürich, Switzerland**
chih-hung.liu@inf.ethz.ch, meghana.mreddy@inf.ethz.ch
2 **Department of Mathematics, Cal State Northridge, Los Angeles, CA, USA**
csaba.toth@csun.edu
3 **Department of Computer Science, Tufts University, Medford, MA, USA**

Abstract

Every graph that admits a topological drawing also admits a simple topological drawing. It is easy to observe that every 1-planar graph admits a 1-plane simple topological drawing. It has been shown that 2-planar graphs and 3-planar graphs also admit 2-plane and 3-plane simple topological drawings respectively. However, nothing has been proved for simple topological drawings of k-planar graphs for $k \geq 4$. In fact, it has been shown that there exist 4-planar graphs which do not admit a 4-plane simple topological drawing, and the idea can be extended to k-planar graphs for $k > 4$. We prove that there exists a function $f : \mathbb{N} \to \mathbb{N}$ such that every k-planar graph admits an $f(k)$-plane simple topological drawing for all $k \in \mathbb{N}$. This answers a question posed by Schaefer.

1 Introduction

1.1 Problem Statement

A *topological drawing* of a graph G in the plane is a representation of G in which the vertices are mapped to distinct points in the plane and edges are mapped to Jordan arcs that do not pass through (the images of) vertices and no three Jordan arcs pass through the same point in the plane. A graph is *k-planar* if it admits a topological drawing in the plane where every edge is crossed by other edges at most k times, and such a drawing is called a *k-plane drawing*. A *simple topological drawing* of a graph refers to a topological drawing where no two edges cross more than once and no two adjacent edges cross. We study simple topological drawings of k-planar graphs.

It is well known that drawings of a graph G that attain the minimum number of crossings (i.e., the *crossing number* $\mathrm{cr}(G)$ of G) are simple topological drawings. However, a drawing that minimizes the total number of crossings need not minimize the maximum number of crossings per edges; and a drawing that minimizes the maximum number of crossings per edge need not be simple. A *k-plane simple topological drawing* is a simple topological drawing where every edge is crossed at most k times. We study the simple topological drawings of k-planar graphs and prove that there exists a function $f : \mathbb{N} \to \mathbb{N}$ such that every k-planar graph admits an $f(k)$-plane simple topological drawing by designing an algorithm to obtain the plane simple topological drawing from a k-plane drawing of a k-planar graph.

In a k-plane drawing of a graph, every edge is crossed at most k times. However, adjacent edges may cross, and a pair of edges may cross multiple times. To obtain a simple topological drawing, we need to eliminate crossings between adjacent edges and ensure that no two edges cross more than once, without introducing self-intersections of edges during the process.

* This work was initiated during the $\mathbf{17}^{\boldsymbol{th}}$ Gremo Workshop on Open Problems 2019. The authors thank the organizers of the workshop for inviting us and providing a productive working atmosphere.
† Supported by the Swiss National Science Foundation within the collaborative DACH project *Arrangements and Drawings* as SNSF Project 200021E-171681.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

1.2 Related Previous Results

It is easy to see that every 1-planar graph admits a 1-plane simple topological drawing [3]. Pach et al. [2, Lemma 1.1] proved that every k-planar graph for $k \leq 3$ admits a k-plane drawing such that any pair of edges have at most one point in common including endpoints. However, these results do not extends to k-planar graphs for $k > 3$. In fact, Schaefer [4, p. 57] constructed k-planar graphs that do not admit a k-plane simple topological drawing for $k = 4$. The construction idea can be extended to all $k > 4$. The *local crossing number* of a graph G, $\mathrm{lcr}(G)$, is the minimum integer k such that G admits a drawing where every edge has at most k crossings. The *simple* local crossing number, $\mathrm{lcr}^*(G)$ minimizes k over simple topological drawings of G. Schaefer [4, p. 59] asked whether the $\mathrm{lcr}^*(G)$ can be bounded by a function of $\mathrm{lcr}(G)$. We answer this question in the affirmative and show that there exists a function $f : \mathbb{N} \to \mathbb{N}$ such that $\mathrm{lcr}^*(G) \leq f(\mathrm{lcr}(G))$.

1.3 Basic Definitions

Given a k-plane drawing D of a graph G, we denote by N the planarization of D, i.e., we introduce a vertex of degree four at every crossing in D. We call this graph a *network*. Since every edge in D has at most k crossings, each edge of G corresponds to a path of length at most $k+1$ in N. Our algorithm successively modifies the drawing D, and ultimately returns a simple topological drawing D of G. We formulate invariants for our algorithm in terms of the planarization N of the initial drawing. In other words, N remains fixed (in particular, N will not be the planarization of the modified drawings). Specifically, our algorithm maintains the following invariants:

(i) every edge in D' closely follows a path of length at most $k+1$ in the network N,
(ii) every pair of edges in D' cross only in a small neighborhood of a node of N, and
(iii) any two edges cross at most once in each such neighborhood.

Lenses in a topological drawings. We start with definitions needed to describe the key operations in our algorithm. In a topological drawing, we define a structure called *lens*. Consider two edges, a and b, that cross more than once. Consider two crossings of the edges a and b, denoted by c_1 and c_2. Let a_{12} denote the portion of the edge a between c_1 and c_2, and b_{12} is defined analogously. The arcs a_{12} and b_{12} together are called a *lens* if the arcs do not intersect except at c_1 and c_2. Similarly a lens could also be formed by two adjacent edges cross each other. Let p be the common vertex of two adjacent edges a and b, and let c be a crossing between a and b. The portions of the edges a and b between p and c form a *lens* if the arcs formed by the portions of the edges between p and c do not cross each other.

▶ **Lemma 1.1.** *If two nonadjacent edges a and b cross more than once, then there exist two crossings c_1 and c_2 of the two edges such that a_{12} and b_{12} form a lens, where a_{12} and b_{12} are the portions of the edges a and b between c_1 and c_2.*

Proof. If the edges a and b cross exactly twice, the arcs between the two crossings form a lens. Now, for the edges a and b that cross more than twice, assume there were no two crossings which form a lens. Then the arcs between any two crossings cross each other. Consider two crossings c_1 and c_2 for which the arcs a_{12} and b_{12} have the minimum number of crossings. Let c' be one of these crossings. Then the portions of the arcs a_{12} and b_{12} between c_1 and c' have at least one fewer crossings, contradicting the minimality assumption in the choice of c_1 and c_2. ◀

▶ **Lemma 1.2.** *If two adjacent edges a and b cross each other once, then there exist two crossings c_1 and c_2 of the two edges such that the arcs a_{12} and b_{12} form a lens, where one of the crossings from c_1 and c_2 can be the common endpoint of the two edges, and a_{12} and b_{12} are the portions of the edges a and b between c_1 and c_2.*

Proof. If the edges a and b crossed exactly once, the arcs between the common endpoint and the point of intersection form a lens. If the edges cross more than once, the proof follows from Lemma 1.1. ◀

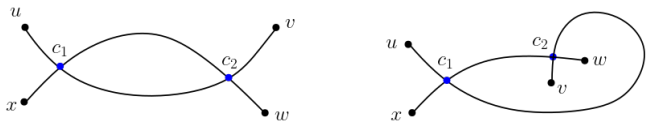


Figure 1 Lenses that can be formed by two edges crossing more than once.

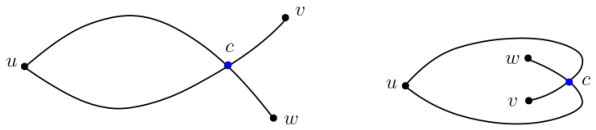


Figure 2 Lenses that can be formed by two adjacent edges crossing each other.

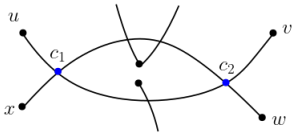
Elimination operation. We further define two operations for our algorithm, each of which modifies one edge in a (current) drawing of G. An *elimination* operation is defined as the redrawing of an edge such that a lens is eliminated.

Let D' be a drawing of G satisfying invariants (i)–(iii). Let $a = (u, v)$ and $b = (x, w)$ be two edges that cross twice, at c_1 and c_2, and let a_{12} and b_{12} be the portions of the edges a and b between c_1 and c_2. Note that other edges may cross the arcs $a_{12}1$ and b_{12}. By (ii) and (iii), the intersection points c_1 and c_2 lie in some small neighborhoods of two distinct nodes of N. By (i), the arcs a_{12} and b_{12} each closely follow some path in N.

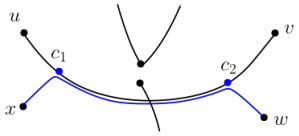
We compare a_{12} and b_{12} using two measures: First, the *length* of a_{12} and b_{12}, respectively, is the (graph-theoretic) length of the path in the network N that they follow. Second, we count the number of crossings of a_{12} (resp., b_{12}) with other edges in the current drawing D'. We define a_{12} to be the *efficient arc* (and b_{12} the *inefficient arc*) if a_{12} has smaller length that b_{12}, or if the two arcs have the same length but a_{12} has fewer crossings in D' and b_{12}. If the two arcs have the same length and the same number of crossings, either of the arcs can be considered as the *efficient arc*. The elimination operation reroutes the inefficient arc to follow the efficient arc to eliminate the lens.

Without loss of generality, assume a_{12} is the efficient arc, $b = xw$, and the points (x, c_1, c_2, w) appear in this order along the drawing of b in D'. To eliminate the lens, we redraw b such that it follows its current arc from x to c_1, and then, without crossing a at c_1, it closely follows the arc a_{12} until c_2, and further follows its current arc from c_2 to w. In this process, the crossing c_1 is eliminated. Figure 3 illustrates this operation where both the crossings are eliminated. Figure 4 illustrates the operation where only one of the crossing is eliminated.

The elimination operation for a lens formed by adjacent edges between the common endpoint and the first crossing is defined similarly. Let $a = (u, v)$ and $b = (u, w)$ be two adjacent edges that cross at c and the arcs a_{12} and b_{12} form a lens. Let a_{12} be the efficient

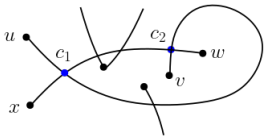


(a) Lens formed by two edges crossing more than once.

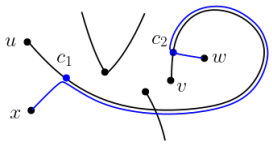


(b) Inefficient curve rerouted and depicted in blue.

Figure 3 Rerouting of edge and eliminating both the crossings.



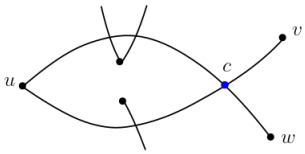
(a) Lens formed by two edges crossing more than once.



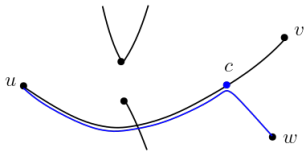
(b) Inefficient curve rerouted and depicted in blue.

Figure 4 Rerouting of edge and eliminating crossing c_1.

arc. We redraw the edge b such that it starts at u and follows arc a_{12} until c and then follows its current arc from c until w by avoiding the crossing c. The redrawing of edge b in this method eliminates the crossing c. Figures 5 and 6 illustrate this operation. If the efficient arc a_{12} crosses the arcs $b \setminus b_{12}2$, then we have introduced self-crossing in the modified edge b. We eliminate self-crossings by removing any loops from the modified arc of b. Note that the elimination operation does not increase the length of any arc or any edge.



(a) Lens formed by two adjacent edges.

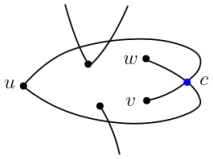


(b) Inefficient curve rerouted and depicted in blue.

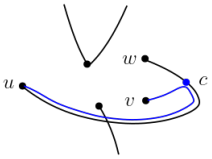
Figure 5 Rerouting of edge to eliminate the lens.

As a result, at least one of the crossings c_1 and c_2 (or crossing c in the case of adjacent edges) is eliminated. However, the elimination of lens may create a new lens if one of the edges crossing the efficient arc crossed the edge corresponding to the inefficient arc as well. This is demonstrated in Figure 7.

In addition, the redrawing of the arcs is done in such a way that the modified arc closely follows an existing path of the network N since the inefficient arc follows closely to the efficient arc. Furthermore, the elimination process can introduce new crossings between the modified inefficient arc and edges crossing the efficient arc. Assume an edge e' crossed the efficient arc at c'. The elimination operation introduces a crossing between e' and the inefficient arc, however, this crossing is very close to the crossing c' and hence is in a small neighborhood of a node of N. Since the new crossing is in a small neighborhood of a node of N, the lengths of the edge e' and the inefficient arc do not increase.



(a) Lens formed by two adjacent edges.



(b) Inefficient curve rerouted and depicted in blue.

Figure 6 Rerouting of edge to eliminate the lens.

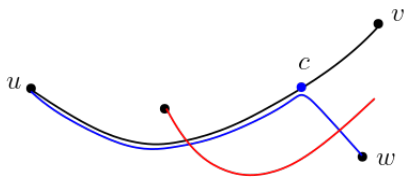


Figure 7 New lens created between red and blue edges after elimination of existing lens.

2 The Algorithm

Given a k-plane topological drawing D of a k-planar graph G, we describe an algorithm using the elimination operations to produce a simple topological drawing D' of graph G from D. Initially, let $D' := D$. While there exists a lens in the drawing D', consider one such lens and eliminate it by modifying the drawing D'. Return the resulting drawing D'.

▶ **Theorem 2.1.** *The algorithm terminates and transforms a k-plane drawing of a k-planar graph into a simple topological drawing.*

Proof. Note that the redrawing of the arcs in the elimination operation was defined with respect to two measures: the length of the arc and number of crossings of the arcs. Let the sum of lengths of all edges in the drawing be defined as the *total length* of the drawing, where the length of an edge is the length of the path in N that the edge closely follows. After each elimination operation, since we reroute the inefficient arc towards the efficient arc, the total length of the drawing monotonically decreases; and if the total length remains the same, then the total number of crossings strictly decreases. Thus, the algorithm terminates, the drawing D' returned by the algorithm does not contain lenses. By Lemmata 1.1 and 1.2, any two edges in the resulting drawing D' cross at most once and adjacent edges do not cross. Additionally, the operations do not introduce any self-crossings. Consequently, the algorithm returns a simple topological drawing. ◀

▶ **Theorem 2.2.** *There exists a function $f(k)$ such that every k-planar graph admits an $f(k)$-plane simple topological drawing, and the $f(k)$-plane simple topological drawing can be obtained from a k-plane drawing of the graph using the above algorithm.*

Proof. Consider the drawing D' returned by our algorithm, and a node v of network N. We analyse the subgraph G_v of G formed by the edges of G that pass through a small neighborhood of v. Let n_v and m_v be the number of vertices and edges of G_v, respectively. Since N is created from a k-plane drawing, and every node corresponding to a crossing has degree 4, there are at most $4 \cdot 3^{k-1}$ vertices of G at distance at most k from v. Hence, $n_v \leq 4 \cdot 3^{k-1}$.

▶ **Lemma 2.3** (Crossing Lemma [1, Theorem 6]). *Let G be a graph with n vertices and m edges and D be a topological drawing of G. Let $cr(D)$ be defined as the total number of crossings in D, and $cr(G)$ be defined as the minimum of $cr(D)$ over all drawings D of G. If $m \geq 6.95n$, then $cr(G) \geq \frac{1}{29}\frac{m^3}{n^2}$.*

We apply Lemma 2.3 to the graph G_v, and obtain two cases:

- Case 1: $m_v < 6.95n$.
- Case 2: $m_v \geq 6.95n$, and thus $\mathrm{cr}(G_v) \geq \frac{1}{29}\frac{m_v^3}{n_v^2}$. Since G_v has m_v edges and each edge has at most k crossings in the drawing D', we obtain $\frac{1}{29}\frac{m_v^3}{n_v^2} \leq \frac{m_v k}{2}$, which implies $m_v \leq \sqrt{\frac{29k}{2}}\, n_v$.

Combining the two cases to obtain an upper bound on m_v, we get $m_v \leq \max\{6.95n_v, \sqrt{\frac{29k}{2}}\, n_v\}$. Further, for $k \geq 4$, $m_v \leq \sqrt{\frac{29k}{2}}\, n_v$.

Since m_v edges pass through a small neighborhood of v, any edge passing through that neighborhood crosses at most $m_v - 1$ edges. Additionally, every edge in G passes through (the neighborhood of) at most k nodes of N. An edge passing through nodes $v_1, \ldots v_k$, crosses at most $\sum_{i=1}^{k}(m_{v_i} - 1)$ edges in D'. Combining the upper bounds on m_v and n_v, we obtain that every edge in the output drawing D' is crossed at most $\sqrt{\frac{29k}{2}} \cdot 4k \cdot 3^{k-1} = \frac{2}{3}\sqrt{58} \cdot k^{3/2} \cdot 3^k$ times for $k \geq 4$. ◀

3 Conclusion

We have proved that every k-planar graph admits a simple topological drawing where every edge is crossed at most $f(k) = \frac{2}{3}\sqrt{58} \cdot k^{3/2} \cdot 3^k$ times. Consequently, $\mathrm{lcr}^* \leq O(\mathrm{lcr}^{3/2} \cdot 3^{\mathrm{lcr}})$. However, we hope that the bound on $f(k)$ can be improved to a polynomial in k by a more careful analysis for the number of crossings per edge created by our algorithm.

References

1 Eyal Ackerman. On topological graphs with at most four crossings per edge. *Computational Geometry*, 85:101574, 2019.

2 János Pach, Radoš Radoičić, Gábor Tardos, and Géza Tóth. Improving the crossing lemma by finding more crossings in sparse graphs. *Discrete Comput. Geom.*, 36(4):527–552, 2006. doi:10.1007/s00454-006-1264-9.

3 Gerhard Ringel. Ein Sechsfarbenproblem auf der Kugel. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 29:107–117, 1965. doi:10.1007/BF02996313.

4 Marcus Schaefer. The graph crossing number and its variants: A survey. *The Electronic Journal of Combinatorics*, 20, 2013. Version 4 (February 14, 2020). doi:10.37236/2713.

Enumerating tilings of triply-periodic minimal surfaces with rotational symmetries

Benedikt Kolbe[1] **and Myfanwy Evans**[2]

1 **Universitè de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France**
`benedikt.kolbe@inria.fr`
2 **Department of Mathematics, Technical University of Berlin, Berlin**

Abstract

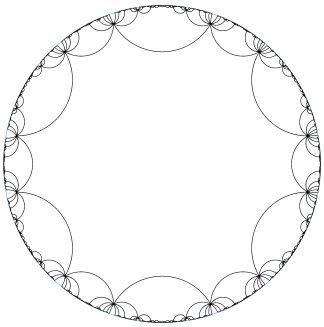
We present a technique for the enumeration of all isotopically distinct ways of tiling, with disks, a hyperbolic surface of finite genus, possibly nonorientable and with punctures and boundary. This provides a generalization of the enumeration of Delaney-Dress combinatorial tiling theory on the basis of isotopic tiling theory. To accomplish this, we derive representations of the mapping class group of the orbifold associated to the symmetry group of the tiling under consideration as a set of algebraic operations on certain generators of the symmetry group. We derive explicit descriptions of certain subgroups of mapping class groups and of tilings as embedded graphs on orbifolds. We further use this explicit description to present an algorithm that we illustrate by producing an array of examples of isotopically distinct tilings of the hyperbolic plane with symmetries generated by rotations that are commensurate with the prominent Primitive, Diamond and Gyroid triply-periodic minimal surfaces, outlining how the approach yields an unambiguous enumeration. We also present the corresponding 3-periodic graphs on these surfaces.

1 Introduction

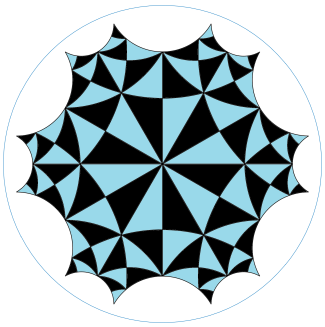
Tesselations from repeating motifs have a long and involved history in mathematics, engineering, art and sciences. Most of the literature has focussed on patterns in Euclidean spaces. However, the role of hyperbolic geometry in Euclidean tilings and more generally for the natural sciences is increasingly recognized. More recently, it has been recognized that assemblies of atoms or molecules in crystalline arrangements that are energetically favourable involve (intrinsic) curvature [15]. Many real Zeolite frameworks and metal-organic frameworks were found to reticulate triply-periodic minimal surfaces (TPMS) [13, 14, 15, 4]. TPMS are minimal surfaces, which locally minimize the surface area relative to a boundary curve of a simply connected neighbourhood around any point, which are furthermore invariant under 3 independent translations. These are covered by the hyperbolic plane $\mathbb{H}^2$ [25] in such a way that the symmetries of the surface correspond to hyperbolic symmetries [19].

The above observations and ideas have led to a novel investigation of 3-dimensional Euclidean networks, where TPMS are used as a convenient route to the enumeration of crystallographic nets and polyhedra in $\mathbb{R}^3$ [32, 26, 31, 16, 30, 3]. The aim of the EPINET project [1] is to produce and analyse chemical structures by investigating how graphs embed on TPMS. Most hyperbolic in-surface symmetries of prominent TPMS manifest as ambient Euclidean symmetries of $\mathbb{R}^3$ [29], so that symmetric tilings of TPMS give rise to symmetric graph embeddings in $\mathbb{R}^3$. Not only does knowing the symmetry of a structure in $\mathbb{R}^3$ facilitate further investigations into structural properties, symmetric structures are prominent because they are candidates for structures found in nature as well as for synthesis, and because they favor self-essembly. Structures such as hyperbolic tilings with disks with kaleidoscopic symmetries generated by reflections [20, 30], some simple hyperbolic tilings with slightly more complicated symmetries [28, 27], simple unbounded tiles with a network-like structure [17, 12, 19, 21, 18, 8, 7], and unbounded tiles with totally geodesic boundaries [9]

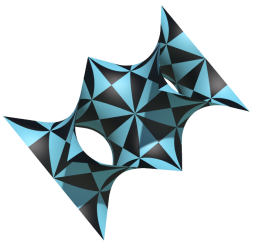
36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.



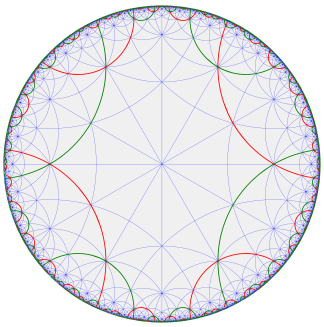
(a) Tesselation of $\mathbb{H}^2$ by dodecagons corresponding to the genus 3 Riemann surface that gives rise to the D-surface.



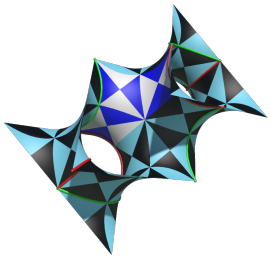
(b) The symmetries of the D-surface within a unit cell in $\mathbb{H}^2$. Each line represents a mirror symmetry.



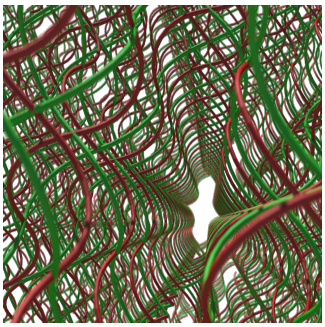
(c) A section of the D-surface in $\mathbb{R}^3$, together with its smallest asymmetric triangle patches.



(d) A tiling of the hyperbolic plane with symmetry group 22222, represented by the green and red edges, with a tiling by triangles with ⋆246 symmetry shown in blue in the background.



(e) The decoration from (d) and a fundamental domain shown as a decoration of the Diamond triply-periodic minimal surface. The triangles illustrate the symmetries of the surface.



(f) The resulting net in $\mathbb{R}^3$ when the tile boundaries are taken as trajectories in Euclidean 3-space.

Figure 1 (a)-(c) shows symmetries of the D-surface in $\mathbb{R}^3$ and its uniformization in $\mathbb{H}^2$. (d)-(f) shows the progression from a tiling of the hyperbolic plane to a 3-dimensional net via a decoration of the Diamond triply-periodic minimal surface.

have been explored, and resulting structures have been used in analysis in real physical systems [22].

The situation can be summarized as in Figure 1, which shows a hyperbolic tiling that respects the translational symmetries, indicated in (a) and (b), of the covering map of the hyperbolic plane onto a prominent TPMS, the diamond D surface, partly shown in (c). When the tile boundaries are considered as trajectories in 3-dimensional Euclidean space rather than curves on the surface, we obtain a symmetric net in $\mathbb{R}^3$. In (b) and (c), the symmetries of the D surface are illustrated by the tiling by triangles, with each triangle boundary corresponding to a reflection in the surface. This symmetry group can be denoted by ⋆246, using Conway's notation[5] for 2D orbifold symmetry groups.

1.1 Summary of the problem and results

We summarize the problem as detailed above. The EPINET project produces and analyses chemical structures by investigating how graphs embed symmetrically on triply periodic minimal surfaces (TPMS). The aim of this paper is to introduce techniques to systematically

enumerate symmetric embeddings of graphs on the gyroid, primitive, and diamond TPMSs, the most prominent and well understood examples of TPMS in nature. We will concentrate on symmetry groups generated by rotations and graphs which admit a 2-cell embedding into the surface in its smallest translational unit in $\mathbb{R}^3$.

Up until now, this problem has been investigated only for certain classes of simple graphs and attempts to further EPINET have involved ad-hoc ideas with limited applications. To organize the ensuing structures, the problem of a complexity ordering in the enumeration takes up an important role in our investigation and we base our ideas on an intuitive ordering. We seek a systematic approach to investigating graph embeddings on surfaces with a given symmetry group, in an attempt to develop a general framework to tackle similar problems in the future. To achieve these goals, we investigate tilings in $\mathbb{H}^2$ with a given symmetry group will be the focus of our interest.

Our results show that, in theory, the enumeration works. However, some of the algorithms used, for example those of computational group theory cannot easily handle the group presentations we work with.

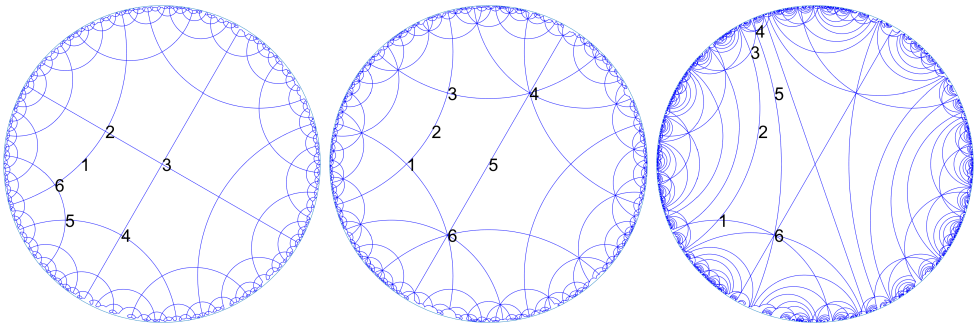
2 Preliminaries and our approach

▶ **Definition 2.1.** A tiling $\mathcal{T}$ of $\mathbb{H}^2$ is a locally finite collection of closed disks in $\mathbb{H}^2$ whose interiors are pairwise disjoint. Let $\mathcal{T}$ be a tiling of $\mathbb{H}^2$ and let Γ be a discrete group of isometries. If $\mathcal{T} = \gamma\mathcal{T} := \{\gamma T \mid T \in \mathcal{T}\}$ for all $\gamma \in \Gamma$ then we call the pair $(\mathcal{T}, \Gamma)$ an *equivariant tiling*. Two tiles $T_1, T_2 \in \mathcal{T}$ are *equivalent* or symmetry-related if there exists $\gamma \in \Gamma$ such that $\gamma T_1 = T_2$. The *orbit* of a tile is the subset of $\mathcal{T}$ given by images of T: $\Gamma.T = \{\gamma T \text{ for } \gamma \in \Gamma\}$. Given a particular tile $T \in \mathcal{T}$, the *stabilizer subgroup* Γ_T is the subgroup of Γ that fixes T, i.e. $\Gamma_T = \{\gamma \in \Gamma \mid \gamma T = T\}$. A tile is called *fundamental* if Γ_T is trivial and we call the whole tiling fundamental if this is true for all tiles. An equivariant tiling is called *tile-k-transitive*, when k is the number of equivalence classes (i.e. distinct orbits) of tile under the action of Γ. Two equivariant tilings $\{(\mathcal{T}_i, \Gamma_i)\}_{i=1}^2$ are **equivariantly equivalent** if there exists a homeomorphism φ such that $\varphi(\mathcal{T}_1) = \mathcal{T}_2$ and $\varphi\Gamma_1\varphi^{-1} = \Gamma_2$.

To simplify the discussion and avoid technical details, we shall assume in this paper that the edge orbits of a tiling are coloured and therefore distinguishable, in the sense that any nontrivial graph isomorphism of the tiling's 1-skeleton changes it. There is a complete invariant for equivariant equivalence classes of tilings, the D-symbol, a weighted graph, whose isomorphism class uniquely determines an equivariant equivalence class of tilings with closed disks in $\mathbb{H}^2$ [11]. There is likewise an algorithm that enumerates all (of the infinitely) possible equivariant equivalence classes of tilings starting from fundamental tile-1-transitive tilings. For a given TPMS, there is a symmetry group S that contains the largest group of translational symmetries T of the TPMS as a subgroup, i.e. $T \subset S$. We call a subgroup H of S satisfying $T \subset H \subset S$ *commensurate* with the symmetries of the TPMS.

▶ **Definition 2.2.** A (hyperbolic) orbifold, for our purposes, is the space $\mathbb{H}^2/\Gamma$, where Γ is a discrete group of isometries of $\mathbb{H}^2$. The orbifold structure is more than the quotient space $\mathbb{H}^2/\Gamma$ with quotient topology in that it keeps track of the data associated to the quotient map $\pi : \mathbb{H}^2 \to \mathbb{H}^2/\Gamma$, so that one can retains the information of how "copies" of $\mathbb{H}^2/\Gamma$ fit together to form $\mathbb{H}^2$. We denote orbifolds by their Conway orbifold symbol [5], a symbol that keeps track of the generators of the symmetry group Γ.

Two main observations lead us to a novel approach to tackle the problem.



(a) A transitive, fundamental equivariant tiling of $\mathbb{H}^2$ with symmetry group 2224.

(b) Another transitive, fundamental equivariant tiling of $\mathbb{H}^2$ with symmetry group 2224.

(c) A different placement of generators, producing a sheared version of the fundamental tiling.

Figure 2 Fundamental tilings with symmetry group 2224.

1. For a given commensurate symmetry group H of a TPMS, two equivariant tilings are equivalent iff they are related by a homeomorphism of $\mathbb{H}^2$ that induces an automorphism of H.
2. Given special sets of geometric generators for H, it is possible to describe graph embeddings in $\mathbb{H}^2$ that give rise to equivariant tilings with symmetry group H as combinatorial descriptions in terms of the generators, see Figure 2. The set of geometric generators we use directly corresponds to a set as given by the orbifold symbol of the symmetry group.

In Figure 2a, consider the rotations corresponding to the generators $r_1, ..., r_4$, with centers $c_1, ..., c_4 \in \mathbb{H}^2$, marked by 1 to 4. It is straightforward to see that the points with rotational symmetry on the polygon's boundary correspond clockwise, starting at c_1, to the points $c_1, c_2, c_3, c_4, r_4(c_3), r_1(c_2)$ and we find similar expressions for other *stellate orbifolds*, i.e. those with only rotations for generators.

Given any generators $r_1, ..., r_4$ for the discrete group of isometries 2224, this description of edges defines a fundamental tiling, in this case with geodesic edges, regardless of the generators placement in $\mathbb{H}^2$. For example, for the fundamental tiling of Figure 2b, the edges are given by hyperbolic lines connecting the points marked 1 to 6 in cyclic order, which correspond to the points $c_1, c_2, r_2(c_1), r_3^{-1}(c_4), c_3, c_4$, respectively. Figure 2c illustrates that this relation for the edges still holds in a sheared version of the fundamental tiling with symmetry group. Here, the sheared fundamental domain exhibits less symmetries of the other two tilings. Note that the symmetry group of all tilings in Figure 2 is identical, namely 2224.

It turns out that the generators of the symmetry group H that give rise to isotopically distinct tilings, in the sense that there is no deformation of one tiling that leads to the other while preserving the symmetries of H at every step, are in one-to-one correspondence with special automorphisms of H. These, on the other hand, are in one-to-one correspondence with the orbifold mapping class group MCG. For our purposes we can briefly define the MCG as follows. Consider the set of homeomorphisms $\{\varphi\}$ of $\mathbb{H}^2$ that satisfy $\varphi^{-1} H \varphi = H$, endow this set with the compact-open topology, and identify those that can be connected through a path of such homeomorphisms. It turns out that the MCG of an orbifold is exactly in one-to-one correspondence to the automorphisms that lead to isotopically distinct tilings. For stellate orbifolds, this is proven in [23]. For more general orbifolds, the statement can also be made precise and generalized, but this lies outside the scope of the this paper.

All in all, by the above, we transfer a problem in graph enumeration to the problem of

(a) Curves representing group elements and the half-twist, indicated in red.

(b) The resulting curves after twisting.

Figure 3 Half-twists

enumerating elements of the MCG. An enumeration of tilings with a given symmetry group follows the following steps. First, we find a commensurate symmetry group of a TPMS of interest. Then, we use the complexity ordering of D-symbols [6] to construct a representative of each equivariant tiling class. Subsequently, using a presentation of the MCG and an action on the generators of the symmetry group, we obtain a list of generators that give rise to an enumeration of distinct isotopy classes of these tilings with the same symmetry group. Using the Weierstrass parametrization for minimal surface along with Schwarz-Christoffel maps to deform hyperbolic polygonal domains into spherical ones one can, with analytic continuation, construct a direct map from $\mathbb{H}^2$ to the TPMS of interest. We briefly explain the MCG approach in the next sections.

3 The mapping class group and its action on sets of generators

To investigate the action of mapping class group elements on sets of generators, we use the following idea. Similarly to the classical setting for surfaces, it is possible to interpret group elements of symmetry groups as special curves in the underlying topological space of the orbifold. Given a representative homeomorphism of an element of the MCG, we apply it to the curves representing the generators of the symmetry group and read off the new curves, which in turn designate new group elements. We focus on the action of well-known generators for the MCG with geometric interpretations.

We consider an example. Figure 3 shows the derivation of the representation of a half-twist around two marked points, corresponding to two rotational center for rotations of the same order, in the orbifold's underlying topological space, with Figure 3b giving the result of the twist indicated in Figure 3a. In formulas, Figure 3 translates to the half twists around S_1 and S_2 taking the form

$$\begin{cases} S_1 \mapsto S_1 S_2 S_1^{-1}, \\ S_2 \mapsto S_1. \end{cases} \tag{1}$$

It is straightforward to check that the transformation (1) leaves the global group relation $S_1 S_2 \cdots S_n$ invariant.

One can draw similar pictures for more complicated symmetry groups, their MCGs and their generators. Using presentations of the mixed braid group [24] and exploiting the relationship between braid groups and MCGs [2, 10], one can derive a (lengthy) presentation of MCGs for stellate orbifolds. Using GAP(Groups, Algorithms, Programming) and the package KBMag, one can use this presentation to enumerate elements of the MCG, which together with the action on generators of the symmetry group and the description of the tilings in terms of these generators yields an enumeration of isotopy classes of (coloured) tilings of $\mathbb{H}^2$ with the given symmetry group.

4 Conclusion and open problems

By using MCGs of orbifolds, we were able to significantly generalize known results on the enumeration of hyperbolic tilings that fit onto hyperbolic surfaces. While we only showed how to proceed in case the orbifold is stellate, our methods remain valid in much more general settings. This work provides the basis for a major extension of the EPINET project and associated databases. The methods of this paper open up the possibility for automated searches for 3-periodic graphs in $\mathbb{R}^3$ with given topological properties that embed on TPMS.

Main difficulties in generalizing our approach lie within the fact that (semi-)algorithms like KBMag are not guaranteed to find an enumeration of MCG elements, and they are very sensitive to parameters. Furthermore, for general TPMS, for our methods to work, one first has to investigate commensurate symmetry groups, find a parametrization of the TPMS and lifts of in-surface symmetries and find the description of edges in terms of generators.

Acknowledgements. We would like to thank Stephen Hyde, Vanessa Robins, and Olaf Delgado-Friedrichs from the Australian National University for hosting the first author on a research stay, where some of this work was done.

References

1 Epinet. Accessed: 08-01-2020. URL: http://epinet.anu.edu.au.

2 Joan S. Birman. Mapping class groups and their relationship to braid groups. *Communications on Pure and Applied Mathematics*, 22(2):213–238, 1969. URL: http://dx.doi.org/10.1002/cpa.3160220206, doi:10.1002/cpa.3160220206.

3 Toen Castle, Myfanwy E. Evans, Stephen T. Hyde, Stuart Ramsden, and Vanessa Robins. Trading spaces: building three-dimensional nets from two-dimensional tilings. *Interface Focus*, 2(January):555–66, 2012. doi:10.1098/rsfs.2011.0115.

4 B. Chen, M. Eddaoudi, S.T. Hyde, M. O'Keeffe, and O. M. Yaghi. Interwoven metal-organic framework on a periodic minimal surface with extra-large pores. *Science*, 291:1021 – 994, 2001.

5 John H. Conway and Daniel H. Huson. The orbifold notation for two-dimensional groups. *Structural Chemistry*, 13(3-4):247–257, 2002. doi:10.1023/A:1015851621002.

6 Olaf Delgado-Friedrichs. Data structures and algorithms for tilings I. *Theoretical Computer Science*, 303(2-3):431–445, 2003. doi:10.1016/S0304-3975(02)00500-5.

7 Myfanwy E. Evans and Stephen T. Hyde. Periodic entanglement III: tangled degree-3 finite and layer net intergrowths from rare forests. *Acta Crystallographica Section A*, 71(6):599–611, Nov 2015. doi:10.1107/S2053273315014710.

8 Myfanwy E. Evans, Vanessa Robins, and Stephen T. Hyde. Periodic entanglement i: networks from hyperbolic reticulations. *Acta Crystallographica Section A*, 69(3):241–261, 2013. URL: http://dx.doi.org/10.1107/S0108767313001670, doi:10.1107/S0108767313001670.

9 Myfanwy E. Evans, Vanessa Robins, and Stephen T. Hyde. Periodic entanglement ii: weavings from hyperbolic line patterns. *Acta Crystallographica Section A*, 69(3):262–275, 2013. URL: http://dx.doi.org/10.1107/S0108767313001682, doi:10.1107/S0108767313001682.

10 Benson Farb and Dan Margalit. *A Primer on Mapping Class Groups (PMS-49)*. Princeton University Press, 2012. URL: http://www.jstor.org/stable/j.ctt7rkjw.

11 Daniel H. Huson. The generation and classification of tile-k-transitive tilings of the euclidean plane, the sphere and the hyperbolic plane. *Geometriae Dedicata*, 47(3):269–296, Sep 1993. doi:10.1007/BF01263661.

12 S. Hyde and S. Ramsden. Chemical frameworks and hyperbolic tilings. In P. Hansen, P. Fowler, and M. Zheng, editors, *Discrete Mathematical Chemistry*, pages 203–224. American Mathematical Society, 2000. doi:10.1090/dimacs/051/15.

13 S. T. Hyde. Hyperbolic surfaces in the solid-state and the structure of ZSM–5 zeolites. *Acta Chem Scand*, 45:860 – 863, 1991.

14 S. T. Hyde. Crystalline frameworks as hyperbolic films. In J.N. Boland and J. D. FitzGerald, editors, *Defects and processes in the solid state: Geoscience applications*. Elsevier, Amsterdam, 1993.

15 S. T. Hyde and S. Andersson. A systematic net description of saddle polyhedra and periodic minimal surfaces. *Z Kristallogr*, 168:221 – 254, 1984.

16 S. T. Hyde, O. Delgado Friedrichs, S. J. Ramsden, and V. Robins. Towards enumeration of crystalline frameworks: the 2D hyperbolic approach. *Solid State Sci*, 8:740 – 752, 2006.

17 S. T. Hyde and C. Oguey. From 2D hyperbolic forests to 3D Euclidean entangled thickets. *European Physical Journal B*, 16(4):613–630, 2000. doi:10.1007/PL00011063.

18 S. T. Hyde and S. Ramsden. Polycontinuous morphologies and interwoven helical networks. *EPL (Europhysics Letters)*, 50(2):135, 2000. URL: http://stacks.iop.org/0295-5075/50/i=2/a=135.

19 S. T. Hyde, S. Ramsden, T. Di Matteo, and J. J. Longdell. Ab-initio construction of some crystalline 3D Euclidean networks. *Solid State Sciences*, 5(1):35–45, 2003. doi:10.1016/S1293-2558(02)00079-1.

20 S. T. Hyde and S. J. Ramsden. Some novel three-dimensional euclidean crystalline networks derived from two-dimensional hyperbolic tilings. *Eur Phys J B*, 31:273 – 284, 2003.

21 Stephen T. Hyde, Ann Kristin Larsson, Tiziana Di Matteo, Stuart Ramsden, and Vanessa Robins. Meditation on an Engraving of Fricke and Klein (The Modular Group and Geometrical Chemistry). In *Australian Journal of Chemistry*, 2003. doi:10.1071/CH03191.

22 Jacob J K Kirkensgaard, Myfanwy E Evans, Liliana de Campo, and Stephen T Hyde. Hierarchical self-assembly of a striped gyroid formed by threaded chiral mesoscale networks. *Proceedings of the National Academy of Sciences of the United States of America*, 111(4):1271–6, 2014. URL: http://www.pnas.org/cgi/content/long/111/4/1271, doi:10.1073/pnas.1316348111.

23 C. Maclachlan and W. J. Harvey. On Mapping-Class Groups and Teichmuller Spaces. *Proceedings of the London Mathematical Society*, s3-30(4):496–512, 1975. URL: http://plms.oxfordjournals.org/cgi/doi/10.1112/plms/s3-30.4.496, doi:10.1112/plms/s3-30.4.496.

24 Sandro Manfredini. Some Subgroups of Artin's Braid Group. *Topology and its Applications*, 78:123–142, 1997.

25 William H. Meeks III. *The Theory of Triply Periodic Minimal Surfaces*. PhD thesis, University of California, Berkeley, 1975.

26 Reinhard Nesper and Stefano Leoni. On tilings and patterns on hyperbolic surfaces and their relation to structural chemistry. *ChemPhysChem*, 2(7):413–422, 2001. doi:10.1002/1439-7641(20010716)2:7<413::AID-CPHC413>3.0.CO;2-V.

27 Martin Cramer Pedersen, Olaf Delgado-friedrichs, and Stephen T Hyde. Surface embeddings of the Klein and the Mobius – Kantor graphs. *Acta Crystallographica Section A*, 74:223–232, 2018. doi:10.1107/S2053273318002036.

28 Martin Cramer Pedersen and Stephen T. Hyde. Polyhedra and packings from hyperbolic honeycombs. *Proceedings of the National Academy of Sciences*, 2018. URL: http://www.pnas.org/content/early/2018/06/19/1720307115, arXiv:http://www.pnas.org/content/early/2018/06/19/1720307115.full.pdf, doi:10.1073/pnas.1720307115.

29 Joaquín Pérez and Antonio Ros. *Properly embedded minimal surfaces with finite total curvature*, pages 15–66. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002. doi:10.1007/978-3-540-45609-4_2.

30 S. J. Ramsden, V. Robins, and S. T. Hyde. Three-dimensional Euclidean nets from two-dimensional hyperbolic tilings: kaleidoscopic examples. *Acta Crystallographica Section A*, 65(2):81–108, Mar 2009. doi:10.1107/S0108767308040592.

31 V. Robins, S. J. Ramsden, and S. T. Hyde. A note on the two symmetry-preserving covering maps of the gyroid minimal surface. *European Physical Journal B*, 48(1):107–111, 2005. doi:10.1140/epjb/e2005-00377-x.

32 J. -F Sadoc and J. Charvolin. Infinite periodic minimal surfaces and their crystallography in the hyperbolic plane. *Acta Crystallographica Section A*, 45(1):10–20, 1989. doi:10.1107/S0108767388008438.

Computing the cut distance of two curves*

Maike Buchin[1], Leonie Ryvkin[2], and Jérôme Urhausen[3]

1 **Faculty of Mathematics, Ruhr University Bochum**
`maike.buchin@rub.de`
2 **Faculty of Mathematics, Ruhr University Bochum**
`leonie.ryvkin@rub.de`
3 **Department of Information and Computing Sciences, Utrecht University**
`J.E.Urhausen@uu.nl`

Abstract

The recently introduced k-Fréchet distance extends the well-known Fréchet distance to objects of rearranged pieces. We focus on a variant of this, namely the cut distance, where the input curves are cut into subcurves, which are then matched regarding their similarity with respect to the weak Fréchet distance. It is NP-hard to decide the cut distance of two curves, however, we hereby present a polynomial-time algorithm in case the curves are only cut into two subcurves.

1 Introduction

Comparing geometric shapes is a topic of great interest as it comes up in many applications [4]. Two measures that have proven useful in many applications are the Hausdorff and the Fréchet distance. While the Hausdorff distance can be computed more efficiently, the Fréchet distance gives more information by taking into account how the curves are traversed.

The k-Fréchet distance bridges between Hausdorff and (weak) Fréchet distance. It comes in two variants: the cut and the cover variant. Here we consider the cut distance, which allows to cut a curve into a number of subcurves where the subcurves resemble each other in terms of the (weak) Fréchet distance. Thus it allows to find similarities between objects of rearranged pieces such as chemical structures or handwritten characters.

Characterizing these distance measures in the free space (defined below) shows that the k-Fréchet distance bridges between the (weak) Fréchet distance and Hausdorff distance (see below for details): the weak Fréchet distance can be characterized by one component in the free space projecting surjectively onto both parameter spaces, whereas the Hausdorff distance can be characterized by the free space projecting surjectively onto both parameters. For the cut distance, we need to find a subdivision of the free space in a (possibly non-uniform) grid with $k \times k$ cells such that we can choose exactly one cell per row and column that contains a component that surjectively projects onto the boundaries of this cell. The cells may but need not share a common corner.

Related work. Efficient algorithms for computing the Fréchet distance and the weak Fréchet distance were presented by Alt and Godau in 1995. They first introduced the concept of the free space diagram, which is key to computing this distance measure and its variants [3]. Following their work, numerous variants and extensions have been considered. Here we mention only those most related to our work. Alt, Knauer and Wenk [5] compared Hausdorff to Fréchet distance and showed that for convex closed curves Hausdorff distance equals Fréchet distance. For curves in one dimension Buchin et al. [6] proved equality of Hausdorff and weak Fréchet distance using the well-known Mountain climbing theorem [8].

* Jérôme Urhausen is supported by the Netherlands Organisation for Scientific Research under project 612.001.651

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

The k-Fréchet distance was first studied by Buchin and Ryvkin [7]. They showed that deciding the cut distance is NP-hard, and optimizing is even APX-hard. Later Akitaya et al. [2, 1] considered a second variant of the k-Fréchet distance, called cover distance. For this, the input curves are also divided into subcurves, which are to resemble each other in terms of the weak Fréchet distance. But in contrast to the cut distance, those subcurves are allowed to overlap. Deciding the cover distance is also NP-hard, but it can be approximated efficiently [1]. Algorithmically the cut distance is even more challenging than the cover distance: whereas the latter "only" asks for k components in free space that completely project onto both parameter spaces, the former additionally requires this without overlap.

Overview. First, in Section 2, we recall some necessary definitions and formally define the cut distance. In Section 3 we present a polynomial-time algorithm for the cut distance for $k = 2$, i.e., cutting both input curves into two pieces.

2 Definitions

The Fréchet distance [3], a well-known measure for curves, is defined as follows: For curves $P, Q\colon [0,1] \to \mathbb{R}^2$, the Fréchet distance is given by

$$\delta_{\mathrm{F}}(P,Q) = \inf_{\sigma,\tau} \max_{t\in[0,1]} \|P(\sigma(t)) - Q(\tau(t))\|,$$

where the reparametrisations $\sigma, \tau\colon [0,1] \to [0,1]$ range over all non-decreasing surjections. A variant is the weak Fréchet distance $\delta_{\mathrm{wF}}(P,Q)$, where both curves are re- parameterised by σ and τ, respectively, which range over all continuous surjections. All computation needs discrete input data, so we assume that curves are polygonal chains in the following.

A well-known characterisation, which is key to efficient algorithms for computing the (weak) Fréchet distance [3], uses the free space diagram. First we define the free space F_ε:

$$F_\varepsilon(P,Q) = \{(t_1,t_2) \in [0,1]^2\colon \|P(t_1) - Q(t_2)\| \leqslant \varepsilon\}.$$

The free space diagram puts this information into an $(n \times m)$-grid, where n, m are the numbers of segments in P and Q, respectively. Bottom and left boundary of the diagram correspond to the *parameter spaces* of the curves P and Q.

The Fréchet distance of two curves is at most a given value ε if there exists a monotone path through the free space connecting the bottom left to the top right corner of the diagram. For the weak Fréchet distance to be at most ε, we need a continuous path through the free space that connects the four boundaries. For the Hausdorff distance it holds that $\delta_{\mathrm{H}} \leq \varepsilon$ if there is a surjective projection of the free space onto both parameter spaces, see Figure 1.

We define further terms regarding the free space diagram below. A *component* is a maximal connected subset $c \subseteq F_\varepsilon(P,Q)$. Figure 2 shows a component ranging over two cells. A set S of (parts of) components *covers* a set $I \subseteq [0,1]_P$ of the parameter space (corresponding to the curve P) if I is a subset of the projection of S onto the above defined parameter space, i.e., $\forall x \in I\colon \exists c \in S, y \in [0,1]_Q\colon (x,y) \in c$. Covering on the second parameter space is defined analogously.

▶ **Definition 2.1.** For polygonal curves P, Q we define the cut version of the k-Fréchet distance (also called *cut distance* for short) as

$$\delta_{\mathrm{cut}}(k,P,Q) = \inf_{\sigma,\tau} \max_{t\in[0,1]} \|P(\sigma(t)) - Q(\tau(t))\|,$$

where now $\sigma, \tau\colon [0,1] \to [0,1]$ range over all continuous, surjective functions with $k' < k$ jump discontinuities, each.

That is, we cut the curves P and Q into at most k pieces, or subcurves, each, such that two resembling subcurves have small weak Fréchet distance. In the free space diagram, we insert the cuts on our curves as horizontal and vertical grid lines. For every row and column of this "cutting grid", we select exactly one cell. The cell corresponds to a matched pair of subcurves and hence needs to contain (a part of) a free space component that projects surjectively onto the cell's boundaries. The cover distance $\delta_{\text{cover}}(k, P, Q)$ is defined analogously, but now the pieces are allowed to overlap [2]. Intuitively, we ask for a selection of at most k components that cover the parameter spaces when using the cover distance.

For the decision problem we ask whether the weak Fréchet distance between pieces can be bounded by a given value ε, where the number of subcurves is upper bounded by k. For fixed ε, we want to minimize k (optimization problem). The value of the cut distance lies in between Hausdorff and weak Fréchet distance and is lower-bounded by the cover distance:

$$\delta_{\text{H}}(P, Q) \leq \delta_{\text{cover}}(k, P, Q) \leq \delta_{\text{cut}}(k, P, Q) \leq \delta_{\text{wF}}(P, Q).$$

Note that it holds that $\delta_{\text{cover}}(1, P, Q) = \delta_{\text{wF}}(P, Q)$ and $\delta_{\text{cover}}(n, P, Q) = \delta_{\text{H}}(P, Q)$. Figure 1 gives an example where cut and cover distance differ.

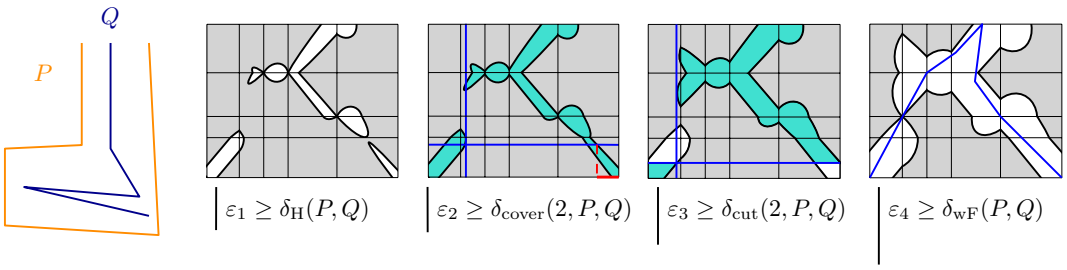


Figure 1 Comparison of distance measures. In the second diagram, two components are sufficient to cover the parameter spaces, but cutting does not work, because by choosing the bottom left and top right cell the red section on the bottom parameter space would not be covered.

3 Polyonomial Time Algorithm for k=2

As we already know that deciding the cut distance problem is NP- and optimizing k is APX-hard [7], we do not expect to solve it or approximate the number of cuts in polynomial time. Instead, we give a polynomial-time algorithm for $k = 2$. As we observe below, already for $k > 2$ placing cuts becomes more difficult and we leave this as an open problem. In the following we always assume that both curves have complexity n.

Cut placements The first difficulty is that there are infinitely many possibilities of placing a cut. For $k = 2$, we can reduce this amount to a finite set of discrete positions. We define *interesting points* to be local extrema of a component's boundary. We call a horizontal or vertical cut line through an interesting point an *interesting line*. We want cut lines ℓ and h such that there are two components covering opposing quadrants formed by ℓ and h. Such a placement of cut lines is called *valid*.

▶ **Theorem 3.1.** *If there exists a valid placement of cut lines in $F_\varepsilon(P, Q)$ for $k = 2$, it is possible to move these cut lines such that at least one of them becomes an interesting line.*

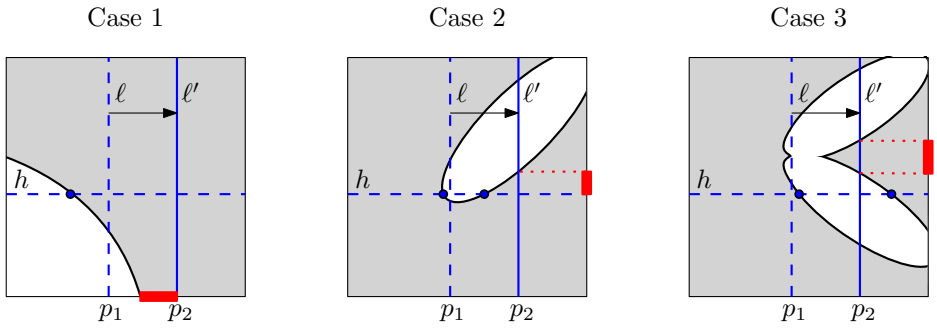


Figure 2 Given valid cut lines, moving one cut line beyond (new) interesting points may alter coverage of a quadrant. New interesting points n_i re marked with blue disks.

Proof. We are given a valid placement of cut lines and assume none of them features an interesting point. The cut lines subdivide the free space diagram into four quadrants of which two opposing ones are covered by components c_1 and c_2.

We fix the horizontal cut line h. It intersects c_1 and c_2 at *new interesting points* $n_1, \dots n_k$. Note that moving the vertical cut line to the right from ℓ through p_1 to ℓ' through p_2 can change the coverage in general, see Figure 2:

1. The left component might not cover a subinterval of $[p_1, p_2]$ on the curve P.
2. The right component might no longer cover an interval on the curve Q.
3. The right component might become disconnected.

These cases can only occur when there is a (new) interesting point between p_1 and p_2. Symmetrically, this holds for moving ℓ to the left, or moving the horizontal line h. Thus, as long as we do not move cut lines past (new) interesting points, the cut lines stay valid.

Let $I = [n_i, n_{i+1}]$ be the interval of adjacent new interesting points containing the given vertical cut line ℓ. If there is an interesting point $p \in I$, we move ℓ to the closest interesting point in I. Else we move ℓ to n_i. The intersection z of ℓ and h now lies on the boundary of a component. Next, we can move ℓ and h simultaneously such that z moves along the cell's boundary until one of the cut lines features an interesting point. ◀

Note that for $k > 2$, moving one cut line can make it necessary to move a second cut line, see Figure 3. This second cut line can cause the necessity to move a third, and so on.

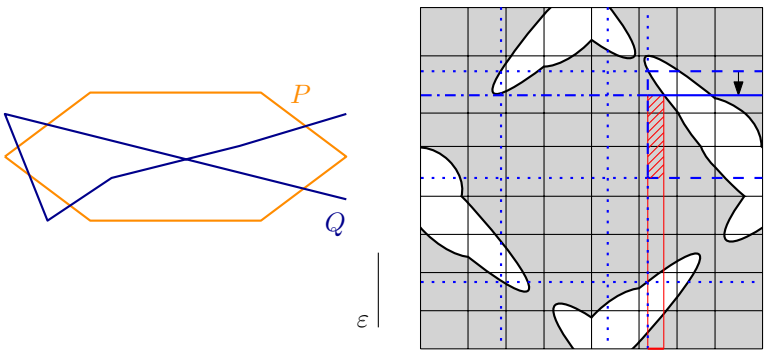


Figure 3 The dotted blue lines are validly placed cuts. The dashed parts enclose the cut cell we focus on: moving the top cut line downwards onto the dash dotted line (such that it coincides with the bottommost point of the upper left component) leaves part of the cut cell uncovered, thus its vertical cut line would need to be moved to the right. This process may repeat.

Algorithm First, we observe the following: The cuts induce a 2×2 grid on the free space diagram, and the cells featuring opposing corners are selected. Those selected cells each need to contain (a part of) one component that surjectively projects onto the cell's boundaries, and thus needs to touch two consecutive boundaries of the free space diagram. We call a component with this property a *candidate component*. As each of the $n \times n$ cells corresponding to a pair of segments contains no more than one cell, we can upper bound the number of candidate components by n. Note that for any pair of cut lines there exists at most one component per cell touching its cell's boundaries.

We give a brief overview of the algorithmic steps before going into detail. Note that we first run the algorithm assuming we select components in the bottom left and top right quadrant of F_ε and repeat steps 2-5 for the other option if the first run yields no solution. If there is still no solution, we repeat the whole process for horizontal interesting lines h instead of vertical interesting lines ℓ.

1. Compute the free space diagram $F_\varepsilon(P, Q)$ and its representative graph G;
2. Identify candidate components $A = \{a_1, a_2, \ldots, a_{|A|}\}$, that touch the bottom and the left boundaries of the free space, and $B = \{b_1, b_2, \ldots, b_{|B|}\}$, that touch the upper and the right boundaries. Sort A from left to right by their first intersection point with the bottom boundary and B from right to left by first intersection with the top boundary;
3. Compute all vertical interesting lines ℓ, i.e., vertical lines that are either tangent to a candidate component's boundary or run through an intersection of the component and the bottom or top boundary of the free space diagram. Store them in a sorted list;
4. For any line ℓ and components a_i, b_j we identify the parts of a_i and b_j that touch ℓ as well as the bottom and left or top and right boundaries, respectively. We call the parts of a_i and b_j fulfilling these requirements a_i^ℓ and b_j^ℓ. If there is no such a_i^ℓ and/or b_j^ℓ, delete ℓ from its list;
5. For each ℓ we recall a_i^ℓ and b_j^ℓ and do:
   a. Determine the topmost (bottommost) horizontal line $h_a^\uparrow$ ($h_b^\downarrow$) that intersects a_i^ℓ (b_j^ℓ);
   b. Determine the bottommost (topmost) horizontal line $h_a^\downarrow$ ($h_b^\uparrow$) such that the part of a_i^ℓ (b_j^ℓ) below (above) that line is still connected and touches ℓ;
   c. Any horizontal line through $[h_a^\downarrow, h_a^\uparrow] \cap [h_b^\downarrow, h_b^\uparrow]$ is a valid horizontal cut h.
6. In case no solution is found, repeat from step 2 for candidate components touching bottom and right or top and left boundary of the free space diagram, respectively;
7. In case no solution is found, repeat from step 3 for interesting horizontal lines h, compute left- and rightmost vertical lines in step 5.
8. If there is a solution, return ℓ and h, as well as the matching of the subcurves.

We first compute the free space diagram and the combinatorial graph G representing it in time $\mathcal{O}(n^2)$. A vertex of G corresponds to a cell or boundary of the diagram and an edge is drawn between vertices of neighboring cells iff the cells share a connected free space component, or between a boundary vertex and a cell vertex iff a component touches the boundary within that cell.

A depth first search (DFS) on G returns all candidate components and a vertical sweep of F_ε yields the interesting lines ℓ. We need $\mathcal{O}(n^2 \log n)$ time to find the candidate components since G has size $\mathcal{O}(n^2)$, and $\mathcal{O}(n^2)$ time to compute and sort all ℓ.

For each interesting line ℓ we then compute the correct pair of candidate components a_i^ℓ and b_j^ℓ as follows: By means of a breadth first search (BFS), where we limit the breadth to the current position of ℓ, we check whether a component touches the neighboring boundaries and ℓ. Most importantly, we ensure that the component is connected to the left (right) of ℓ

(in Figure 2, the dashed vertical cut line disconnects the component to the right of it). We need to check at most two candidates for each ℓ: considering them from left to right, the first interesting line has only one possible candidate, the first component. In the following, this candidate component either still intersects the next ℓ, or the next component in the sorted list has to be checked. We stop and continue the same BFS for all ℓ and all candidate components of a quadrant, and visiting each candidate once we only take quadratic time.

For each valid ℓ we perform a sweep to identify the interval in which the correct candidate components overlap vertically, and delete ℓ if they do not overlap. Within the overlap, any horizontal line is a valid cut placement, w.l.o.g. we choose the lower bound. There are $\mathcal{O}(n)$ extrema of a component's boundary curve to consider, so we take linear time here.

Overall, the runtime of the algorithm is bounded by $\mathcal{O}(n^2 \log n)$. Hence we conclude:

▶ **Theorem 3.2.** *The cut distance for $k = 2$ of two polygonal curves of complexity n can be decided in $\mathcal{O}(n^2 \log n)$ time.*

4 Conclusion

We presented a polynomial time algorithm for deciding the cut distance in case $k = 2$. For general k, we know that deciding the cut distance is NP-hard, and approximating the number of cuts k is APX-hard. For $k \geq 3$, we conjecture that cuts may have to be placed at non-interesting points, see Figure 4.

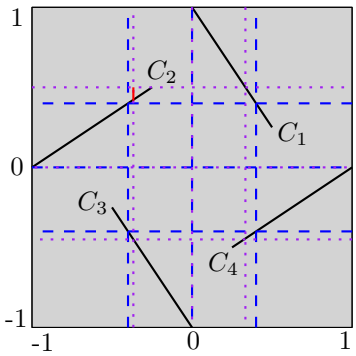


Figure 4 We are given four components $C_1, \ldots, C_4$ and apply four valid cut lines (dashed blue). The solution is unique and does not involve interesting points. The purple dotted lines feature the tip of C_2, but, as indicated by the red segment, C_2 does not cover its (purple) cell's boundaries.

References

1 Hugo Akitaya, Maike Buchin, Leonie Ryvkin, and Jérôme Urhausen. The k-Fréchet distance revisited and extended. In *35th European Workshop on Computational Geometry (EuroCG)*, page 7, 2019. URL: `http://www.eurocg2019.uu.nl/papers/41.pdf`.

2 Hugo Alves Akitaya, Maike Buchin, Leonie Ryvkin, and Jérôme Urhausen. The k-Fréchet distance: How to walk your dog while teleporting. In *ISAAC*, volume 149 of *LIPIcs*, pages 50:1–50:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

3 Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *Internat. J. Comput. Geom. Appl.*, 5(1-2):75–91, 1995.

4 Helmut Alt and Leonidas Guibas. Discrete geometric shapes: Matching, interpolation, and approximation: A survey. *Handbook of Computational Geometry*, 1997.

5 Helmut Alt, Christian Knauer, and Carola Wenk. Comparison of distance measures for planar curves. *Algorithmica*, 38(1):45–58, 2004.

6 Kevin Buchin, Maike Buchin, Christian Knauer, Günther Rote, and Carola Wenk. How difficult is it to walk the dog? In *Proc. 23rd Europ. Workshop on Comp. Geom.*, pages 170–173, 2007.

7 Maike Buchin and Leonie Ryvkin. The k-Fréchet distance of polygonal curves. In *34th European Workshop on Computational Geometry (EuroCG), Book of Abstracts*, page 4, 2018. URL: `conference.imp.fu-berlin.de/eurocg18/`.

8 Jacob E. Goodman, János Pach, and Chee-K. Yap. Mountain climbing, ladder moving, and the ring-width of a polygon. *Amer. Math. Monthly*, 96(6):494–510, 1989.

Tight Rectilinear Hulls of Simple Polygons

Annika Bonerath[1], Jan-Henrik Haunert[1], and Benjamin Niedermann[1]

1 **University of Bonn**
lastname@igg.uni-bonn.de

Abstract

A polygon is called $\mathcal{C}$-oriented if the orientations of all its edges stem from a pre-defined set $\mathcal{C}$. The schematization of a polygon is then a $\mathcal{C}$-oriented polygon that describes and simplifies the shape of the input polygon with respect to given hard and soft constraints. We study the case that the $\mathcal{C}$-oriented polygon needs to contain the input polygon such that it is tight in the sense that it cannot be shrunk without starting to overlap with the input polygon; we call this a *tight* $\mathcal{C}$*-hull* of the polygon. We restrict the tight $\mathcal{C}$-hull to be a simple polygon. We aim at a tight $\mathcal{C}$-hull that optimally balances the number of bends, the total edge length and the enclosed area. For the case that both polygons are rectilinear, we present a dynamic-programming approach that yields such a tight hull in polynomial time. For arbitrary simple polygons we can use the same approach to obtain approximate tight rectilinear hulls.

1 Introduction

Schematization has become a common tool for creating simplified visualizations of geometric objects such as paths, networks and regions. The purpose of this technique is to reduce the visual complexity of an object by describing its geometry based on a restricted and pre-defined set $\mathcal{C}$ of orientations. Most prominently, it is used for drawing maps of metro systems [10, 12], in which each edge is drawn either vertically, horizontally or diagonally; those maps became known as octilinear maps. An important core problem is the simplification of a polyline such that the result is $\mathcal{C}$*-oriented*, i.e., each edge of the resulting polyline has an orientation that stems from $\mathcal{C}$. Finding $\mathcal{C}$-oriented paths between two points in a polygon [1, 6, 9] or homotopic $\mathcal{C}$-oriented paths between obstacles [11] is closely related.

In this paper, we study the schematization of simple polygons, i.e., for a given simple polygon P we aim for a $\mathcal{C}$-oriented simple polygon Q that describes the shape of P with respect to pre-defined hard and soft constraints. For constructing $\mathcal{C}$-oriented polygons several approaches have been presented, e.g., [2, 3, 4, 5, 7, 8].

We present a novel approach for schematizing a given simple polygon P by a $\mathcal{C}$-oriented simple polygon Q. In contrast to previous work, we construct Q such that it encloses P. Further, Q should mimic the shape of P without having too many bends and without using unnecessarily much space; see Fig. 1. As application we have the schematization of plane graph drawings in mind whose outer faces we want to roughly sketch. We plan to use our approach for travel-time maps visualizing the reachable part within a road network (see Fig. 2) as well as for schematic representations of point sets. In the latter case the idea is to compute a planar graph representing a geometric spanner of the points and then to schematize the graph drawing.

We formalize the constraint that the original polygon P must be contained in the schematized polygon Q and mimics the shape of P in such a way that Q cannot be shrunk without intersecting P. More specifically, let Q and Q' be two simple polygons with edges $e_1, \ldots, e_n$ and $e'_1, \ldots, e'_n$, respectively. Further, let $\vec{v}_1, \ldots, \vec{v}_n$ and $\vec{v}'_1, \ldots, \vec{v}'_n$ be the vectors that describe the directions and lengths of $e_1, \ldots, e_n$ and $e'_1, \ldots, e'_n$, respectively. The

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

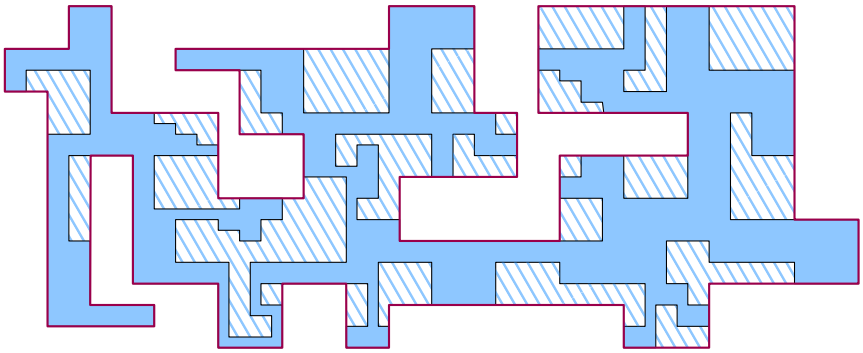
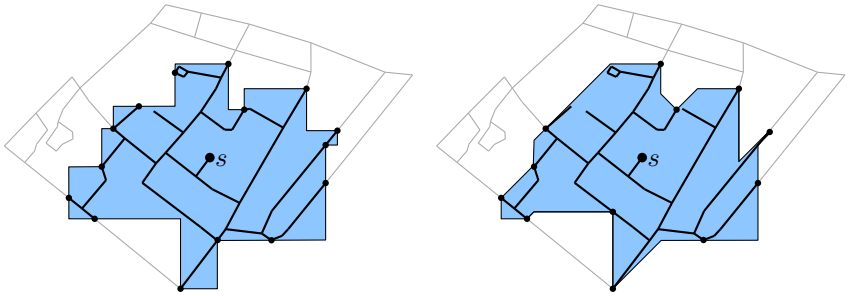


Figure 1 A rectilinear polygon P (blue) and a tight rectilinear hull of P (lilac).



(a) rectilinear tight hull **(b)** octilinear tight hull

Figure 2 A sketch of tight hulls enclosing the road network reachable from s within a given time. The input polygon is the outer face of the reachable sub-graph. We note that we can adapt the definition of tight hulls to also respect the non-reachable part.

polygon Q' is a *linear distortion* of Q if there are positive constants $c_1, \ldots, c_n$ such that $\vec{v}'_1 = c_1 \cdot \vec{v}_1, \ldots, \vec{v}'_n = c_n \cdot \vec{v}_n$, i.e., each edge of Q can be scaled and translated such that the polygon Q' results; see Fig. 3a. A simple polygon Q is a *tight hull* of another polygon P if Q contains P and there is no linear distortion of Q that lies in Q and contains P. We emphasize that a tight hull has no self-intersections. In case that edges of Q only use orientations from $\mathcal{C}$ we call Q a *tight $\mathcal{C}$-hull* of P. Altogether, we formalize the schematization problem as finding a tight $\mathcal{C}$-hull of P. In the special case that $\mathcal{C}$ only contains diagonal, vertical and horizontal orientations, we call Q a *tight octilinear hull* of P; see Fig. 3b. If it only contains vertical and horizontal orientations, we call Q a *tight rectilinear hull* of P; see Fig. 3c.

We aim at a tight $\mathcal{C}$-hull Q of P that is a good compromise between its edge length, its area and its number of bends, where a vertex is counted as bend if its incident edges have different orientations. More formally, for $\alpha = (\alpha_1, \alpha_2, \alpha_3)$ with $\alpha_i \geq 0$ we define the *cost* of Q as $\text{cost}(Q) = \alpha_1 \cdot \text{length}(Q) + \alpha_2 \cdot \text{area}(Q) + \alpha_3 \cdot \text{bends}(Q)$, where $\text{length}(Q)$ is the total edge length of Q, $\text{area}(Q)$ is the area of Q and $\text{bends}(Q)$ is the number of bends of Q. We call a tight $\mathcal{C}$-hull Q of P *α-optimal* if for any other tight $\mathcal{C}$-hull Q' of P we have $\text{cost}(Q') \geq \text{cost}(Q)$. Throughout the rest of this paper we study the special case in which we aim for a tight rectilinear hull Q of a rectilinear polygon P; see Fig. 3c. We use this fairly strong restriction to conduct a proof of concept for schematized tight hulls of polygons. Finally, we sketch how to use the approach for approximate tight hulls of not necessarily rectilinear polygons. We are currently extending our approach to more general settings, e.g., octilinear orientations as well as arbitrary polygons that are schematized.

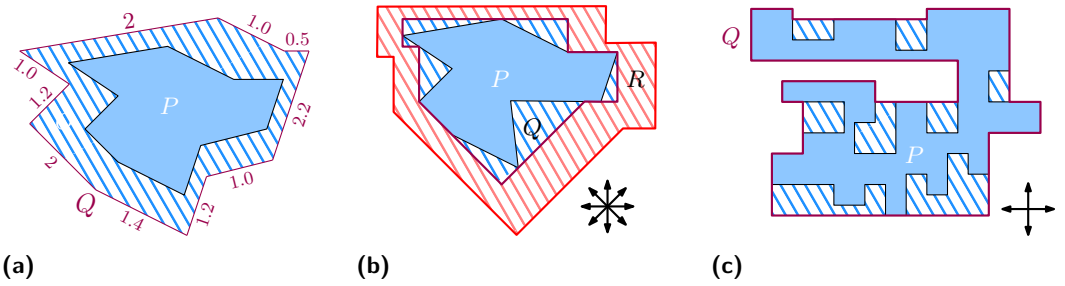


Figure 3 (a) The polygon Q is a linear distortion of the polygon P. For each edge of Q the according scaling factor is shown. (b) Q is a tight octilinear hull of P. The polygon R is not a tight hull of P, as Q is a linear distortion of R contained in R. (c) Q is a tight rectilinear hull of P.

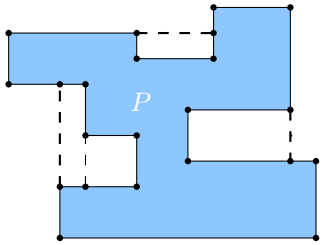


Figure 4 Example of a maximally subdivided polygon.

2 Structural Properties of Tight Rectilinear Hulls

Let P be a rectilinear polygon with n vertices and let Q be a tight rectilinear hull of P. We call a rectilinear polygon *maximally subdivided* if for each vertical and horizontal ray emanating from any vertex of P into its exterior the first contact point with P is again a vertex; see Fig. 4. In the remainder, we assume the input polygon P is maximally subdivided.

▶ **Lemma 2.1.** *Every vertex of Q on P is also a vertex of P.*

In the proof of Lemma 2.1 we assume that there is a vertex v of Q on P that is not a vertex of P; see Fig. 5. We show that this contradicts the assumptions that P is maximally subdivided (Fig. 5a) and Q is tight (see Fig. 5b–5c). Thus, Lemma 2.1 shows that we can build the solution based on the vertices of P. The next lemma shows that Q lies in the bounding box of P. The proof uses similar arguments as the proof of Lemma 2.1.

▶ **Lemma 2.2.** *The bounding box $\mathbb{B}$ of P is a tight rectilinear hull and any other tight rectilinear hull of P is contained in $\mathbb{B}$.*

In the following we describe how any tight rectilinear hull Q can be successively derived from the bounding box $\mathbb{B}$. Figuratively, this process can be understood as *carving* Q out of $\mathcal{B}$. More precisely, we obtain Q from $\mathbb{B}$ by successively refining the edges of $\mathbb{B}$ by replacing them with more and more complex polylines. As basic building block for this replacement procedure we use L-shaped polylines, which we call *bridges*. More specifically, a rectilinear polyline B is a *bridge* of P if B starts and ends at vertices of P and B can be partitioned into a prefix and a (possibly empty) suffix such that the edges of the prefix have the same orientation and the edges of the suffix have the same orientation. Hence, each bridge corresponds to a line segment or two incident line segments forming an "L"; see Fig. 6. The region enclosed by B and the polyline of P connecting the same vertices as B is the *bag* of B. We observe that B may consist of multiple regions and have multiple edges with P in common; see Fig. 6c.

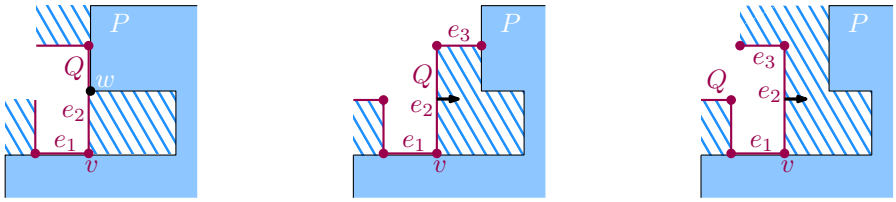


Figure 5 Illustration of the proof of Lemma 2.1. (a) At the end point of e_1 the polygon P has a vertex. (b)–(c) The edges e_1 and e_3 can be scaled such that Q shrinks but contains P.

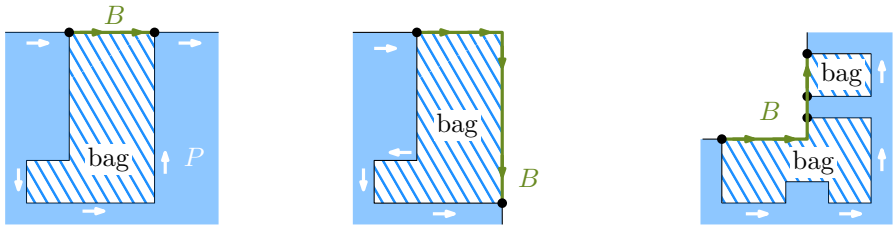


Figure 6 Examples of rectlinear polylines (green) forming bridges of P (blue).

▶ **Lemma 2.3.** *Every tight rectilinear hull of P can be partitioned into a sequence of bridges.*

The bounding box $\mathbb{B}$ of P can be partitioned into four bridges B_1, B_2, B_3 and B_4 such that they contain the top-left, top-right, bottom-right and bottom-left vertices of $\mathbb{B}$, respectively; see Fig. 7. The starting and end points of the four bridges lie on Q such that they split Q into four polylines Q_1, Q_2, Q_3 and Q_4 that are contained in the bags of B_1, B_2, B_3 and B_4, respectively. Our approach is based on the idea that each bridge B_i defines a sub-instance I_i that is solved independently from the others. The sub-instance I_i is defined by B_i and its bag; see Fig. 7c.

We now sketch a recursive procedure that creates Q_i from B_i. In general we can describe this setting by a bridge B that contains a subpath H of Q_i; when the recursion starts we have $B = B_i$ and $H = Q_i$. In the base case of the recursion the bridge B equals H. In the general case we recursively describe H by bridges; see Fig. 8. More specifically for B we can find up to three connected bridges C_1, C_2, and C_3 in the bag of B such that the polyline that is defined by these bridges connects the start and end point of B. Each bridge C_j forms a geometrically independent instance, i.e., the bridges C_1, C_2, and C_3 have pairwise disjoint bags. Further, the end points of C_1, C_2, and C_3 partition H into three subpaths H_1, H_2 and H_3 that lie in the bags of C_1, C_2, and C_3, respectively. Hence, the three bridges C_1, C_2, and C_3 partition the bag of B into smaller sub-instances defined by C_1, C_2, and C_3 containing the paths H_1, H_2 und H_3, respectively.

This provides us with the possibility of recursively describing Q_i; Figure 9 shows the recursion tree T for B_1 and Q_1 of the polygon presented in Fig. 7. We call T the *derivation tree* of B_1 and Q_1 the *derivative path* of B_1. In general we show the following theorem.

▶ **Theorem 2.4.** *For every bridge B and every path H of bridges that is contained in the bag of B and connects the start and end point of B, there is a derivation tree T_B such that H is the derivative path of T_B.*

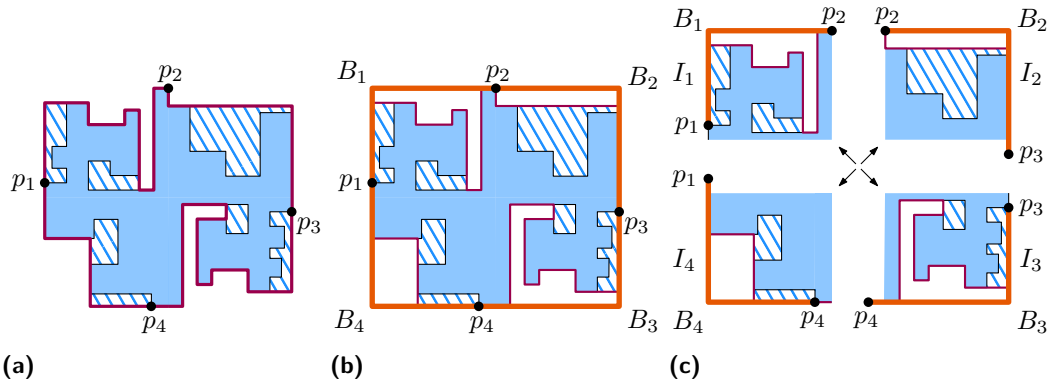


Figure 7 (a) A rectilinear polygon P (blue) and a tight rectilinear hull of P (lilac). (b) The bounding box of P is partitioned into the four bridges B_1, B_2, B_3 and B_4. (c) Each bridge defines an instance that is considered independently from the other instances.

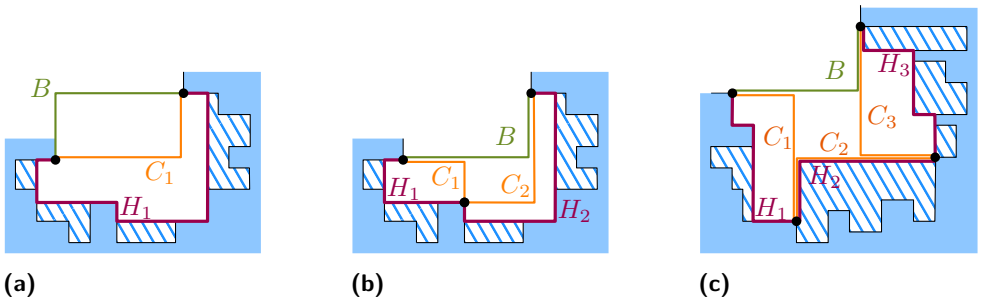


Figure 8 The bag of the bridge B defines an sub-instance. Further, there is a sub-path H (lilac) of Q that goes through the bag of B. If B is not part of Q, we can construct up to three bridges C_1, C_2 and C_3 whose bag form three geometrically independent instances that partition H.

To prove Theorem 2.4 we distinguish nineteen geometrical settings of the bridge B and the path H. We use six different methods for the construction of the child nodes $C_1, \ldots, C_k$ with $1 \leq k \leq 3$; see Fig. 10. We can show for each construction that the path H can be split into subpaths $H_1, \ldots, H_k$ so that each H_j with $1 \leq j \leq k$ is contained in the bag of C_j. For example, we use Construction M in the case that B and P share more than two vertices; see Fig. 10. In that case, we insert two child nodes for B in T_B containing the bridges C_1 and C_2, where C_1 is the path of B from the beginning to the first shared vertex u with P and C_2 contains the remaining part. We show that if we split H at u into subpaths H_1 and H_2, the path H_1 is contained in the bag of C_1 and the path H_2 is contained in the bag of C_2. The Constructions A-E assume that B shares exactly two vertices with P, and yield bridges $C_1, \ldots, C_k$ that not only lie on B but also in the interior of the bag of B without crossing H.

The constructions are more general than necessary such that they also work for any rectilinear polygon Q that consists of bridges of P. We conjecture that when exploiting the tightness only two children per node is sufficient, which later on would lead to an improvement of the running time by a linear factor. However, at latest when generalizing the result to the case that P is not rectilinear, we can show that three children are necessary.

Altogether, due to the construction of the decomposition tree its derivative path H does not intersect itself. In particular, two bridges B_1 and B_3 that intersect as shown in Fig. 11 can not belong to the same decomposition tree as neither one bag contains the other nor their bags are disjoint.

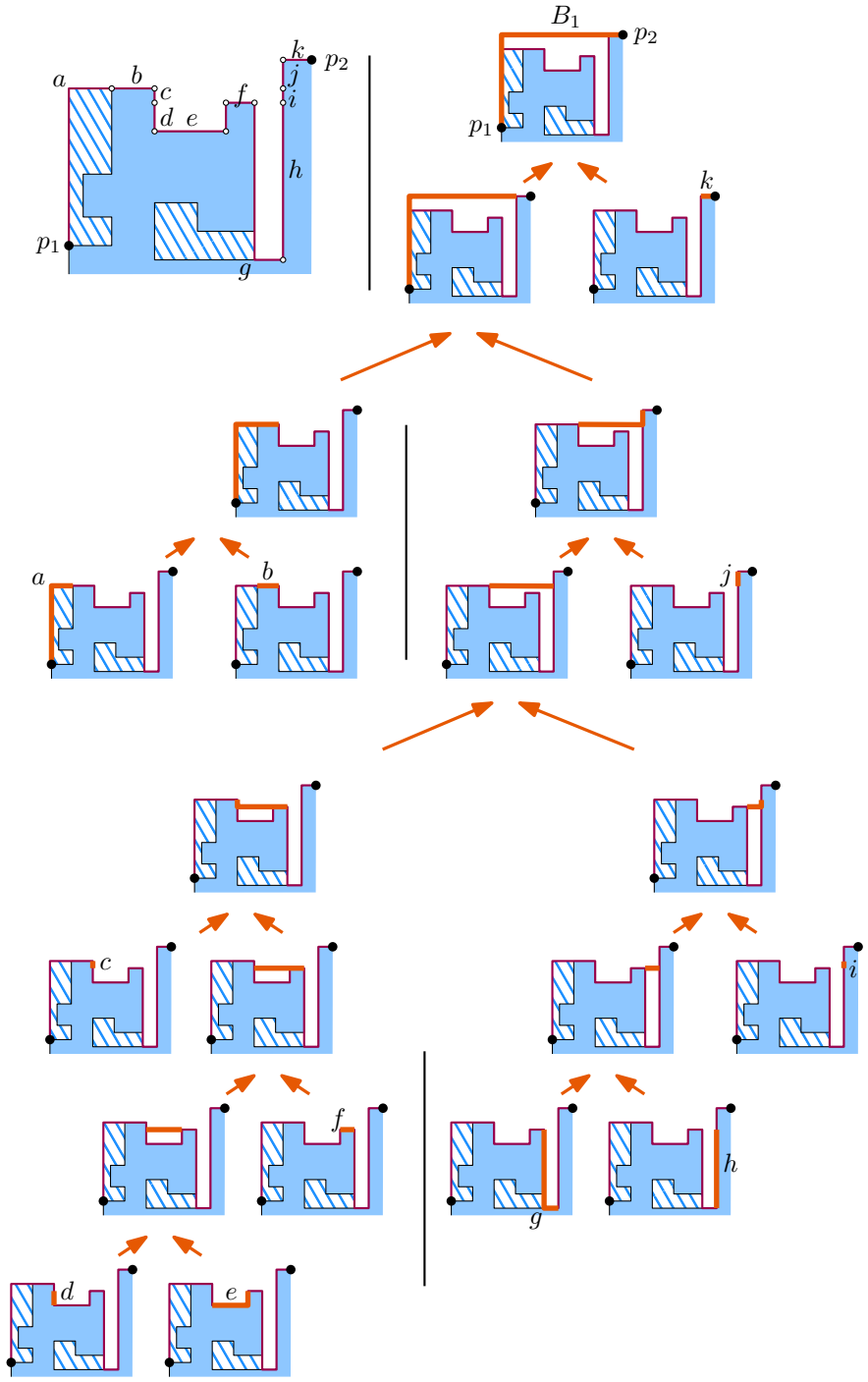


Figure 9 A recursion tree for the top-left part of the polygon shown in Fig. 7. On each level the bags of the bridges (orange) form geometrically independent sub-instances that are solved independently. Composing the bridges of the child nodes yields a path that connects the starting with the end point of the bridge of the parent node. Collecting the bridges of the leaves in pre-order yields the path Q_1 (lilac), which is part of Q.

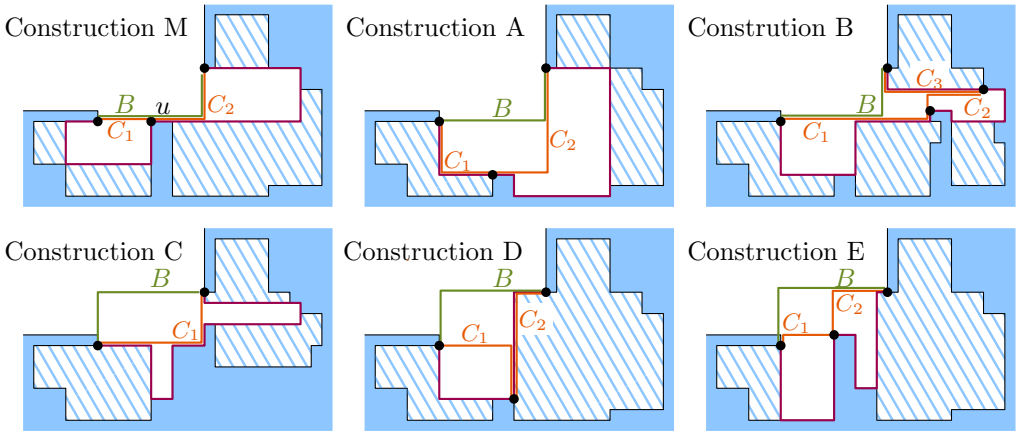


Figure 10 Construction types for the proof of Theorem 2.4.

3 Algorithm for Tight Rectilinear Hulls

We present an algorithm that consists of three steps. In the first step, we build an orthogonal grid $\mathcal{G}$ based on the vertices of P such that $\mathcal{G}$ lies in the interior of $\mathbb{B}$ and the exterior of P; see Fig. 12a. In the second step, we create the set $\mathcal{B}$ of all valid bridges based on $\mathcal{G}$ using depth-first searches; see Fig. 12b. In the third step, we compute an α-optimal tight rectilinear hull Q of P as follows. We split the bounding box $\mathbb{B}$ into the four bridges B_1, B_2, B_3 and B_4 as described in Section 2. These bridges split Q into four paths Q_i contained in the bags of B_i (with $1 \leq i \leq n$), respectively. We compute each Q_i by constructing its derivation tree T_i over $\mathcal{B}$ using dynamic programming. We finally assemble Q_i to Q. From a technical point of view we need to take special care about correctly accounting for the bends at the vertices connecting two sub-instances. We prove that the dynamic programming approach, which is the most time consuming part of the algorithm, needs $O(n^4)$ time and $O(n^2)$ space.

▶ **Theorem 3.1.** *The α-optimal tight rectilinear hull of a rectilinear polygon P can be computed in $O(n^4)$ time and $O(n^2)$ space.*

We observe that our approach is only based on the bridges that we compute using the grid $\mathcal{G}$. On that account a simple approach to support arbitrary simple polygons is discretizing P by subdividing each edge of P with additional vertices; see Fig. 13. We then build $\mathcal{G}$ based on the new and old vertices of P. As one can show, the result is a (not necessarily α-optimal) tight hull of P. Depending on the desired quality, we choose the degree of discretization.

4 Conclusion

We have introduced the concept of tight hulls of polygons. In contrast to previous schematization techniques, we require that the input polygon is contained in the schematization. We

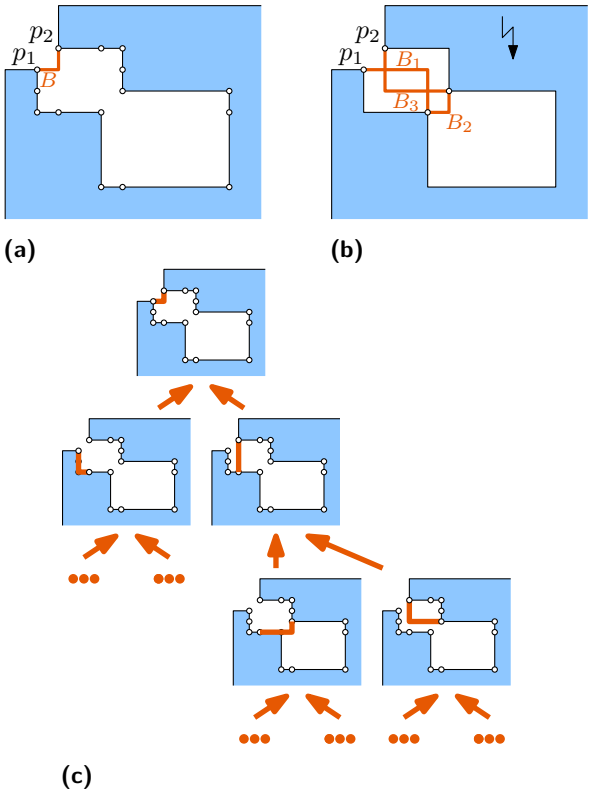


Figure 11 Decompositions of a bridge B. (a) The bridge B. (b) A decomposition of B into three bridges B_1, B_2 and B_3 such that B_1 and B_3 intersect. Such decompositions are excluded from the decomposition tree by construction. (c) A valid decomposition tree for B.

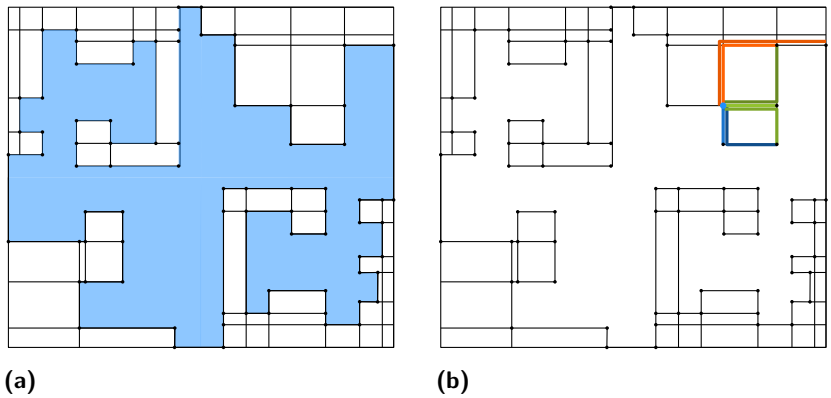


Figure 12 Step 1 and Step 2 of the algorithm. (a) The grid $\mathcal{G}$ in the exterior of P is created based on the vertices of P. (b) For each vertex of P all possible bridges to its successors are created.

have undertaken a proof of concept for rectilinear polygons and tight rectilinear hulls sketching a generic algorithm based on a dynamic programming approach. For simple polygons our approach yields approximate tight hulls. We are currently extending the algorithm to tight octilinear hulls as well as to α-optimal tight hulls of general simple polygons.

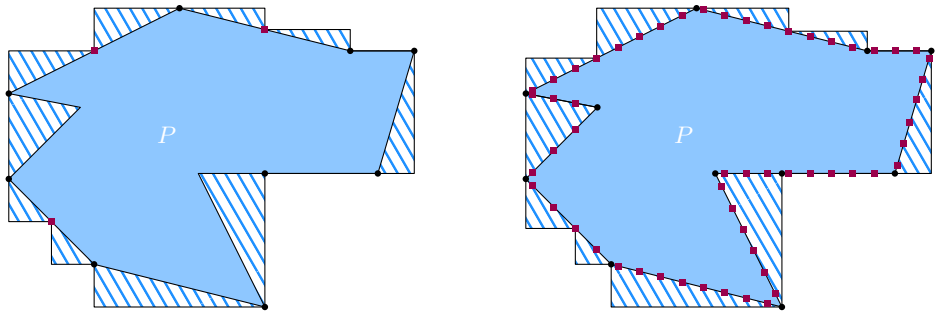


Figure 13 Tight rectilinear hulls of a simple maximal subdivided polygon P (vertices of P are black points). (a) Lemma 2.1 is not true any more as Q has fixed vertices (lilac squares) that are not vertices of P. (b) The tight hull of P is based on the vertices of P and additional vertices (lilac squares) subdividing the edges of P.

Acknowledgments. This work was partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC 2070 – 390732324.

References

1 John Adegeest, Mark Overmars, and Jack Snoeyink. Minimum-link c-oriented paths: Single-source queries. *International Journal of Computational Geometry & Applications*, 04(01):39–51, 1994. doi:10.1142/S0218195994000045.

2 Annika Bonerath, Benjamin Niedermann, and Jan-Henrik Haunert. Retrieving alpha-Shapes and Schematic Polygonal Approximations for Sets of Points within Queried Temporal Ranges. In *Proc. 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL '19)*, pages 249–258, 2019. doi:10.1145/3347146.3359087.

3 Quirijn W. Bouts, Irina Kostitsyna, Marc van Kreveld, Wouter Meulemans, Willem Sonke, and Kevin Verbeek. Mapping Polygons to the Grid with Small Hausdorff and Fréchet Distance. In Piotr Sankowski and Christos Zaroliagis, editors, *24th Annual European Symposium on Algorithms (ESA 2016)*, volume 57 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 22:1–22:16. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.ESA.2016.22.

4 Kevin Buchin, Wouter Meulemans, André Van Renssen, and Bettina Speckmann. Area-preserving simplification and schematization of polygonal subdivisions. *ACM Transactions on Spatial Algorithms and Systems*, 2(1):1–36, 2016. doi:10.1145/2818373.

5 Jan-Henrik Haunert and Alexander Wolff. Optimal and topologically safe simplification of building footprints. In *Proc. 18th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL GIS '10)*, pages 192–201, 2010. doi:10.1145/1869790.1869819.

6 Der-Tasi Lee, Chungdo Yang, and Chak Kuen Wong. Rectilinear paths among rectilinear obstacles. *Discrete Applied Mathematics*, 70(3):185 – 215, 1996. doi:10.1016/0166-218X(96)80467-7.

7 Maarten Löffler and Wouter Meulemans. Discretized approaches to schematization. In *29th Canadian Conference on Computational Geometry (CCCG)*, pages 220–225, 2017.

8 Wouter Meulemans, André van Renssen, and Bettina Speckmann. Area-preserving subdivision schematization. In Sara Irina Fabrikant, Tumasch Reichenbacher, Marc van Kreveld,

and Christoph Schlieder, editors, *Geographic Information Science*, pages 160–174. Springer Berlin Heidelberg, 2010.

9 Joseph S.B. Mitchell, Valentin Polishchuk, and Mikko Sysikaski. Minimum-link paths revisited. *Computational Geometry*, 47(6):651 – 667, 2014. doi:10.1016/j.comgeo.2013.12.005.

10 Martin Nöllenburg. A survey on automated metro map layout methods. In *Schematic Mapping Workshop 2014*, 2014.

11 Bettina Speckmann and Kevin Verbeek. Homotopic c-oriented routing with few links and thick edges. *Computational Geometry*, 67:11–28, 2018. doi:10.1016/j.comgeo.2017.10.005.

12 Hsiang-Yun Wu, Benjamin Niedermann, Shigeo Takahashi, and Martin Nöllenburg. A survey on computing schematic network maps: The challenge to interactivity. In *Schematic Mapping Workshop 2019*, 2019.

Approximating the Packing of Unit Disks into Simple Containers*

Helmut Alt and Nadja Seiferth

Institute for Computer Science, Freie Universität Berlin
lastname@mi.fu-berlin.de

Abstract

We study the following problems: Given a triangle or parallelogram, how many unit-disks can be packed without overlap into it? It is not known whether these problems are NP-hard or in NP. We give the first results on their approximability and therefore a better understanding of their complexity. In the case that all inner angles are bounded from below by a constant, we give a PTAS for each problem. For the case of arbitrarily small inner angles, we give an algorithm with constant-factor approximation and polynomial running time for triangles.

1 Introduction

Efficient disjoint packing of disks and spheres has been considered a natural and challenging problem in geometry for centuries. Kepler's famous conjecture about the most dense packing of spheres in three-dimensional space has been proven after nearly three hundred years by Hales and his group [5]. The densest packing of unit radius disks in the plane being the hexagonal grid has been known before. Earliest attempts for a proof go back to Lagrange although a rigorous proof was eventually given by Fejes-Tóth [4].

Algorithmic questions arise if instead of the whole space certain finite *containers* and sets of spheres or disks are considered. In the web, data bases can be found of the smallest containers (circles, squares etc.) which are known to hold n unit disks for n from 1 to several thousands [1].

We consider the converse problem of packing a maximal number of unit disks into a given finite size container. Even for the simplest container shapes such as disks or squares the problem is neither known to be solvable in polynomial time nor to be NP-hard. One major problem is that the number of disks to be computed can be exponential in the input size.

In a previous article, we developed PTAS for circular and square containers [2]. A natural representation for more general shapes would be simple or convex polygons. This paper is a small first step in this direction, showing a constant factor approximation algorithm for triangles. We can find PTAS for triangles and parallelograms if the containers are *fat*, i.e., their smallest angle is bounded from below by a constant.

2 First Order Theory of the Reals

We first observe that the decision problem, whether n unit circles can be packed into some container C is *decidable* if the shape of C can be described by finitely many polynomial inequalities. In fact, in this case the problem can be described by a formula in the first order theory of the reals. This formula contains $2n$ variables $s_1, t_1, ..., s_n, t_n$ which represent

* Supported by the German Science Foundation within the collaborative DACH project *Arrangements and Drawings* as DFG Project MU3501/3-1.

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

the coordinates where to place the centers of the unit circles. Moreover, the following conditions have to be stated in the formula:

1. all unit circles lie completely inside C
2. no two unit circles may intersect

Clearly, if C has a constant size description, condition 1 is the conjunction of $O(n)$ and condition 2 the conjunction of $O(n^2)$ formulas of constant size. The final formula is obtained by putting $\exists s_1, t_1, ..., s_n, t_n$ in front of these conditions.

The truth of first order formulas of the reals was shown to be decidable by Tarski in the 1950s. The fastest algorithm for this task is given by Basu et al. [3], Theorem 1.3.2. Plugging in the parameters of our problem, we obtain that the number of arithmetic operations to solve the problem is $n^{O(n)}$.

In order to determine the maximum number n_{max} of unit disks that can be packed into C we can make an exponential and binary search for n_{max} involving the decision algorithm in each of the $\log(n_{max})$ steps. If n_{max} is bounded by a constant we obtain:

▶ **Lemma 2.1.** *Suppose a container C described by constantly many polynomial inequalities is given by the coefficients of these polynomials, which are are rational numbers in binary notation. Then, if the maximum number n_{max} of unit circles that can be packed into C is bounded by a constant, it can be computed in polynomial time.*

It should be mentioned that the runtime of the algorithm is exponential in n_{max}, however.

3 Fat Containers

3.1 Parallelograms

We first consider the construction of a PTAS for determining the number of unit disks that can be packed into a *fat* parallelogram P. "Fat" means that the smaller angle γ of P is bounded from below by some constant γ_0. We will assume without loss of generality that two sides of P are parallel to the x-axis. Let a be the length of these sides. The input to the algorithm is a and (b_x, b_y) which are the coordinates of the left upper vertex of P, assuming that the left lower vertex of P is at the origin. These values are represented by rational numbers, where enumerator and denominator are given as binary integers.

Our **algorithm** has some similarity to the general technique by Hochbaum and Maass [6]. It subdivides P by lines parallel to its sides into congruent rhombi whose side length is some constant w to be determined later, see Figure 1. Since the rhombi will be truncated at the upper and right boundaries of P we get (up to) four different shapes of cells into which P is subdivided. For each of the shapes 1,..,4 we can determine the maximum number of unit disks it can hold in polynomial time using the algorithm given in Section 2.

Finally, we multiply the numbers obtained with the number of occurrences of the corresponding shapes (which can be computed by standard arithmetic), sum up, and return the obtained value as an approximation for the maximum number of unit disks that can be packed into P.

It remains to be shown that this algorithm is a PTAS for our problem if the input parallelogram P is fat. More precisely, for each $\varepsilon > 0$ it is possible to find a value for the parameter w such that the number n_{app} returned is at least $(1 - \varepsilon)n_{max}$ where n_{max} is the maximum number of unit disks that can be packed into P.

To see this we consider an optimal solution OPT and assign its disks to the cells of the decomposition as follows: If a disk is contained in a cell, assign it to that cell. If it intersects only the horizontal boundary of cells, assign it to the lower cell that it intersects. If it

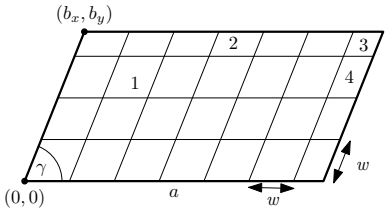


Figure 1 Parallelogram divided into cells defined by rhombus-grid with side length w.

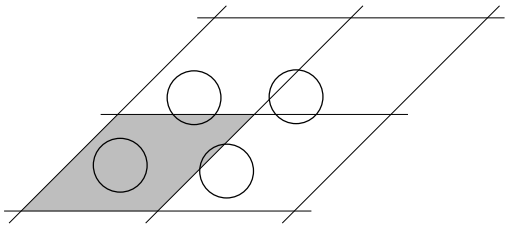


Figure 2 Disks assigned to a cell.

intersects only the vertical boundary of cells, assign it to the leftmost cell that it intersects. If it intersects a horizontal and a vertical grid-line, assign it to the cell at the lower left of the intersection of these two grid-lines, see Figure 2. In other words, all disks completely lying inside a cell q extended by a strip of width 2 to the top and to the right are assigned to q. For a cell q let n_i^q be the number of disks completely contained in q and n_b^q the disks assigned to it but not completely contained in it.

All n_i^q disks that are put completely inside a decomposition cell q by OPT are accounted for in n_{app} since there the maximum number n_m fitting into q is taken. However, the n_b^q disks assigned to but not completely contained in q are disregarded. Each of these disks must completely lie in a strip of width 4 around q. Since this strip has area $O(w)$ if the parallelogram P is fat we have $n_b^q = O(w)$.

On the other hand, Figure 3 shows that if $w \geq 2$ then $n_m \geq \lfloor (w \sin\gamma)/2 \rfloor^2 = \Omega(w^2 \sin^2\gamma)$ which is $\Omega(w^2)$ if P is fat. So we have $n_b^q / n_i^q \leq c/w$ for some constant c and for each cell q. Summing over all cells q we have

$$n_{max} = \sum_q (n_b^q + n_i^q) \leq (1 + c/w) \sum_q n_i^q \leq (1 + c/w) n_{app}$$

Choosing $w \geq d/\varepsilon$ for a suitable constant d yields:

▶ **Theorem 3.1.** *The algorithm described is a PTAS for the problem of finding the maximum number of unit disks that can be packed into a fat parallelogram P.*

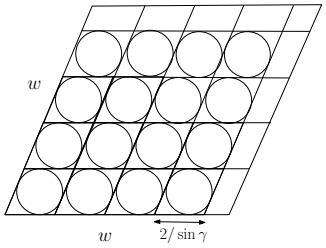


Figure 3 Lower Bound on the number of unit disks that can be packed into q

3.2 Triangles

Next, we will develop a PTAS for fat triangles, i.e., all of their angles are bounded from below by some constant α_0. Suppose that a triangle T is given by the lengths of its sides a, b, c with $a \geq b \geq c$. Side a is, without loss of generality, parallel to the x-axis and has its left vertex at the origin. Similarly to the case of parallelograms we decompose T into constant size cells by sets of lines which are parallel to one of two chosen sides a, b of T, see Figure 4. The distance of these lines is chosen such that both sides are divided into an equal

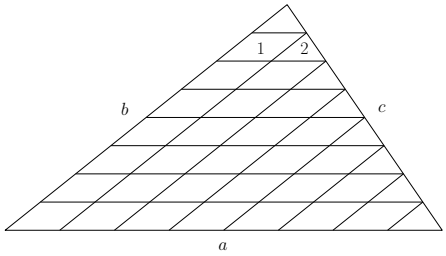


Figure 4 Dividing a triangle into cells with a parallelogram-grid with side lengths $\frac{a}{g}, \frac{b}{g}$ yields only two cell-types. Here $g = 8$.

number g of parts. As a consequence, there are only two types of congruent cells, which are parallelograms and triangles. As before, our algorithm computes the exact maximum number of disks fitting into each cell type by the technique of Section 2, multiplies it with the number of occurrences of the cell type and returns the sum of these numbers.

Analogousy to the corresponding proof for parallelograms, assigning disks of the optimal solution to the cells, it can be shown that for a given $\varepsilon > 0$ there is a choice of g, namely proportional to $b \cdot \varepsilon$ such that our algorithm computes a number n_{app} which is at least $(1 - \varepsilon)n_{max}$, i.e., we have:

▶ **Theorem 3.2.** *The algorithm described is a PTAS for the problem of finding the maximum number of unit disks that can be packed into a fat triangle T.*

4 Arbitrary Triangular Containers

For arbitrary triangular containers T, we can still construct in polynomial time a constant factor approximation for the maximum number n_{max} of unit disks that can be packed into T.

As before, let a be the longest side of T (we will also use a for the length of a), suppose again that it is parallel to the x-axis with its left vertex being the origin. Let h be the height of T perpendicular to a.

The idea of the algorithm is as follows: We first check whether $n_{max} = 0$ or $n_{max} = 1$. If not we imagine decomposing T into slabs of height 2 by lines parallel to a. Then the widths of the slabs are $s_1, ..., s_k$ where $s_i = a - 2i \cdot a/h$ for $i = 0, \cdots k$, see Figure 5.

Let k be the largest integer for which $s_k \geq 2$, namely $k = \lfloor (a-2)h/(2a) \rfloor$. In each slab the rectangle where the height of the slab is equal to 2 is filled contiguously with unit disks starting from the left end. Observe, that we cannot do this construction explicitely since the number of disks involved is exponential in the (bit-)size of the input.

However, the total number of disks packed can be computed, which is

$$\sum_{i=1}^{k} \lfloor s_i/2 \rfloor > \sum_{i=1}^{k} (s_i/2 - 1) > \lfloor ka/2 - ak(k+1)/(2h) - k \rfloor$$

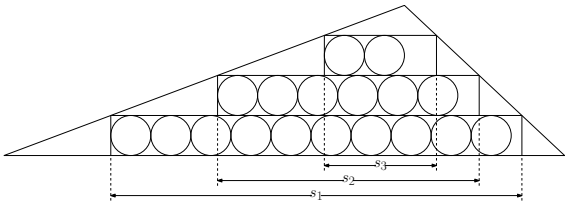


Figure 5 Disks packed in layers of height 2 into a general triangle.

The algorithm therefore returns $n_{app} = max(k, \lfloor ka/2 - ak(k+1)/(2h) - k \rfloor)$ as an approximation of n_{max}. As can be easily seen, this is at least half of the true number of disks packed.

It remains to be shown that n_{app} is only by a constant factor smaller than n_{max}.

Let us call a placement of a set of unit disks *maximal* (with respect to T) if no unit disk inside T can be added to the placement without intersecting others. Therefore, any disk of an optimal packing must be intersected (properly) by a disk of a maximal placement. On the other hand, since the kissing number of disks is 6, any disk of the maximal placement can intersect at most 5 disks of the optimal packing. Therefore, if there is a maximal placement of size n than the maximum packing has size at most $5n$.

As can be seen, the packing in Figure 5 is not yet maximal. We achieve this by an enlarged placement (which is not a packing any more) as shown in Figure 6: all layers

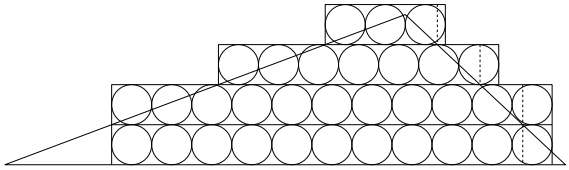


Figure 6 Disks placed in layers of height 2 onto a general triangle.

$i = 1, \cdots, k$ are moved up one level and an additional disk is added at their right end. Only the new layer 1 is the old one with the added disk. As can be easily seen, this placement is maximal and has at most three times as many disks as the previous packing.

Combining these ideas and summarizing we obtain:

▶ **Theorem 4.1.** *Given an arbitrary triangle T as input, the algorithm described returns a value n_{app} which is a constant factor approximation of the maximum number n_{max} of unit disks that can be packed into T.*

5 Conclusion

As was mentioned before, this contribution is only a small step towards determining complexity aspects of packing unit disks into given containers. Work in progress is the problem for arbitrary skinny parallelograms. In general, it would be interesting to gain more insight in the case of polygonal containers, even under the restriction that they are convex.

Acknowledgments. We thank Sándor Fekete for valuable hints concerning this reseach.

References

1 Packomania. URL: http://www.packomania.com/.

2 Helmut Alt and Nadja Scharf. Polynomial Time Approximation Schemes for Circle Packing Problems. In *Proceedings of the 33th European Workshop on Computational Geometry (EuroCG)*, pages 101–104, 2017. URL: http://csconferences.mah.se/eurocg2017/proceedings.pdf.

3 Saugata Basu, Richard Pollack, and Marie-Françoise Roy. On the Combinatorial and Algebraic Complexity of Quantifier Elimination. *J. ACM*, 43(6):1002–1045, 1996.

4 L. Fejes Tóth. *Lagerungen in der Ebene auf der Kugel und im Raum.* Springer Berlin Heidelberg, 2nd edition, 1972.

5 Thomas Hales, Mark Adams, Gertrud Bauer, Dat Dang, John Harrison, Truong Hoang, Cezary Kaliszyk, Victor Magron, Sean McLaughlin, Thang Nguyen, Truong Nguyen, Tobias Nipkow, Steven Obua, Joseph Pleso, Jason Rute, Alexey Solovyev, An Ta, Trân Trung, Diep Trieu, and Roland Zumkeller. A formal proof of the Kepler conjecture. *Forum of Mathematics, Pi*, 5, 2015.

6 Dorit S. Hochbaum and Wolfgang Maass. Approximation Schemes for Covering and Packing Problems in Image Processing and VLSI. *J. ACM*, 32(1):130–136, 1985.

Improved constant factor for the unit distance problem

Péter Ágoston[1] and Dömötör Pálvölgyi[2]

1 **MTA-ELTE Lendület Combinatorial Geometry Research Group, Institute of Mathematics, Eötvös Loránd University (ELTE), Budapest, Hungary**
agostonp@cs.elte.hu

2 **MTA-ELTE Lendület Combinatorial Geometry Research Group, Institute of Mathematics, Eötvös Loránd University (ELTE), Budapest, Hungary**
dom@cs.elte.hu

Abstract

We prove that the number of unit distances among n planar points is at most $2.09 \cdot n^{4/3}$, improving on the previous best bound of $8n^{4/3}$. We also improve the best known extremal values for $n \geq 22$.

1 Introduction

Call a simple graph a *unit distance graph* (UDG) if its vertices can be represented by distinct points in the plane so that the pairs of vertices connected with an edge correspond to pairs of points at unit distance apart. Denote the maximal number of edges in a unit distance graph with n vertices by $u(n)$. Erdős [4] raised the problem to determine $u(n)$ and this question became known as the *Erdős Unit Distance Problem*. Erdős established the bounds $n^{1+c/\log\log n} \leq u(n) \leq O(n^{3/2})$. The lower bound remained unchanged, but the upper bound has been improved several times, the current best has been $O(n^{4/3})$ [8] for more than 35 years. For a detailed survey, see [10].

It turned out during the Polymath16 project[1] that improved bounds even for small values of n might give better bounds for questions related to the chromatic number of the plane. Our goal is to give an explicit upper bound, thus a constant factor improvement of the $O(n^{4/3})$ bound. Prior to our work, the best explicit constant (we know of) is the one derived from an argument of Székely [9], which gives $u(n) \leq 8n^{4/3}$ for all n. Our main result is the following constant factor improvement.

▸ **Theorem 1.1.** $u(n) \leq \frac{\sqrt[3]{\frac{2}{3}(2+\sqrt{3})\cdot 29}}{2} \cdot n^{4/3} = 2.08\ldots \cdot n^{4/3}$.

Our proof is based on a careful examination of Székely's argument to get rid of a few extra factors, an improved crossing lemma and some simple observations about UDG.

1.1 The crossing lemma

Draw a (not necessarily simple) graph in the plane so that vertices are mapped to points and edges to simple curves that do not go through the images of the vertices other than their endpoints'. The *crossing number of a graph* G, denoted by $cr(G)$, is defined as the minimum number of intersection points among the edges of G in such drawings, counted with multiplicity. The crossing lemma, which was first proved by Ajtai, Chvátal, Newborn, Szemerédi [2] and, independently, by Leighton [5] is that for any simple graph $cr(G) \geq \Omega(\frac{e^3}{n^2})$

[1] https://dustingmixon.wordpress.com/2018/04/14/polymath16

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

if $e \geq 4n$, where n is the number of vertices and e is the number of edges. The hidden constant has been improved several times; we will need the following variant.

▸ **Lemma 1.2** (Ackerman [1]). *If a simple graph has n vertices and e edges, then $cr(G) \geq \frac{e^3}{29n^2} - \frac{35n}{29}$. Moreover, if $e \geq 6.95n$, then $cr(G) \geq \frac{e^3}{29n^2}$.*

2 Proof of the improved bound

Fix a UDG on n vertices with $u(n)$ edges and the images of the vertices of one of its planar realizations; denote these n points by P and the UDG by G. In the following, we do not differentiate between vertices and their images. Note that due to the maximality of G, any two points at unit distance form an edge.

▸ **Lemma 2.1.** *If $n \geq 3$, then all vertices of G have degree at least 2, and if $n \geq 7$, then there is at least one UDG with the same vertex and edge number as G, for which there is at most one vertex with degree 2.*

Proof. Suppose that a vertex Z has degree one. Take a Euclidean coordinate system so that no two points have the same x coordinate: if for some coordinate system there were two such points, rotate with a sufficiently small angle. Delete Z and order the unit edges of the remaining graph $G \setminus Z$ such that $AB < CD$ if for the x-coordinates of these four points we have $\min(A_x, B_x) < \min(C_x, D_x)$ or $(\min(A_x, B_x) = \min(C_x, D_x)) \wedge (\max(A_x, B_x) < \max(C_x, D_x))$. The above is a total ordering as for any pair of different edges, either the left or the right endpoints are different from each other and so are their x coordinates. Take the smallest unit edge AB according to this ordering. For (at least) one of the points forming a unit equilateral triangle with AB, say, C, we would have $AC < AB$ or $BC < AB$ if $C \in P \setminus Z$. This would contradict the minimality of AB, thus $C \notin P \setminus Z$. Moreover, $C \neq Z$, as then its degree would have been at least two. But replacing Z with C gives a UDG with more edges, contradicting the maximality of G.

For the second statement, first delete all the vertices with degree at most 2, recursively, until no vertices with degree at most 2 are left. It is not possible that all vertices were deleted as that would mean that there were at most $2n-3$ edges (along with the last two vertices we have deleted at most 1 and 0 edges, respectively, while along with the other ones, we have deleted at most 2), but for $n \geq 7$, there exist UDGs with more than $2n-3$ edges, for example the ones constructed similarly to the graph in Figure 1.

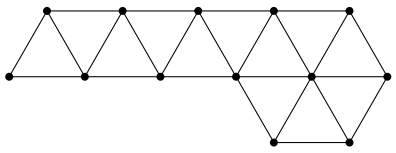


Figure 1

So after the deletions, we have a graph with a positive number of vertices all having degree at least 3. Thus, we can insert a new vertex connected to the two endpoints of a minimal edge in the current graph according to the ordering in the first part. We repeat this for each deleted vertex. In each step at least one of the new edges will be smaller (according to the ordering) than all the previously existing ones, so we always add the new vertex such that it is neighbouring the previously added one, guaranteeing that the previous one now

has degree at least 3. So in the end at most one vertex with degree 2 will remain (the one added last) and the graph will have the same number of vertices and edges as G had. ◂

Now we proceed with the proof of Theorem 1.1. The statement is true for $n \le 6$, so from now on we assume $n \ge 7$. From Lemma 2.1, we can assume that P was chosen in a way that G has no vertex with degree at most 1 and has at most one vertex with degree 2.

Following Székely [9], define another graph on P, denoted by H, whose edges are those arcs on the unit circles around the points of P, which end in points from P and do not contain points of P in their interior. As any unit circle around a point of P has at least two points from P on it, H has no loops, but it might have multiple edges. Also, $|E(H)| = 2u(n)$, since for every point of P, there are exactly as many arcs on the unit circle around it as the degree of the corresponding point in G, so each edge is counted twice.

The second part of Lemma 2.1 implies that there is at most one pair of points which are connected on the same circle with two arcs. If this happens, we delete one of these two edges, to get a graph H^- with $|E(H^-)| = 2u(n) - 1$. Otherwise, to simplify presentation, delete an arbitrary edge to obtain H^-. The crossing number of H^- is bounded as $cr(H^-) \le 2 \cdot \binom{n}{2} = n^2 - n$, since all pairs of circles intersect each other in at most 2 points.

First of all, to deal with the case $n \le 500$, we lower bound the intersections among these unit circles in the following way. For a vertex v with degree $deg(v)$, there are exactly $\binom{deg(v)}{2}$ pairs of circles that intersect in v. Therefore, we have $n^2 - n \ge \sum_v \binom{deg(v)}{2} \ge n \cdot \binom{\sum_v deg(v)/n}{2} = u(n)\left(\frac{2u(n)}{n} - 1\right)$ from Jensen's inequality. This is roughly $u(n) \le \sqrt{n^3/2}$ and the RHS is less than $2n^{4/3}$ if $n < 512$ – a more precise calculation confirms the statement in this range.[2] Moreover, quite surprisingly, this simple bound (combined with a linear lower bound for the number of crossings in H) beats the best previous bound for small n starting from $n = 25$. (These values can be found in Table 1.)

The only possibility left of having multiple edges in H^- is that there can be more unit circles centered at some points of P passing through a pair of points from P. Since there are at most two unit circles through any pair of points, all edges occur with multiplicity at most 2 in H^-. Denote the simple graph formed by the edges of H^- by H_1 and the *simple* graph formed by the edges that have multiplicity 2 by H_2, and the number of their edges by e_1 and e_2, respectively, so $e_1 + e_2 = 2u(n) - 1$ and $e_1 \ge e_2$.

Now we prove $cr(H_1) + 3 \cdot cr(H_2) \le cr(H^-)$. From a drawing of H^- we can obtain a drawing of H_1 or H_2 by picking one of the two embeddings of each edge of H_2 randomly, independently from each other, with probability $\frac{1}{2}$. In this drawing of H_1 any crossing of two edges of $H_1 \setminus H_2$ is preserved, while a crossing of an edge of $H_1 \setminus H_2$ and an edge of H_2 is preserved with probability $\frac{1}{2}$, and a crossing of two edges of H_2 is preserved with probability $\frac{1}{4}$. In the drawing of H_2 only crossings of two edges of H_2 are preserved, each with probability $\frac{1}{4}$. Summing these up, the expected value of crossings in the random drawing of H_1 plus three times the crossings in the random drawing of H_2 obtained this way is at most $cr(H^-)$.

First, suppose that $e_2 \ge 6.95n$. By applying Lemma 1.2 to H_1 and H_2, respectively, we get $e_1 \le \sqrt[3]{29n^2 \cdot cr(H_1)}$ and $e_2 \le \sqrt[3]{29n^2 \cdot cr(H_2)}$. From these, $2u(n) - 1 = e_1 + e_2 \le \sqrt[3]{29n^2 \cdot cr(H_1)} + \sqrt[3]{29n^2 \cdot cr(H_2)}$. If we write $x = \frac{cr(H_1)}{cr(H^-)}$, then $\sqrt[3]{cr(H_1)} + \sqrt[3]{cr(H_2)} \le \sqrt[3]{x \cdot cr(H^-)} + \sqrt[3]{\frac{1-x}{3} \cdot cr(H^-)} = \left(\sqrt[3]{x} + \sqrt[3]{\frac{1-x}{3}}\right) \cdot \sqrt[3]{cr(H^-)}$ for $0 \le x \le 1$, so it is enough to

[2] Note that in the later parts of our proof we could also reduce $cr(H^-)$ with $u(n)(\frac{2u(n)}{n} - 1)$ using this argument, but it would not change its order of magnitude or effect the constant we obtain.

maximize $\sqrt[3]{x} + \sqrt[3]{\frac{1-x}{3}}$. And since its derivative is $\frac{1}{3} \cdot (x)^{-\frac{2}{3}} - \frac{1}{9} \cdot \left(\frac{1-x}{3}\right)^{-\frac{2}{3}}$, it has a zero value exactly if $x^{-\frac{2}{3}} = \frac{1}{3} \cdot \left(\frac{1-x}{3}\right)^{-\frac{2}{3}}$, i.e., if $x = \sqrt{3}(1-x)$, which gives $x = \frac{\sqrt{3}}{1+\sqrt{3}}$ (we could do these as both x and $\frac{1-x}{3}$ are positive in the interior of the $[0,1]$ interval). Also, both of the summands in the derivative are monotonously decreasing, thus so is the derivative, so the maximum is attained at the above x, and equals to $\sqrt[3]{\frac{2}{3}\left(2+\sqrt{3}\right)}$. From here the maximum of $2u(n) - 1 = e_1 + e_2$ is $\sqrt[3]{\frac{2}{3}\left(2+\sqrt{3}\right)} \cdot \sqrt[3]{29n^2 \cdot cr(H^-)} \leq \sqrt[3]{\frac{2}{3}\left(2+\sqrt{3}\right)} \cdot \sqrt[3]{29 \cdot (n^4 - n^3)}$. This implies $u(n) \leq \frac{\sqrt[3]{\frac{2}{3} \cdot (2+\sqrt{3}) \cdot 29}}{2} \cdot n^{\frac{4}{3}} \approx 2.082 \cdot n^{\frac{4}{3}}$.

The second case is when $e_2 < 6.95n$ and $e_1 < 6.95n$. Then $2u(n) - 1 = e_1 + e_2 < 13.9n$, from which $u(n) < 6.95n + 0.5$, which is less than $\frac{\sqrt[3]{\frac{2}{3} \cdot (2+\sqrt{3}) \cdot 29}}{2} \cdot n^{\frac{4}{3}}$ for $n > 37$.

Finally, if $e_2 < 6.95n$ and $e_1 \geq 6.95n$, then $cr(H_1) \leq n^2 - n$, which implies $e_1 \leq \sqrt[3]{29 \cdot (n^4 - n^3)}$. So $2 \cdot u(n) - 1 = e_1 + e_2 \leq \sqrt[3]{29 \cdot (n^4 - n^3)} + 6.95n$ meaning that $u(n) \leq \frac{\sqrt[3]{29 \cdot (n^4 - n^3)}}{2} + 3.475n + 0.5$, which is less than $\frac{\sqrt[3]{\frac{2}{3} \cdot (2+\sqrt{3}) \cdot 29}}{2} \cdot n^{\frac{4}{3}}$ for $n > 256$.

This finishes the proof of Theorem 1.1.

3 Best bounds for $n \leq 30$

In Table 1 we list the best known bounds, along with constructions.

For $n \leq 14$, the exact values of $u(n)$ are known, established in the thesis of Schade [7]. For $n \leq 13$, the drawn ones are known to be the only maximal UDGs.[3] In general, the upper bounds can be obtained using the inequality $u(n) \leq \lfloor \frac{n}{n-2} \cdot u(n-1) \rfloor$ (for $n \geq 3$), which is true because the edge density of the maximal UDGs with n vertices is monotonously decreasing in n as all subgraphs of a UDG are also UDGs. This was observed by Schade, who sometimes also applied additional tricks – the values where such tricks are needed are denoted by a star.

For $n \geq 15$, the lower bounding graphs are also by Schade, with the exception of the ones for $n = 29$, $n = 30$ and the second graph for $n = 28$, which are our improvements based on the graph for $n = 27$ by Schade. The upper bounding values from $n \geq 22$ are also our improvements, derived from a refinement of the inequality $n^2 - n \geq \sum_v \binom{deg(v)}{2}$. (The improved values are marked in bold.) For the refined inequality, we need the following lemma.

▸ **Lemma 3.1.** *For a graph H with n vertices in which all edges have multiplicity at most 2 and the number of edges (counted with multiplicity) is denoted by e, $cr(H) \geq 2 \cdot (e - 2 \cdot (3n-6)) = 2e - 12n + 24$.*

Proof. Let H be a graph with n vertices and e edges satisfying the conditions for which $cr(H)$ is minimal. Take a drawing of H with the minimal possible number of crossings. We can suppose that the two copies of an edge that occurs with multiplicity run close to each other, otherwise we could redraw any one of them sufficiently close to the other one (to whichever has fewer crossings). We can also suppose that H contains at most one single edge as otherwise we could take any two single edges and replace the one with the more crossings with another one close to (and parallel with) the one with the fewer crossings without increasing the number of crossings (note that this step also changes H, not only the drawing).

[3] Note that in [3] it is incorrectly stated also for $n = 14$ that the constructions were proved to be unique.

Take a maximal plane subgraph H' of H that has the maximal number of edges with multiplicity two in it. This has at most $2 \cdot (3n-6)$ edges and all the other edges cross at least one of the double edges of H', thus adding at least $2 \cdot (e - 2 \cdot (3n-6))$ crossings. ◂

Recall that $n^2 - n \geq cr(H) + \sum_v \binom{deg_G(v)}{2}$ where G is a unit distance graph and H is the graph obtained from G, as described previously, with edges of multiplicity at most two, except one, that could have multiplicity three. This, however, could only occur if G had a vertex of degree two, which is impossible if $u(n) \geq u(n-1) + 3$, which we can assume. Thus, applying Lemma 3.1 to H gives $n^2 - n \geq 4 \cdot |E(G)| - 12n + 24 + \sum_{v \in V(G)} \binom{deg(v)}{2} \geq 4 \cdot u(n) - 12n + 24 + n \cdot \left(1 - \left\{\frac{2u(n)}{n}\right\}\right) \cdot \binom{\lfloor \frac{2u(n)}{n} \rfloor}{2} + n \cdot \left\{\frac{2u(n)}{n}\right\} \cdot \binom{\lceil \frac{2u(n)}{n} \rceil}{2}$ (where $\{x\}$ denotes the fractional part of x), which gives our improved upper bounds for $u(n)$.

As can be seen, the upper bounds diverge quite fast from the lower bounds. From around $n = 600$, Theorem 1.1 gives the best upper bound.

Acknowledgement

The main result was obtained while working on the currently ongoing Polymath16 project about the Hadwiger–Nelson problem and is related, but not directly connected to it.

References

1 E. Ackerman: On topological graphs with at most four crossings per edge *Computational Geometry* **85**, December 2019, 101574, 35 pages.

2 M. Ajtai, V. Chvátal, M. M. Newborn, E. Szemerédi (1982), Crossing-free subgraphs, *Theory and practice of combinatorics*, North-Holland Mathematics Studies, 60, North-Holland, Amsterdam, 9–12.

3 P. Brass, W. O. J. Moser, J. Pach: Research Problems in Discrete Geometry (2005).

4 P. Erdős: On sets of distances of n points, Amer. Math. Monthly 53 (1946), 248-–250.

5 T. Leighton, Complexity Issues in VLSI, Foundations of Computing Series, Cambridge, MA: MIT Press. (1983).

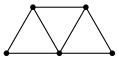
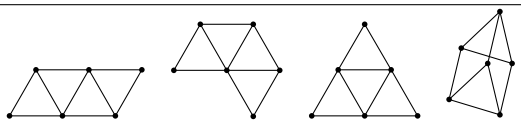
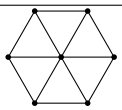
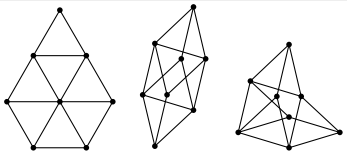
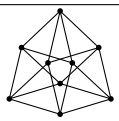
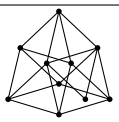
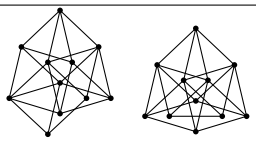
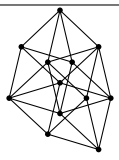
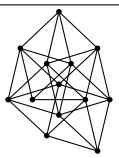
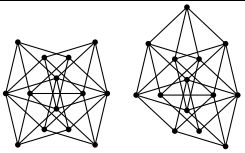
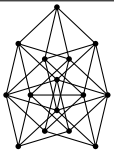
6 J. Pach, R. Radoičić, G. Tardos, G. Tóth: Improving the crossing lemma by finding more crossings in sparse graphs, *Discrete and Computational Geometry* (2006), **36** (4): 527–552.

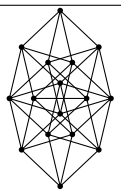
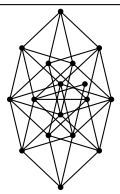
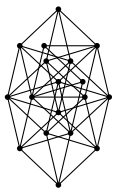
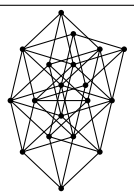
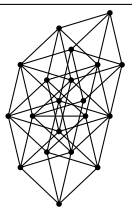
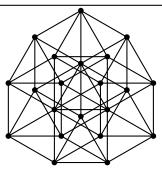
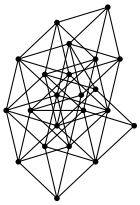
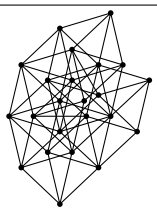
7 C. Schade: Exakte maximale Anzahlen gleicher Abstände (1993).

8 J. Spencer, E. Szemerédi, W. Trotter Jr.: Unit Distances in the Euclidean Plane, *Graph Theory and Combinatorics* (1984), 293–303.

9 L. Székely: Crossing numbers and hard Erdős problems in discrete geometry, *Combinatorics, Probability and Computing* **6** (1997), 353–358.

10 E. Szemerédi: Erdős's Unit Distance Problem, *Open Problems in Mathematics* (2016), 459–477.

n	$u(n)$	Lower bounding graph(s)
1	0	
2	1	
3	3	
4	5*	
5	7*	
6	9*	
7	12	
8	14*	
9	18	
10	20*	
11	23*	
12	27	
13	30*	
14	33*	
15	37 or 38	

n	$u(n)$	Lower bounding graph(s)
16	41 or 42*	
17	43–47	
18	46–52	
19	50–57*	
20	54–63	
21	57–68*	
22	60–**72**	
23	64–**77**	

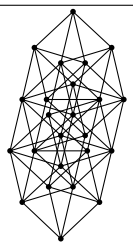
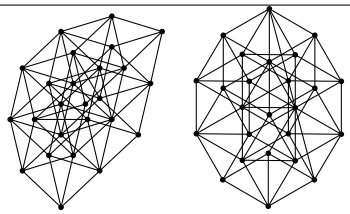
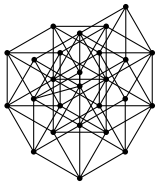
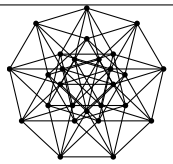
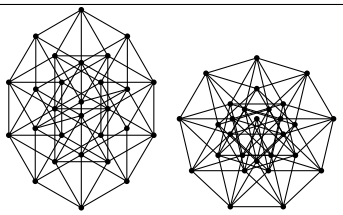
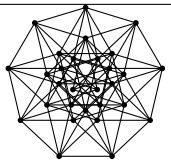
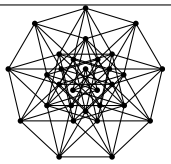
n	$u(n)$	Lower bounding graph(s)
24	68–**82**	
25	72–**87**	
26	76–**92**	
27	81–**97**	
28	85-**102**	
29	**89–108**	
30	**93–113**	

Table 1 Best bounds for the maximal number of unit distances $u(n)$ among n planar points.