Algorithmic Enumeration of Surrounding Polygons

Katsuhisa Yamanaka[1], Takashi Horiyama[2], Yoshio Okamoto[3,4], Ryuhei Uehara[5], and Tanami Yamauchi[1]

1 **Iwate University, Japan**
yamanaka@cis.iwate-u.ac.jp, tanami@kono.cis.iwate-u.ac.jp
2 **Saitama University, Japan**
horiyama@al.ics.saitama-u.ac.jp
3 **The University of Electro-Communications, Japan**
okamotoy@uec.ac.jp
4 **RIKEN Center for Advanced Intelligence Project, Japan**
5 **Japan Advanced Institute of Science and Technology, Japan**
uehara@jaist.ac.jp

Abstract

We are given a set S of points in the Euclidean plane. We assume that S is in general position. A simple polygon P is a *surrounding polygon* of S if each vertex of P is a point in S and every point in S is either inside P or a vertex of P. In this paper, we present an enumeration algorithm of the surrounding polygons for a given point set. Our algorithm is based on reverse search by Avis and Fukuda and enumerates all the surrounding polygons in polynomial delay.

1 Introduction

Enumeration problems are fundamental and important in computer science. Enumerating geometric objects are studied for triangulations [2, 3, 9], non-crossing spanning trees [9], pseudoline arrangements [20], non-crossing matchings [19], unfoldings of Platonic solids [8], and so on. In this paper, we focus on an enumeration problem of simple polygons of a given point set. We are given a set S of n points in the Euclidean plane. A *surrounding polygon* of S is a simple polygon P such that each vertex of P is a point in S and every point in S is either inside the polygon or a vertex of the polygon. A surrounding polygon P of S is a *simple polygonization*[1] of S if every point of S is a vertex of P. See Figure 1 for examples.

Simple polygonizations are studied from various perspectives. As for the counting, the current fastest algorithm was given by Marx and Miltzou [10], and it runs in $n^{O\sqrt{n}}$ time when a set of n points is given. It is still an outstanding open problem to propose a polynomial-time algorithm that counts the number of simple polygonizations of a given point set [12]. Much attention has been paid for combinatorial counting, too. A history on the lower and upper bounds is summarized by Demaine [4] and O'Rourke *et al.* [14]. Let b_P be the number of simple polygonizations of a point set P, and let b_n be the maximum of b_P among all the sets P of n points. The current best lower and upper bounds for b_n are 4.64^n [5] and 54.55^n [15], respectively.

Another research topic is a random generation of simple polygonizations. Since no polynomial-time counting algorithm is known for simple polygonizations, it seems to be a hard task to propose a polynomial-time algorithm that uniformly generates simple polygonizations. However, uniformly random generations are known for restricted classes: x-monotone polygons [21] and star-shaped polygons [16]. These uniform random generations are based

[1] The simple polygonizations are also called spanning cycles, Hamiltonian polygons, and planar traveling salesman tours.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

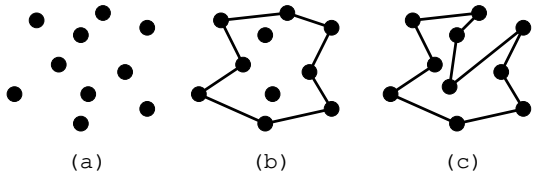


Figure 1 (a) A point set S. (b) A surrounding polygon of S. (c) A simple polygonization of S.

on counting. For general simple polygonizations, heuristic algorithms are known [1, 17, 21]. Those algorithms efficiently generate simple polygons, but not uniformly at random.

On the other hand, nothing is known for the problem of enumerating all the simple polygonizations, as mentioned in [18]. A trivial enumeration is to generate all the permutations of given points, then output only simple polygonizations. However, this is clearly a time-consuming algorithm. It is an interesting and challenging question whether all the simple polygonizations of a given point set can be enumerated efficiently (for example, in output-polynomial time[2] or in polynomial delay[3]).

As the first step toward the question, we consider the problem of enumerating the surrounding polygons of a given point set S. From the definition, the set of surrounding polygons of S includes the set of simple polygonizations of S. We show that, for this enumeration problem, the reverse search by Avis and Fukuda [2] can be applied. First, we introduce an "embedding" operation: deleting a vertex from a surrounding polygons and putting it inside the polygon. Then, using this operation, we define a rooted tree structure among the set of surrounding polygons of S. We show that, by traversing the tree, one can enumerate all the surrounding polygons. The proposed algorithm enumerates them in polynomial delay.

Due to space limitation, all the proofs and some details are omitted.

2 Preliminaries

A *simple polygon* is a closed region of the plane enclosed by a simple cycle of edges. Here, a simple cycle means that two adjacent line segments intersect only at their common endpoint and no two non-adjacent line segments intersect. An *ear* of a simple polygon P is a triangle such that one of its edges is a diagonal of P and the remaining two edges are edges of P. The following theorem for ears is known.

▶ **Theorem 2.1** ([11]). *Every simple polygon with $n \geq 4$ vertices has at least two non-overlapping ears.*

Let S be a set of n points in the Euclidean plane. We assume that S is in general position, i.e., no three points are collinear. The *upper-left point* of S is the point with the minimum x-coordinate. If a tie exists, we choose the point with the maximum y-coordinate among them. A *surrounding polygon* of S is a simple polygon such that every point in S is either inside the polygon or a vertex of the polygon. For example, the convex hull of S is a

[2] The running time of an enumeration algorithm A for an enumeration problem is *output-polynomial* if the total running time of A is bounded by a polynomial in the input and output size of the problem.

[3] The running time of an enumeration algorithm A for an enumeration problem is *polynomial-delay* if the delay, which is the maximum computation time between any two output, of A is bounded by a polynomial in the input size of the problem.

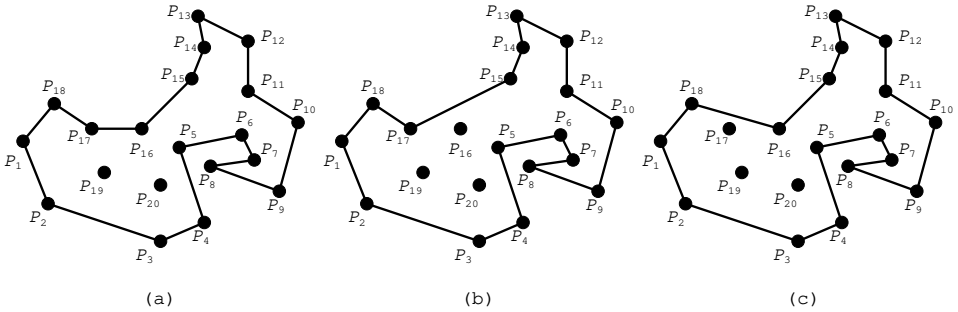


Figure 2 (a) A surrounding polygon, where $p_6, p_7, p_{11}, p_{14}, p_{15}, p_{16}$, and p_{17} are embeddable. (b) The surrounding polygon obtained by embedding p_{16}. The point p_{16} is embedded inside the polygon. (c) The parent of the polygon in (a), which is obtained by embedding p_{17}.

surrounding polygon of S. Note that any surrounding polygon has the upper-left point in S as a vertex.

We denote by $\mathcal{P}(S)$ the set of surrounding polygons of S, and denote by $\mathsf{CH}(S)$ the convex hull of S. We denote a surrounding polygon of S by a (cyclic) sequence of the vertices in the surrounding polygon. Let $P = \langle p_1, p_2, \ldots, p_k \rangle$ be a surrounding polygon of S. Throughout this paper, we assume that p_1 is the upper-left point in S, the vertices on P appear in counterclockwise order, and the successor of p_k is p_1. Let p be a vertex of a surrounding polygon P of S. We denote by $\mathsf{pred}(p)$ and $\mathsf{succ}(p)$ the predecessor and successor of p on P, respectively.

3 Family tree

Let S be a set of n points in the Euclidean plane, and let $\mathcal{P}(S)$ be the set of surrounding polygons of S. In this section, we define a tree structure over $\mathcal{P}(S)$ such that its nodes correspond to the surrounding polygons. To define a tree structure, we first define the parent of a surrounding polygon using the "embedding operation" defined below. Then, using the parent-child relationship, we define the tree structure rooted at $\mathsf{CH}(S)$.

Now, we introduce some notations. Let $P = \langle p_1, p_2, \ldots, p_k \rangle$ be a surrounding polygon of S. Recall that p_1 is the upper-left vertex on P and the vertices on P are arranged in the counterclockwise order. We denote by $p_i \prec p_j$ if $i < j$ holds, and we say that p_j is *larger than* p_i. The vertex p of P is *embeddable* if the triangle consisting of $\mathsf{pred}(p)$, p, and $\mathsf{succ}(p)$ does not intersect the interior of P. See examples in Figure 2(a). In the figure, $p_6, p_7, p_{11}, p_{14}, p_{15}, p_{16}$, and p_{17} are embeddable.

▶ **Lemma 3.1.** *Let S be a set of points, and let P be a surrounding polygon in $\mathcal{P}(S) \setminus \{\mathsf{CH}(S)\}$. Then, P has at least one embeddable vertex.*

Now, let us define an operation that makes another surrounding polygon from a surrounding polygon. Let p be an embeddable vertex on P. An *embedding operation* is to remove the two edges $(\mathsf{pred}(p), p)$ and $(p, \mathsf{succ}(p))$ and insert the edge $(\mathsf{pred}(p), \mathsf{succ}(p))$. Intuitively, an embedding operation "embeds" a vertex into the interior of P. See Figure 2.

We denote by $\mathsf{larg}(P)$ the largest embeddable vertex on P. The *parent* of P, denoted by $\mathsf{par}(P)$, is the polygon obtained by embedding $\mathsf{larg}(P)$ on P. Note that $\mathsf{par}(P)$ is also a surrounding polygon of S. By repeatedly finding the parents from P, we obtain a sequence of surrounding polygons. The *parent sequence* $\mathsf{PS}(P) = \langle P_1, P_2, \ldots, P_\ell \rangle$ of P is a sequence of

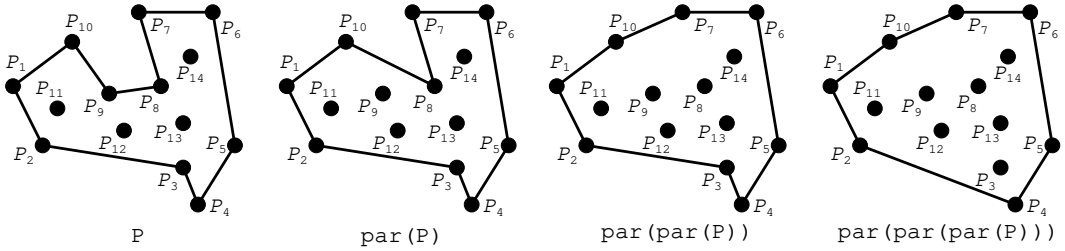


Figure 3 A parent sequence.

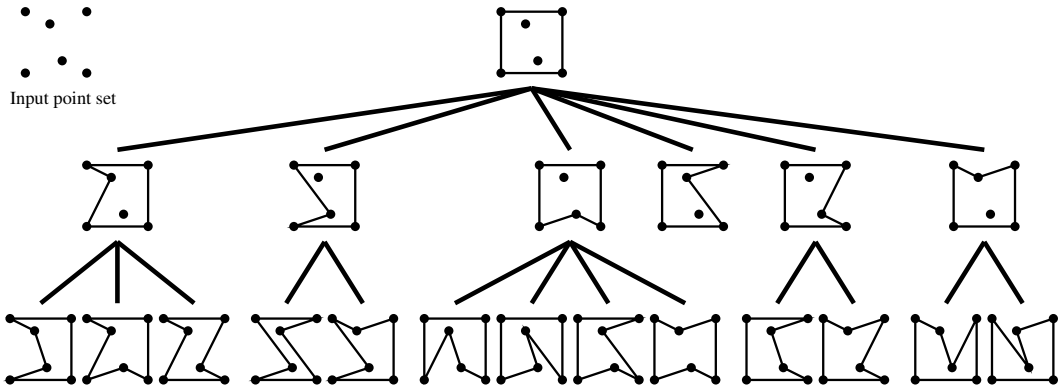


Figure 4 An example of a family tree.

surrounding polygons such that the first polygon is P itself and P_i is the parent of P_{i-1} for each $i = 2, 3, \ldots, \ell$. See Figure 3. As we can see in the following lemma, the last polygon in a parent sequence is always $\mathsf{CH}(P)$.

▶ **Lemma 3.2.** *Let S be a set of n points in the Euclidean plane, and let P be a surrounding polygon in $\mathcal{P}(S) \setminus \{\mathsf{CH}(S)\}$. The last polygon of $\mathsf{PS}(P)$ is $\mathsf{CH}(S)$.*

From Lemma 3.2, for any surrounding polygon, the last polygon of its parent sequence is the convex hull. By merging the parent sequences for all surrounding polygons in $\mathcal{P}(S)$, we have the tree structure rooted at $\mathsf{CH}(S)$. We call such a tree the *family tree*. An example of the family tree is shown in Figure 4.

4 Enumeration algorithm

In this section, we present an algorithm that, for a given set S of n points, enumerates all the surrounding polygons in $\mathcal{P}(S)$. In the previous section, we defined the family tree among $\mathcal{P}(S)$. We know that the root of the family tree is the convex hull of S. Hence, we have the following enumeration algorithm. We first construct the convex hull of S. Then, we traverse the (implicitly defined) family tree with depth first search. This algorithm can enumerate all the surrounding polygons in $\mathcal{P}(S)$. To perform the search, we design an algorithm that finds all the children of any surrounding polygon of S. Starting from the root, we apply the child-enumeration algorithm recursively, and then we can traverse the family tree.

To describe how to construct children, we introduce some notations. Let $P = \langle p_1, p_2, \ldots, p_k \rangle$ be a surrounding polygon in $\mathcal{P}(S)$. For an edge (p_i, p_{i+1}) of P and a point p inside P, we denote by $P(p_i, p_{i+1}; p)$ the polygon obtained by removing (p_i, p_{i+1}) and inserting two edges

(p_i, p) and (p, p_{i+1}). Intuitively, this operation is the reverse one of embedding operation. We call it a *dig operation*. Any child of P is described as $P(p_i, p_{i+1}; p)$ for some p, p_i, and p_{i+1}. Hence, for all possible $P(p_i, p_{i+1}; p)$, if we can check whether or not $P(p_i, p_{i+1}; p)$ is a child, then one can enumerate all the children. We have the following observation.

▶ **Lemma 4.1.** *Let P be a surrounding polygon of a set of points. For an edge (p_i, p_{i+1}) of P and a point p inside P, $P(p_i, p_{i+1}; p)$ is a child of P if*
(1) $P(p_i, p_{i+1}; p)$ is a surrounding polygon of S and
(2) $\mathsf{par}(P(p_i, p_{i+1}; p)) = P$ holds.

Note that the condition (2) in Lemma 4.1 can be rephrased as follows: p is the largest embeddable vertex in $P(p_i, p_{i+1}; p)$. Using the conditions in Lemma 4.1, we obtain the child-enumeration algorithm. For every possible $P(p_i, p_{i+1}; p)$, we check whether or not $P(p_i, p_{i+1}; p)$ is a child of P. We apply the algorithm recursively starting from the convex hull. Thus, we can traverse the family tree. In this way, one can enumerate all the surrounding polygons. In each recursive call, there are $\mathrm{O}(n^2)$ child candidates $P(p_i, p_{i+1}; p)$. We can check whether or not $P(p_i, p_{i+1}; p)$ is a child in $\mathrm{O}(\log n)$ time using triangular range query [6] with $\mathrm{O}(n^2)$-time preprocessing and $\mathrm{O}(n^2)$ additional space for an input point set and shortest path query [7] with $\mathrm{O}(n)$-time preprocessing for each surrounding polygon. Thus, each recursive call takes $\mathrm{O}(n^2 \log n)$ time. Now we have the following theorem.

▶ **Theorem 4.2.** *Let S be a set of n points in the Euclidean plane. One can enumerate all the surrounding polygons in $\mathcal{P}(S)$ in $O(n^2 \log n\, |\mathcal{P}(S)|)$-time and $O(n^2)$ space.*

From the theorem above, one can see that our algorithm is output-polynomial. Using the even-odd traversal in [13], we have a polynomial-delay enumeration algorithm. In the traversal, the algorithm outputs polygons with even depth when we go down the family tree and output polygons with odd depth when we go up. See [13] for further details. We have the following corollary.

▶ **Corollary 4.3.** *Let S be a set of n points in the Euclidean plane. There is an $O(n^2 \log n)$-delay and $O(n^2)$-space algorithm that enumerates all the surrounding polygons in $\mathcal{P}(S)$.*

Acknowledgments. Part of this work has been discussed during the Japan-Austria Bilateral Seminar: Computational Geometry Seminar with Applications to Sensor Networks in November 2018. The authors thank the organizers for providing an encouraging atmosphere. They also thank the anonymous referees for their valuable comments. This work was supported by JSPS KAKENHI Grant Numbers JP15K00009, JP15H05711, JP16K00002, JP17H06287, JP18H04091, JP18K11153. The third author is also supported by JST CREST Grant Number JPMJCR1402 and Kayamori Foundation of Informational Science Advancement.

References

1 Thomas Auer and Martin Held. Heuristics for the generation of random polygons. In *Proceedings of the 8th Canadian Conference on Computational Geometry*, pages 38–43, 1996.
2 David Avis and Komei Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65(1-3):21–46, 1996.
3 Sergei Bespamyatnikh. An efficient algorithm for enumeration of triangulations. *Computational Geometry Theory and Applications*, 23(3):271–279, 2002.
4 Erik D. Demaine. `http://erikdemaine.org/polygonization/`, 2012.

5 Alfredo García, Marc Noy, and Javier Tejel. Lower bounds on the number of crossing-free subgraphs of K_N. *Computational Geometry*, 16(4):211–221, 2000.

6 Partha P. Goswami, Sandip Das, and Subhas C. Nandy. Triangular range counting query in 2D and its application in finding k nearest neighbors of a line segment. *Computational Geometry*, 29(3):163 – 175, 2004.

7 Leonidas J. Guibas and John Hershberger. Optimal shortest path queries in a simple polygon. *Journal of Computer and System Sciences*, 39(2):126 – 152, 1989.

8 Takashi Horiyama and Wataru Shoji. Edge unfoldings of platonic solids never overlap. In *Proceedings of the 23rd Annual Canadian Conference on Computational Geometry*, pages 65–70, 2011.

9 Naoki Katoh and Shin-ichi Tanigawa. Enumerating edge-constrained triangulations and edge-constrained non-crossing geometric spanning trees. *Discrete Applied Mathematics*, 157(17):3569–3585, 2009.

10 Dániel Marx and Tillmann Miltzow. Peeling and nibbling the cactus: Subexponential-time algorithms for counting triangulations and related problems. In *32nd International Symposium on Computational Geometry, SoCG 2016*, pages 52:1–52:16, 2016.

11 Gary H. Meisters. Polygons have ears. *American Mathematical Monthly*, 82(6):648–651, 1975.

12 Joseph S. B. Mitchell and Joseph O'Rourke. Computational geometry column 42. *International Journal of Computational Geometry and Applications*, 11(5):573–582, 2001.

13 Shin-ichi Nakano and Takeaki Uno. Generating colored trees. *Proceedings of the 31th Workshop on Graph-Theoretic Concepts in Computer Science, (WG 2005)*, LNCS 3787:249–260, 2005.

14 Joseph O'Rourke, Subhash Suri, and Csaba D. Tóth. Polygons. In *Handbook of Discrete and Computational Geometry, Third Edition.*, pages 787–810. Chapman and Hall/CRC, 2017.

15 Micha Sharir, Adam Sheffer, and Emo Welzl. Counting plane graphs: Perfect matchings, spanning cycles, and Kasteleyn's technique. *J. Comb. Theory Ser. A*, 120(4):777–794, 2013.

16 Christian Sohler. Generating random star-shaped polygons. In *Proceedings of the 11th Canadian Conference on Computational Geometry*, pages 174–177, 1999.

17 Sachio Teramoto, Mitsuo Motoki, Ryuhei Uehara, and Tetsuo Asano. Heuristics for generating a simple polygonalization. IPSJ SIG Technical Report 2006-AL-106(6), Information Processing Society of Japan, May 2006.

18 Emo Welzl. Counting simple polygonizations of planar point sets. In *Proceedings of the 23rd Annual Canadian Conference on Computational Geometry*, 2011. URL: `http://www.cccg.ca/proceedings/2011/papers/invited3.pdf`.

19 Manuel Wettstein. Counting and enumerating crossing-free geometric graphs. *Journal of Computational Geometry*, 8(1):47–77, 2017.

20 Katsuhisa Yamanaka, Shin-ichi Nakano, Yasuko Matsui, Ryuhei Uehara, and Kento Nakada. Efficient enumeration of pseudoline arrangements. In *Proceedings of European Workshop on Computational Geometry 2009*, pages 143–146, March 2009.

21 Chong Zhu, Gopalakrishnan Sundaram, Jack Snoeyink, and Joseph S. B. Mitchell. Generating random polygons with given vertices. *Computational Geometry: Theory and Applications*, 6:277–290, 1996.

A 1/4-Approximation Algorithm for the Maximum Hidden Vertex Set Problem in Simple Polygons*

Carlos Alegría[†1], Pritam Bhattacharya[‡2], and Subir Kumar Ghosh[3]

1 **Posgrado en Ciencia e Ingeniería de la Computación, Universidad Nacional Autónoma de México**
calegria@uxmcc2.iimas.unam.mx

2 **Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur**
pritam.bhattacharya@cse.iitkgp.ernet.in

3 **Department of Computer Science, School of Mathematical Sciences, Ramakrishna Mission Vivekananda University**
subir.ghosh@rkmvu.ac.in

Abstract

Given a simple polygon, two points in its interior are said to be *hidden* to each other if the straight line segment connecting them intersects the exterior of the polygon. We study the *Maximum Hidden Vertex Set* problem, where given a simple polygon, we are required to find a subset of vertices of maximum cardinality such that every pair of them are hidden to each other. This problem is known to be NP-hard, and in fact also APX-hard. In this paper we present a $O(n^2)$ time algorithm to compute a 1/4-approximation to the maximum hidden vertex set of a simple polygon. Although exact algorithms are known for some special classes of polygons (such as polygons that are weakly visible from a convex edge), to the best of our knowledge this is the first deterministic polynomial-time algorithm to compute a constant-factor approximation to the optimal solution for general simple polygons without holes.

Lines 170

1 Introduction

Visibility problems are some of the most prominent and intensively studied problems in Computational Geometry. Since the classic Art Gallery Problem was proposed in 1973 by Victor Klee, many extensions have been studied [11, 15, 9], and combinatorial as well as computational results have been applied to practical problems that can commonly be found in computer generated graphics [4], computer vision [6], and robotics [10].

A well known class of visibility problems are those related to hiding. Given a simple polygon, we say that two points in its interior are *visible* to each other if the line segment connecting the points does not intersect the exterior of the polygon. Conversely, the points are said to be *hidden* to each other if they are not mutually visible. In this paper we study the so called Maximum Hidden Vertex Set (MHVS) problem, where given a simple polygon

* Preliminary results were obtained during the working and interacting sessions of the Intensive Research Program in Discrete, Combinatorial and Computational Geometry. We thank the Centre de Recerca Matemàtica, Universitat Autónoma de Barcelona, for hosting this event and the organizers for providing us with the platform to meet and collaborate. We also thank Jayson Lynch from Massachusetts Institute of Technology and Bodhayan Roy from Masaryk University for useful discussions.

† Partially supported by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 734922.

‡ Supported by cycle 11 of the Tata Consultancy Services (TCS) Research Scholarship.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 19–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

P, we want to find a subset of vertices of maximum cardinality such that every pair of them are hidden to each other.

The MHVS problem, which can also be looked upon as the problem of computing a maximum independent set in the vertex visibility graph of P, is known to be NP-hard [13]. It is in fact not even easy to find an approximate solution. It was shown to be APX-hard by Eidenbenz [5] even in polygons with no holes. Nevertheless, an exact solution can be computed in polynomial time for special classes of simple polygons. A maximum hidden vertex set can be computed in $O(n^2)$ time in a polygon weakly visible from a convex edge [7] (we describe weakly visible polygons in Section 2), and in $O(ne)$ time in the class of so called convex fans, where e is the number of edges of their vertex visibility graph [8]. Heuristic-based algorithms have also been explored which seem to work well in practice, as evidenced by experimental results showing that they provide solutions that are usually quite close to the exact solution for input polygons without holes [1]. Recently, there has also been a study on gender-aware facility location problems [12], which are closely related to the MHVS problem.

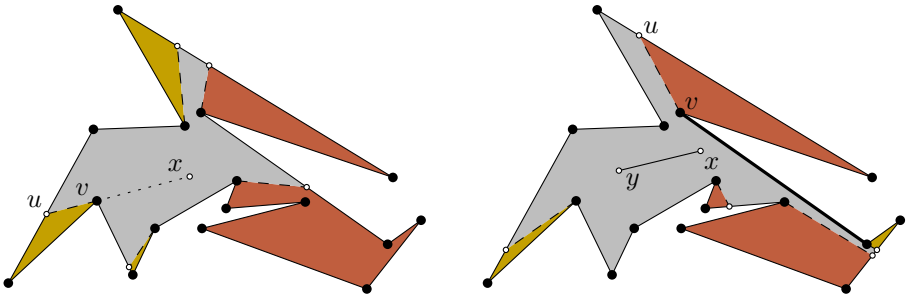
In this paper, we describe a deterministic $O(n^2)$ time algorithm to compute a 1/4-approximation to the maximum hidden vertex set of an n-sided simple polygon with no holes. As far as we are aware, this is the first deterministic algorithm to compute a constant-factor approximation to the optimal solution of the MHVS problem for general simple polygons.

2 Preliminaries

Hereafter, let P be a simple polygon with no holes. For the sake of simplicity, we assume that no three vertices of P are collinear. Given a point x inside P, we denote with $\mathcal{VP}(x)$ the visibility polygon of x. The boundary of $\mathcal{VP}(x)$ is a closed polygonal chain formed by polygonal edges and non-polygonal edges called *constructed edges*. A constructed edge connects a reflex vertex v with a point u lying on an edge of P (see Figure 1a), where the points x, v, and u are collinear.

Let x and y be two points inside P that are visible to each other. A point inside P is said to be *weakly visible* from the line segment $\overline{xy}$, if it is visible from at least one point of $\overline{xy}$. The set of points inside P that are weakly visible from $\overline{xy}$ is called the *weak visibility polygon* of $\overline{xy}$. We denote this polygon with $\mathcal{VP}(\overline{xy})$. Like the visibility polygon of a point, the boundary of $\mathcal{VP}(\overline{xy})$ is a closed polygonal chain formed by polygonal edges and constructed edges. If the boundary of $\mathcal{VP}(\overline{xy})$ contains no constructed edges, then $\mathcal{VP}(\overline{xy}) = P$ and the polygon is said to be *weakly visible* from $\overline{xy}$ (see Figure 1b).

Let ∂P denote the boundary of P. Given two points $a, b \in \partial P$, let $bd(a, b)$ denote the clockwise boundary of P from a to b, so we have $\partial P = bd(a, a) = bd(a, b) \cup bd(b, a)$. Consider a point x inside P and a constructed edge $\overline{vu}$ of $\mathcal{VP}(x)$, where v is a vertex and u is a point on an edge of P. The segment $\overline{vu}$ divides P into two subpolygons: one bounded by $bd(v, u) \cup \overline{vu}$ and the second one bounded by $bd(u, v) \cup \overline{vu}$. Out of these two, the subpolygon that does not contain x is called a *pocket* of $\mathcal{VP}(x)$. We denote this pocket with $P(v, u)$. If x is not contained in the polygon bounded by $bd(v, u) \cup \overline{vu}$, then $\overline{vu}$ is called a *left constructed edge* and $P(v, u)$ is called a *left pocket*. Otherwise, $\overline{vu}$ is called a *right constructed edge* and $P(v, u)$ is called a *right pocket*. The constructed edges and left and right pockets of a weak visibility polygon are defined in a similar way. Examples of these definitions are shown in Figure 1. In particular, in Figure 1a the segment $\overline{vu}$ is a left constructed edge and $P(v, u)$ is a left pocket of $\mathcal{VP}(x)$. On the other hand, in Figure 1b the segment $\overline{vu}$ is a right constructed edge and $P(v, u)$ is a right pocket of $\mathcal{VP}(\overline{xy})$.



(a) The visibility polygon of x. **(b)** The weak visibility polygon of $\overline{xy}$.

Figure 1 A simple polygon along with (a) the visibility polygon of a point, and (b) the weak visibility polygon of a line segment, both in gray. The constructed edges of the visibility polygons are shown in dashed lines, the left pockets in yellow, and the right pockets in red. The polygon is weakly visible from the thick edge.

▶ **Lemma 2.1** (Lemma 2 from Bärtschi et al. [2]). *Let $\overline{v_1u_1}$ and $\overline{v_2u_2}$ be two left constructed edges (or similarly, two right constructed edges) of $\mathcal{VP}(\overline{xy})$, where v_1 and v_2 are reflex vertices of P. Possibly excluding v_1 and v_2, the vertices of P inside the left (or right) pocket $P(v_1, u_1)$ are hidden from the vertices of P inside the left (or right) pocket $P(v_2, u_2)$.*

3 The polygon partition

Our algorithm is based on a link-distance-based partition from Bhattacharya et al. [3] (which is itself adapted from the partitioning method used by Suri [14]) that decomposes the polygon P into a set of disjoint visibility windows. We next outline how this partition is constructed and describe properties that are relevant to our algorithm.

Given two points x and y inside P, the *link distance* from x to y is the minimum number of line segments required in a polygonal chain inside P to connect x to y. The visibility window decomposition is a hierarchical partition of P into visibility polygons, where polygons on the same level contain points at the same link distance from a given vertex p. The first level of the hierarchy is formed by the set $V_1 = \{\mathcal{VP}(p)\}$ that contains the points of P at link distance one from p. Let $\overline{v_1u_1}, \ldots, \overline{v_cu_c}$ be the constructed edges of $\mathcal{VP}(p)$ in clockwise order around p, where v_i is a vertex and u_i is a point lying on an edge of P. The region $P \setminus \mathcal{VP}(p)$ consists of c disjoint polygons we denote with $P_1, \ldots, P_c$. Let $V_{2,i} = \mathcal{VP}(\overline{v_iu_i}) \cap P_i$ be the weak visibility polygon of $\overline{v_iu_i}$ inside P_i. The second level of the hierarchy is formed by the set $V_2 = \{V_{2,1}, \ldots, V_{2,c}\}$ of disjoint weak visibility polygons. The remaining levels are formed by the sets $V_3, V_4, \ldots$ obtained by repeating the previous process until we have a set V_d of disjoint weak visibility polygons with no constructed edges. We thus have that $P = V_1 \cup \cdots \cup V_d = \mathcal{VP}(p) \cup V_{2,1} \cup V_{2,2} \cup \cdots \cup V_{d,1} \cup V_{d,2} \cup \cdots$. The set V_i is formed by the disjoint regions containing the points at link distance i from p, and d is the maximum link distance from p to any point inside P (see Figure 2).

▶ **Lemma 3.1.** *The following statements hold true for the visibility window partition of P:*

i) The vertices of P lying in any subpolygon belonging to V_i are hidden from the vertices of P lying in any subpolygon belonging to V_j, unless $|j - i| \leq 1$.

ii) Let $\overline{uv}$ be a constructed edge of the weak visibility polygon $V_{i,j}$. Then, the constructed edge $\overline{uv}$ is actually a convex edge with respect to every subpolygon $V_{i+1,j} = \mathcal{VP}(\overline{uv}) \cup P_j$.

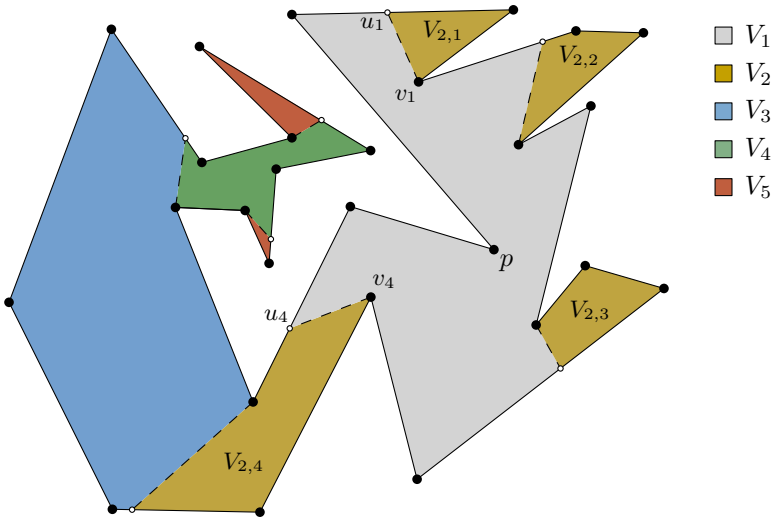


Figure 2 The partition of a simple polygon into weak visibility subpolygons, where each subpolygon consists of points at the same link distance from the vertex p.

Proof. The lemma follows directly from the construction of the partition. For details, please refer to the proof of Lemma 2 from Bhattacharya et al. [3]. ◀

4 The algorithm

We now describe an $O(n^2)$ time 1/4-approximation algorithm for the Maximum Hidden Vertex Set problem in simple polygons. The overall strategy of our algorithm is to decompose P into regions as we described in Section 3, and classify the regions of the partition into four disjoint sets such that vertices of regions belonging to different sets are hidden to each other. We then compute the (exact) maximum hidden set of every region using the algorithm from Ghosh et al. [7], and keep the hidden set with more guards in the same group. Hereafter, we denote by n the number of vertices of P.

1. **Partition the polygon P using link distance**
   Create the decomposition of P based on the link-distance from an arbitrary vertex that we described in Section 3. This partition can be created in $O(n)$ time [3].
2. **Classify visibility windows**
   As P is a simple polygon without any holes, the dual graph of the partition is a tree. Each node of this tree represents a visibility polygon of the partition, and the children of a node are the regions inside the pockets formed by constructed edges belonging to their parent's visibility polygon. Using this tree, we separate the nodes into four disjoint sets in the following manner. First we separate the nodes into two sets: those appearing at odd levels in the tree, and those appearing at even levels in the tree. Then, we further separate the nodes in each of the above sets into two subsets: those created due to a left constructed edge, and those created due to a right constructed edge. At the end of this separation process, we obtain four disjoint subsets, which are as follows:
   - R_1, containing regions at odd levels in the tree created by a left constructed edge
   - R_2, containing regions at odd levels in the tree created by a right constructed edge
   - R_3, containing regions at even levels in the tree created by a left constructed edge
   - R_4, containing regions at even levels in the tree created by a right constructed edge

Observe that the vertices of P inside different regions of the same set are hidden from each other (see Figure 3), either because they belong to non-consecutive levels of the tree (see Lemma 3.1), or because both of them belong to the same level in the tree and are both created by a left (or right) constructed edge (see Lemma 2.1). Also note that the separation process can be completed in $O(n)$ time.

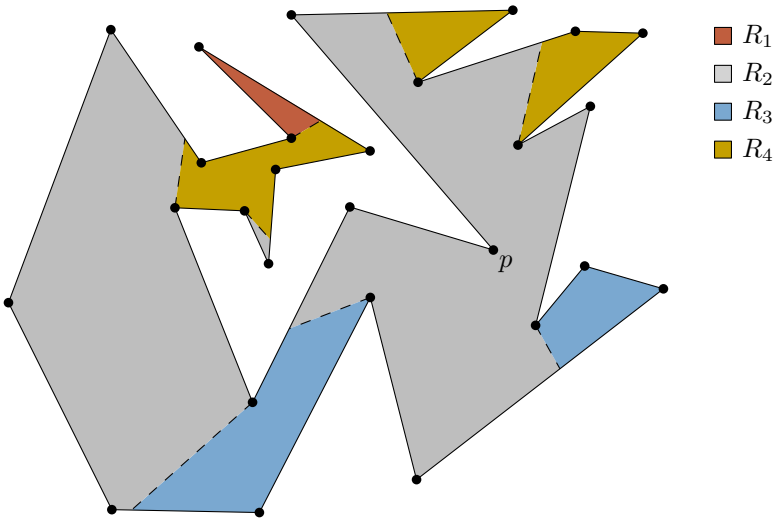


Figure 3 The separation of the regions into four independent sets.

3. Compute an approximate maximum hidden vertex set

Observe that each region of the partition is a polygon which is weakly visible from the constructed edge (of its parent's visibility polygon) that created it. So, within each region, we can compute the (exact) maximum hidden set of vertices using the algorithm by Ghosh et al. [7], which computes the maximum hidden set of a weak visibility polygon with n vertices in $O(n^2)$ time.

Since the vertices of P inside two regions belonging to the same set (from among R_1, R_2, R_3, R_4) are hidden from each other, the union of the maximum hidden sets of the regions in a particular set is a valid hidden set for P. Thus, we can compute four valid hidden sets S_1, S_2, S_3, S_4, that correspond to the sets R_1, R_2, R_3, R_4 respectively, in $O(n^2)$ time. Out of these four valid hidden vertex sets of P, we choose as our approximation of the maximum hidden set the one containing the most number of vertices.

Let $\mathcal{S}^{\text{opt}}$ denote an exact maximum hidden vertex set of P. Also, let $\mathcal{S}^{\text{opt}}_{i,j} \subseteq \mathcal{S}^{\text{opt}}$ denote the subset of vertices that lie within the weak visibility subpolygon $V_{i,j}$ in the partitioning of P. If we denote the exact maximum hidden set computed for each subpolygon $V_{i,j}$ by $\mathcal{S}^{*}_{i,j}$, then observe that $|S^{*}_{i,j}| \geq |S^{\text{opt}}_{i,j}|$. Therefore, we have:

$$|S_1| + |S_2| + |S_3| + |S_4| = \sum_{i,j} |S^{*}_{i,j}| \geq \sum_{i,j} |S^{\text{opt}}_{i,j}| = |\mathcal{S}_{\text{opt}}|$$

$$\max\left(|S_1|, |S_2|, |S_3|, |S_4|\right) \geq \frac{|S_1| + |S_2| + |S_3| + |S_4|}{4} \geq \frac{|\mathcal{S}_{\text{opt}}|}{4}$$

Therefore, by choosing from among $S_1, \ldots, S_4$ the set containing the maximum number of vertices, we obtain a $\frac{1}{4}$-approximation of $\mathcal{S}_{\max}$. Note that Step 3 is the most expensive

step of the algorithm described above, so the algorithm runs in $O(n^2)$ time. This leads us to our main result, which we summarize below.

▶ **Theorem 4.1.** *Given a simple polygon P with n vertices, there exists a $\frac{1}{4}$-approximation algorithm for computing the maximum hidden vertex set in P, which runs in $O(n^2)$ time.*

5 Concluding remarks

We present a $O(n^2)$ time algorithm to compute a 1/4-approximation to the maximum hidden vertex set of a simple polygon with n vertices and no holes. To the best of our knowledge, this is the first constant-factor approximation algorithm for general simple polygons without holes. Observe that our current algorithm cannot be applied when the input polygon P has holes, since then the dual graph of the visibility window partitioning of P is no longer guaranteed to be a tree. However, we are currently investigating possible improvements to our algorithm which could make it work even for input polygons containing holes. Another future research direction is to explore variants of the problem involving restricted-orientation models of visibility, such as rectangular, periscope, or $\mathcal{O}$-visibility.

Acknowledgments

We thank the reviewers for their careful reading and many useful comments, which really helped us to improve the quality and readability of this paper. In particular we thank the reviewer who pointed out to us some interesting references that might motivate future research directions.

References

1 Antonio L. Bajuelos, Samtiago Canales, Gregorio Hernández, and A. Mafalda Martins. Estimating the maximum hidden vertex set in polygons. In *International Conference on Computational Sciences and Its Applications, ICCSA 2018*, pages 421–432, 2008. doi:10.1109/ICCSA.2008.19.

2 Andreas Bärtschi, Subir Kumar Ghosh, Matúš Mihalák, Thomas Tschager, and Peter Widmayer. Improved bounds for the conflict-free chromatic art gallery problem. In *Proceedings of the Thirtieth Annual Symposium on Computational Geometry*, SOCG'14, pages 144:144–144:153. ACM, 2014. doi:10.1145/2582112.2582117.

3 Pritam Bhattacharya, Subir Kumar Ghosh, and Sudebkumar Pal. Constant approximation algorithms for guarding simple polygons using vertex guards. *ArXiv e-prints*, 2017. arXiv:1712.05492v2.

4 Michael F. Cohen and John R. Wallace. *Radiosity and realistic image syntheses*. Academic Press Professional, 1993.

5 Stephan Eidenbenz. Inapproximability of finding maximum hidden sets on polygons and terrains. *Computational Geometry*, 21(3):139–153, 2002. doi:10.1016/S0925-7721(01)00029-3.

6 Olivier Faugeras. *Three-Dimensional Computer Vision*. The MIT Press, 1993.

7 Subir Kumar Ghosh, Anil Maheshwari, Sudebkumar Prasant Pal, Sanjeev Saluja, and C.E. Veni Madhavan. Characterizing and recognizing weak visibility polygons. *Computational Geometry*, 3(4):213–233, 1993. doi:10.1016/0925-7721(93)90010-4.

8 Subir Kumar Ghosh, Thomas Caton Shermer, Binay Kumar Bhattacharya, and Partha Pratim Goswami. Computing the maximum clique in the visibility graph of a simple polygon. *Journal of Discrete Algorithms*, 5(3):524–532, 2007. doi:10.1016/j.jda.2006.09.004.

9 Sumir Kubar Ghosh. *Visibility algorithms in the plane*. Cambridge University Press, 2007.

10 Jean-Claude Latombe. *Robot motion planning*. Kluwer Academic Publishers, 1991.

11 Joseph O'Rourke. *Art gallery theorems and algorithms*. The international series of monographs on Computer Science. Oxford University Press, 1987.

12 Valentin Polishchuk and Leonid Sedov. Gender-Aware Facility Location in Multi-Gender World. In Hiro Ito, Stefano Leonardi, Linda Pagli, and Giuseppe Prencipe, editors, *9th International Conference on Fun with Algorithms (FUN 2018)*, volume 100 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 28:1–28:16, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.FUN.2018.28.

13 T. Shermer. Hiding people in polygons. *Computing*, 42(2):109–131, 1989. doi:10.1007/BF02239742.

14 Subhash Suri. *Minimum link paths in polygons and related problems*. PhD thesis, The Johns Hopkins University, Baltimore, Maryland, 1987.

15 Jorge Urrutia. *Handbook of computational geometry*, chapter Art Gallery and illumination problems, pages 973–1027. Elsevier, 2000.

Skeleton-based decomposition of simple polygons

Maike Buchin[1], Axel Mosig[2], and Leonie Selbach[2]

1 **Technical University Dortmund**
`maike.buchin@tu-dortmund.de`
2 **Ruhr University Bochum**
`Axel.Mosig|Leonie.Selbach@rub.de`

Abstract

In the application of polygon decomposition for the dissection of tissue samples certain constraints on the size and convexity of the subpolygons are given. We present a decomposition method in which different feasibility criteria can be included. Our method is based on a discrete skeleton of the given polygon and can be modified for different optimization problems.

1 Introduction

Polygon decomposition is a common method in algorithmic geometry. Depending on the application different constraints for the shape of the subpolygons are used, for example in triangulations or convex decompositions. We present a skeleton-based decomposition method where cuts are restricted by the skeleton points and various constraints for size or shape can be incorporated. Our work is motivated by a problem that arises in histopathology when dissecting disease-specific subregions from tissue samples using the so-called laser capture microdissection (LCM) [3]. The extraction with LCM is not successful unless the regions fulfill certain conditions based on their size and shape. Hence we develop a method to decompose the regions of interest into smaller parts which all satisfy the given constraints.

1.1 Problem Statement and Solution

Let $\mathcal{P}$ be a simple polygon without holes. We want to find a feasible decomposition Z of $\mathcal{P}$ given some feasibility criteria, where a decomposition Z is *feasible* if every polygon in Z is feasible. Our method is based on the medial axis or skeleton of $\mathcal{P}$ and allows only specific cuts. Because discrete data in form of digital images is given and a discrete output is expected we use discrete skeletons, that is skeletons consisting of a finite set of points resp. pixels. This leads naturally to a finite number of possible cuts we have to consider. Let $\mathcal{S}$ be the skeleton of $\mathcal{P}$ consisting of n skeleton points. If the degree of the skeleton points does not exceed three, a feasible decomposition of $\mathcal{P}$ based on $\mathcal{S}$ can be computed in time $\mathcal{O}(n^k)$, where k is the number of skeleton points with degree one. This holds also for the minimum number problem and the minimum edge length problem that is minimizing the number of subpolygons in the decomposition or the total length of inserted cuts.

Here we will disregard the actual computation of the feasibility and assume that the feasibility of a subpolygon can be tested efficiently. In our research we consider criteria such as size or (approximate) convexity. In both cases we can compute those values for all adjacent cuts beforehand in time $\mathcal{O}(m)$ for m being the number of boundary points of the input polygon. Given the information for adjacent cuts we can iteratively generate all information needed during the execution of our algorithm in constant time. For other feasibility criteria this may not be the case, in which case this would need to be included in the overall runtime.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

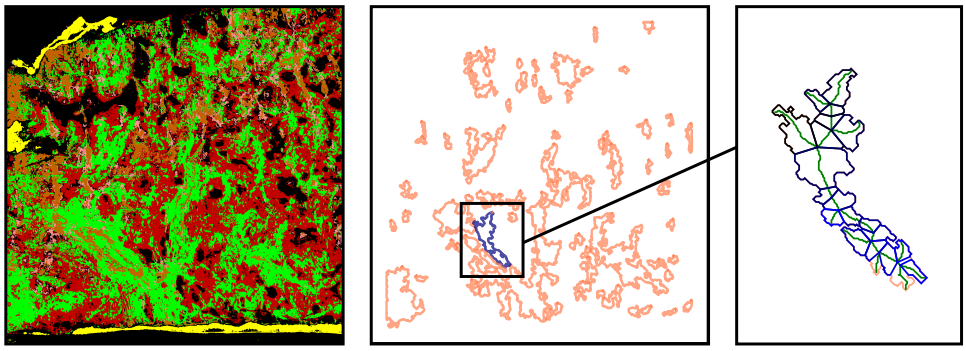


Figure 1 Decomposition in a histopathological tissue sample given as a classified image (left). This decomposition was generated using the algorithm from [7] using area for feasibility.

1.2 Basic Definitions

Let $D \subset \mathbb{R}^2$ be a connected bounded domain. The *medial axis* or *skeleton* $S(D)$ of the set D is the locus of centers of maximal disks in D. A *maximal disk* B in D is a closed disk contained in D such that every other disk containing B is not contained in D. Let s be the center of a maximal disk $B(s)$, s is called a *skeleton point*. We define the *contact set* of s as $\mathcal{C}(s) = B(s) \cap \partial D$. A connected component of $\mathcal{C}(s)$ is called a *contact component* of s and the elements of $\mathcal{C}(s)$ are called *contact points*. The *degree* of a skeleton point is defined as the number of its contact components. A skeleton S is given as a graph consisting of connected arcs S_k which are called *skeleton branches*. Skeleton branches meet at skeleton points of degree three or higher. We call these points *branching points*.

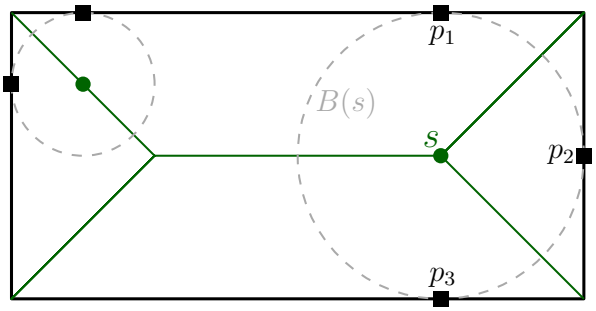


Figure 2 A skeleton point s with its maximal disk $B(s)$ and contact points $p_1, p_2, p_3 \in \mathcal{C}(s)$.

In our application we consider digital images where we interpret a given object as a polygon by defining each boundary pixel as a corner vertex. We skeletonize the polygon resulting in a simplified discrete skeleton using the method from [2]. The discrete skeleton consists of pixels but fulfills some of the basic properties of the medial axis such that contact components are given. In the skeleton-based decomposition of a polygon $\mathcal{P}$ the cuts are restricted to line segments connecting a skeleton point to a contact point. The cuts induce subpolygons between two or more consecutive skeleton points. $P_k(i, j)$ denotes a polygon generated by two skeleton points i, j on the same skeleton branch S_k as shown in Figure 3.

Since a branching point belongs to more than one branch and has at least three contact points those two points corresponding to the considered branch are chosen, see Figure 4 (a). Notice that a polygon generated by more than two skeleton points can always be represented as a union of subpolygons generated by two skeleton points. See Figure 4 (b) for an example.

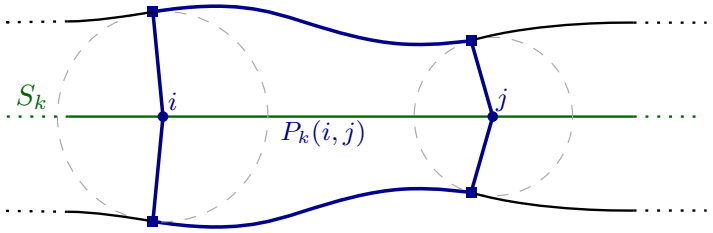
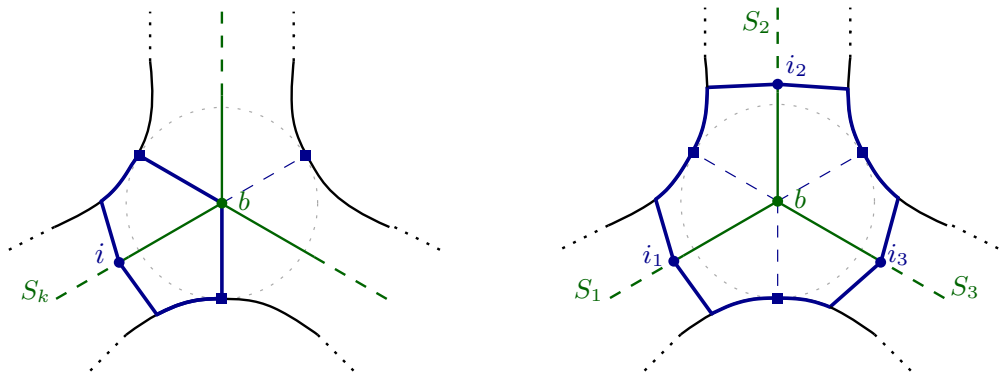


Figure 3 Polygon $P_k(i,j)$ generated by two skeleton points i,j on the same branch S_k.



(a) In blue: $P_k(b,i)$.

(b) In blue: $P_1(b,i_1) \cup P_2(b,i_2) \cup P_3(b,i_3)$.

Figure 4 Subpolygons at a branching point (left) and generated by three skeleton points (right).

1.3 Related Results from the Literature

Skeletons are used in many applications such as object recognition, medical image analysis and shape decomposition [6]. Leonard et al. [4] use the medial axis for the decomposition of 2D objects to determine a parts hierarchy. Simmons and Séquin [9] compute a hierarchical decomposition of an object using the related axial shape graph. Tănase and Veltkamp [10] use the straight line skeleton to compute decompositions of polygonal shapes into possibly overlapping parts. There are several methods which use a one-dimensional curve skeleton of 3D shapes. Reniers and Telea [5] use the curve skeleton for the segmentation of 3D shapes into meaningful components. Serino et al. [8] propose a method for decomposing a 3D object by using a polygonal approximation of the curve skeleton.

2 Decomposition algorithms

For a polygon $\mathcal{P}$ without holes the skeleton S is given as an acyclic graph. We represent S as a tree T. For this we pick an arbitrary branching point as root r. All other vertices are labeled v_k and correspond to a skeleton branch S_k. We define $C(v)$ as the set of children of a vertex v. This tree gives us a chronological order of how to work our way through the skeleton. The skeleton points on each branch S_k are labeled from top to bottom – according to the chosen tree representation T – starting with 1 at the top. See Figure 5 for an example.

Before we describe our general decomposition method we consider a special case – which is discussed by Selbach in [7] – where we decompose the polygon by considering each branch of the skeleton on its own. In this case we only have to deal with linear skeletons.

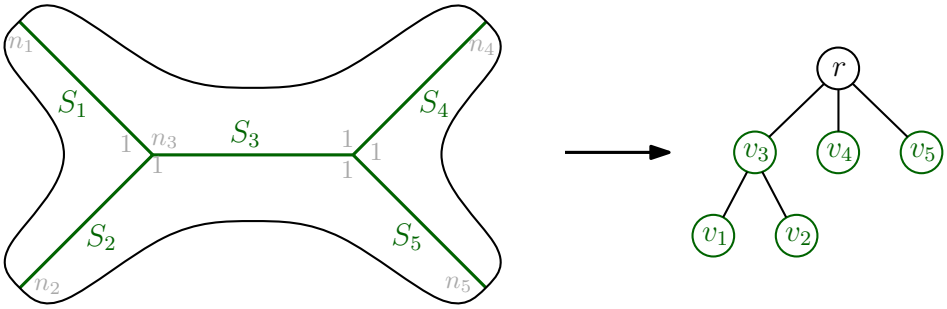


Figure 5 A representation of a skeleton with two branching points as a tree.

2.1 Decomposition based on linear skeletons

Given a polygon $\mathcal{P}_k$ belonging to a skeleton branch S_k with a linear skeleton of size n_k, i.e. $\mathcal{P}_k = P_k(1, n_k)$. A feasible decomposition can be found by dynamic programming, using an array X_k such that $X_k(i)$ equals `True` if there exists a feasible decomposition of $P_k(i, n_k)$.

$$X_k(i) = \begin{cases} \texttt{True} & \text{if } \exists\, j : i < j \leq n_k \text{ s.t. } P_k(i,j) \text{ is feasible and } X_k(j) = \texttt{True}. \\ \texttt{False} & \text{else.} \end{cases}$$

We can adjust the formula of $X_k(i)$ easily to solve different optimization problems. By defining $X_k(i)$ as $\min_{i<j\leq n_k} X_k(j) + 1$ resp. $r(i)$, where $P_k(i,j)$ is feasible, we can solve the minimum number problem resp. the minimum edge length problem.

This results in an $\mathcal{O}(n^2)$ time algorithm [7] for computing a feasible decomposition of a polygon with a linear skeleton. Note that for certain combinations of (simple) feasibility criteria and optimization goals decomposing polygons with linear skeletons is closely related to segmentation and can be done more efficiently, see for example [1].

2.2 General decomposition

In the following we restrict ourselves to skeletons where the degree of the skeleton points does not exceed three. If there are m branching points, there will be $m + 2$ end points, namely skeleton points of degree one. We now consider decompositions consisting of subpolygons which can be generated by more than two skeleton points. As stated above those polygons can be represented as a union of subpolygons that are generated by two skeleton points. Notice that the largest number of skeleton points generating a polygon is equal to the number of leaf vertices in the skeleton tree.

The decomposition problem can be solved using a bottom-up approach in the skeleton tree. We will present the method for the minimum number problem, but as before our method can be modified for other optimization problems. The general idea is that we compute for each skeleton point the size of a minimal feasible decomposition of the subpolygon up to this point. For this, we compute entries $X_k(i)$ for all $i \in S_k$ for every vertex v_k working our way up the tree T. Here $X_k(i)$ is the number of polygons in the minimal feasible decomposition of the polygon $P_k(i)$ corresponding to skeleton point i on branch S_k, see Figure 6. We eventually compute the value X_r for the root vertex, which is defined as the number of polygons in the minimal feasible decomposition of the entire polygon.

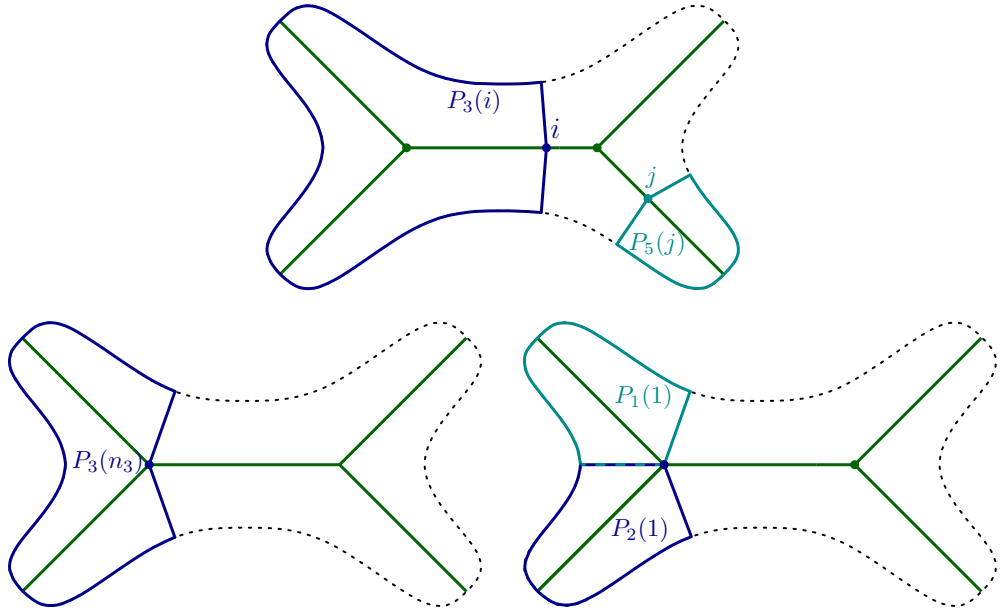


Figure 6 Different subpolygons according to the tree representation given in Figure 5.

For computing the entry $X_k(i)$ observe that a minimal feasible decomposition of $P_k(i)$ consists of a feasible subpolygon P ending at i and minimal feasible decompositions of the connected components of $P_k(i) \backslash P$. Hence to compute $X_k(i)$ we search over all possible combinations of cuts, i.e. skeleton points, in the corresponding subpolygon $P_k(i)$ that together with i generate a feasible polygon P, and return the minimal size at these.

To do this we use a function $Q[s, I, a, P]$, which searches through the subtree rooted at s. The parameter I is a set of vertices which corresponds to the currently considered skeleton branches, on which we are searching for cuts – starting at index s. In P the generated polygon is stored and updated as the search continues. The parameter a is the size of minimal decomposition where cuts have already been chosen. Hence initially, P is empty, $a = 0$, $s = i$ and $I = \{v_k\}$ for a skeleton point i on branch S_k. Now for every vertex $v_k \in I$ we have two choices: Either we cut on the branch S_k and check the possible cuts on the other branches in $I \setminus \{v_k\}$. Or we continue the search in the subtree of v_k by including the children $C(v_k)$ into the set of considered branches – in this case the generated polygon contains the whole subpolygon $P_k(s, n_k)$. For the computation of $Q[s, I, a, P]$ we choose an arbitrary vertex $v_k \in I$ for the first iteration. The function is then defined as follows:

$$Q[s, I, a, P] = \min \begin{cases} \min\limits_{j \geq s \in S_k} Q[1, I \setminus \{v_k\}, a + X_k(j), P \cup P_k(s, j)], & \text{(1a)} \\ Q[1, (I \setminus \{v_k\}) \cup C(v_k), a, P_k(s, n_k)] & \text{(1b)} \end{cases}$$

$$Q[1, \emptyset, a, P] = \begin{cases} a + 1 & \text{if } P \text{ is feasible.} \\ \infty & \text{else.} \end{cases}$$

We define $X_k(i) = Q[i, \{v_k\}, 0, \emptyset]$ and $X_r = Q[1, C(r), 0, \emptyset]$. Notice that when $s = 1$ we search over all $j \in S_k$. This is the case in every iteration except for the initial one.

▶ **Example 2.1** (Two branching points)**.** In case of two branching points we have three different cases of computation to consider, see Figure 7 for an illustration.

1. Leaf vertex: For computation of the entries of X_k for $k = 1, 2, 4, 5$ the equation for $X_k(i)$ equals the formula for the linear skeleton.

2. Inner vertex: A feasible decomposition of the polygon up to the skeleton point i on the branch S_3 contains a polygon generated by either i and some $j > i$ on branch S_3 or i and two points $(i_1, i_2) \in S_1 \times S_2$. Where the first case is calculated by (1a) as for linear skeletons and the second is calculated by (1b) as follows:

$$\begin{aligned}
&Q[1, \{v_1, v_2\}, 0, P_3(i, n_3)] \\
&= \min_{i_1 \in S_1} Q[1, \{v_2\}, X_1(i_1), P_3(i, n_3) \cup P_1(1, i_1)] \\
&= \min_{(i_1, i_2) \in S_1 \times S_2} Q[1, \emptyset, X_1(i_1) + X_2(i_2), P_3(i, n_3) \cup P_1(1, i_1) \cup P_2(1, i_2)] \\
&= \min_{(i_1, i_2) \in S_1 \times S_2} \{X_1(i_1) + X_2(i_2) + 1 \mid P_3(i, n_3) \cup P_1(1, i_1) \cup P_2(1, i_2) \text{ is feasible}\}
\end{aligned}$$

3. Root vertex: For the root we compute:

$$X_r = Q[1, \{v_3, v_4, v_5\}, 0, \emptyset] \; = \min \begin{cases} \min\limits_{i_3 \in S_3} Q[1, \{v_4, v_5\}, X_3(i_3), P_3(1, i_3)] \\ Q[1, \{v_1, v_2, v_4, v_5\}, 0, P_3(1, n_3)] \end{cases}$$

This results in a calculation of the minimum feasible decomposition size over $(i_3, i_4, i_5) \in S_3 \times S_4 \times S_5$ for the first case and $(i_1, i_2, i_3, i_4) \in S_1 \times S_2 \times S_4 \times S_5$ for the second.

▶ **Theorem 2.2.** *Let $\mathcal{P}$ be a polygon with skeleton S. Let S consist of n skeleton points with degree less or equal 3. A feasible decomposition of $\mathcal{P}$ based on S can be computed in time $\mathcal{O}(n^k)$, where k is the number of leaves in the skeleton tree (or skeleton points with degree 1).*

Proof. We argue the runtime by assigning weights to the skeleton tree. The weight of a vertex v is $g(v) = |S_v| + \prod_{w \in C(v)} g(w)$, which is the maximal number of skeleton points considered in the computation of one $X_v(i)$. The first part of this sum corresponds to (1a) and the second part to (1b). The computation of $X_v(i)$ for all $i \in S_v$ takes $|S_v| \cdot g(v)$ time. Let $L(v)$ be the number of leaves in the subtree with root v. We can show that $g(v) = \mathcal{O}(n^{L(v)})$. The overall runtime is asymptotically dominated by the computation of X_r which takes time

$$|S_r| \cdot g(r) = g(r) = \mathcal{O}(n^{L(r)}) = \mathcal{O}(n^k). \tag{2}$$

Sketch of correctness: As we compute $X_k(i)$: A feasible decomposition of the considered subpolygon up to skeleton point i consists of a feasible polygon P generated by the skeleton point i and feasible decompositions of the remaining polygon (or polygons). The polygon P is either generated by another skeleton point on the branch S_k – as computed by (1a) – or by some other skeleton points in the subtree of v_k – as computed recursively by (1b). ◀

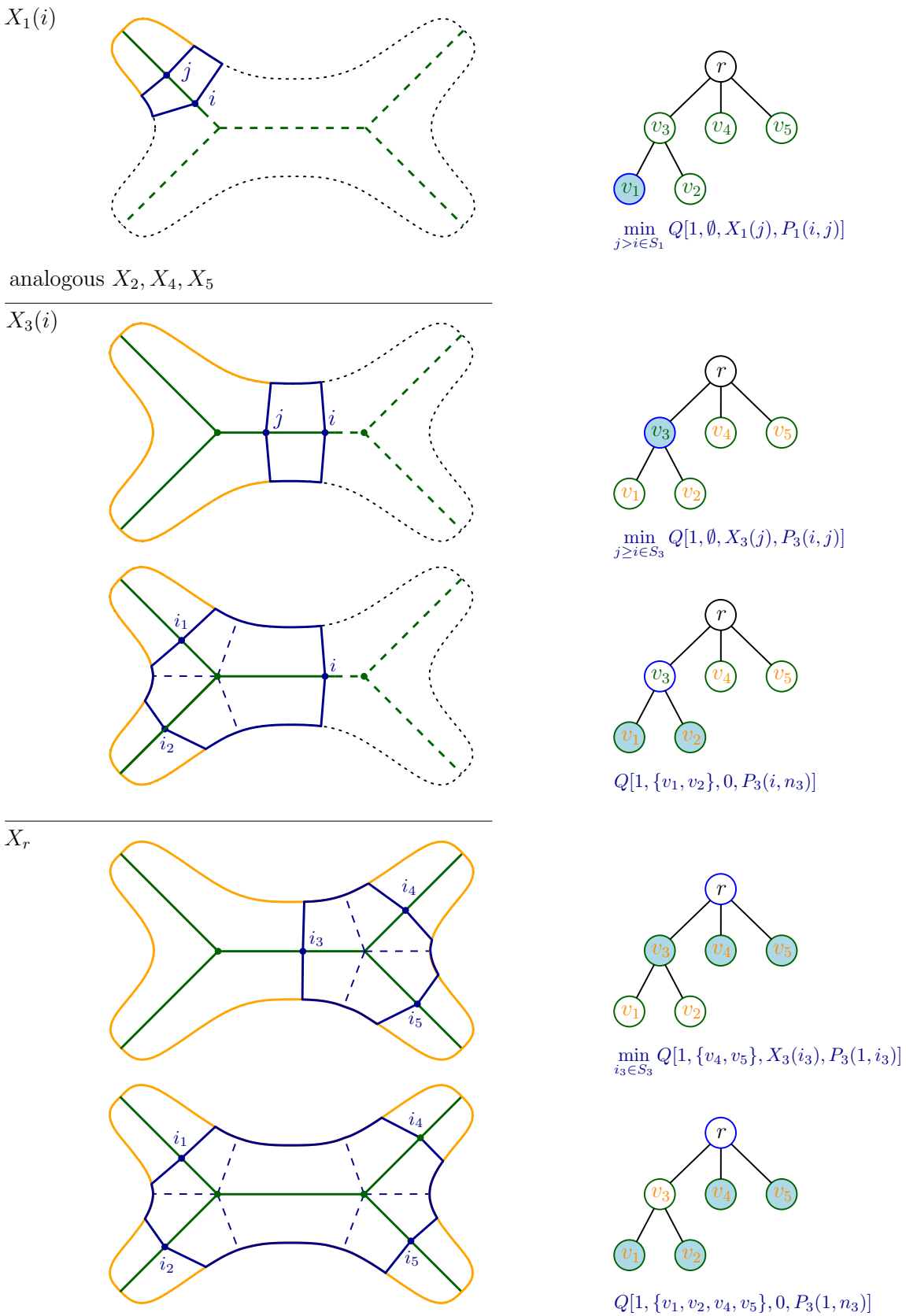


Figure 7 Decomposition of a polygon with two branching points. The non-dashed polygon is the currently considered subpolygon and the polygon generated in a certain iteration is shown in blue.

3 Conclusion

We presented a method to find an optimal feasible decomposition of a simple polygon based on a discrete skeleton, which allows to include different feasibility criteria. The algorithm will be further implemented and analysed in the application. In practice we do not expect the runtime to be $\mathcal{O}(n^k)$ – as we may stop the iteration if P is no longer feasible. Also we use a pruned skeleton which means that have some control over the factor k – for the tissue sample in Figure 1 it was $k < 10$. We are currently working on an algorithm for higher degrees. Also we are looking into the similarities to tree resp. graph decomposition/partition problems. And there is the question if our methods can be adjusted for polygons with holes.

References

1 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, Dallas/Fort Worth, TX, USA, November 4-7, 2014*, pages 351–360, 2014.

2 Xiang Bai, Longin Jan Latecki, and Wen-Yu Liu. Skeleton pruning by contour partitioning with discrete curve evolution. *IEEE transactions on pattern analysis and machine intelligence*, 29(3), 2007.

3 Frederik Großerueschkamp, Thilo Bracht, Hanna C Diehl, Claus Kuepper, Maike Ahrens, Angela Kallenbach-Thieltges, Axel Mosig, Martin Eisenacher, Katrin Marcus, Thomas Behrens, et al. Spatial and molecular resolution of diffuse malignant mesothelioma heterogeneity by integrating label-free ftir imaging, laser capture microdissection and proteomics. *Scientific reports*, 7:44829, 2017.

4 Kathryn Leonard, Geraldine Morin, Stefanie Hahmann, and Axel Carlier. A 2d shape structure for decomposition and part similarity. In *Pattern Recognition (ICPR), 2016 23rd International Conference on*, pages 3216–3221. IEEE, 2016.

5 Dennie Reniers and Alexandru Telea. Skeleton-based hierarchical shape segmentation. In *Shape Modeling and Applications, 2007. SMI'07. IEEE International Conference on*, pages 179–188. IEEE, 2007.

6 Punam K Saha, Gunilla Borgefors, and Gabriella Sanniti di Baja. A survey on skeletonization algorithms and their applications. *Pattern Recognition Letters*, 76:3–12, 2016.

7 Leonie Selbach. Algorithmen zur Zerlegung von einfachen Polygonen unter Nebenbedingungen. Master's thesis, Ruhr-Universität Bochum, 2018.

8 Luca Serino, Carlo Arcelli, and Gabriella Sanniti di Baja. Decomposing 3d objects in simple parts characterized by rectilinear spines. *International Journal of Pattern Recognition and Artificial Intelligence*, 28(07):1460010, 2014.

9 Maryann Simmons and Carlo H Séquin. 2d shape decomposition and the automatic generation of hierarchical representations. *International Journal of Shape Modeling*, 4(01n02):63–78, 1998.

10 Mirela Tănase and Remco C Veltkamp. Polygon decomposition based on the straight line skeleton. In *Geometry, Morphology, and Computational Imaging*, pages 247–268. Springer, 2003.

Recognizing Visibility Graphs of Polygons with Holes

Hossein Boomari[1], Mojtaba Ostovari[2], and Alireza Zarei[3]

1 **Mathematical Science Faculty, Sharif University of Technology**
`h.boomari1@student.sharif.ir`
2 **Mathematical Science Faculty, Sharif University of Technology**
`mojtaba.ostovari@alum.sharif.ir`
3 **Mathematical Science Faculty, Sharif University of Technology**
`zarei@sharif.ir`

Abstract

The visibility graph of a polygon corresponds to its internal diagonals and boundary edges. For each vertex on the boundary of the polygon, we have a vertex in this graph and if two vertices of the polygon see each other there is an edge between their corresponding vertices in the graph. Two vertices of a polygon see each other if and only if their connecting line segment completely lies inside the polygon, and they are externally visible if and only if this line segment completely lies outside the polygon. Recognizing visibility graphs is the problem of deciding whether there is a simple polygon whose visibility graph is isomorphic to a given input graph. This problem is well-known and well-studied, but yet widely open in geometric graphs and computational geometry.

Existential Theory of the Reals is the complexity class of problems that can be reduced to the problem of deciding whether there exists a solution to a quantifier-free formula $F(X_1, X_2, ..., X_n)$, involving equalities and inequalities of real polynomials with real variables. The complete problems for this complexity class are called $\exists\mathbb{R}$-*Complete*.

In this paper, we show that recognizing visibility graphs of polygons with holes is $\exists\mathbb{R}$-*Complete*.

1 Introduction

The visibility graph of a simple planar polygon is a graph in which there is a vertex for each vertex of the polygon and for each pair of visible vertices of the polygon there is an edge between their corresponding vertices in this graph. Two points in a simple polygon are visible from each other if and only if their connecting segment completely lies inside the polygon. In this definition, each pair of adjacent vertices on the boundary of the polygon are assumed to be visible from each other. This implies that we always have a Hamiltonian cycle in a visibility graph which determines the order of vertices on the boundary of the corresponding polygon. A polygon with holes has some non-intersecting holes inside the boundary of the polygon. In these polygons the area inside a hole is considered as the outside area and internal and external visibility graphs of such polygons are defined in the same way as defined for simple polygons. In the visibility graph of a polygon with holes, we have the sequence of vertices corresponding to the boundary of each hole, as well.

Computing the visibility graph of a given simple polygon has many applications in computer graphics [19], computational geometry [11] and robotics [2]. There are several efficient polynomial time algorithms for this problem [11].

This concept has been studied in reverse as well: Is there any simple polygon whose visibility graph is isomorphic to a given graph, and, if there is such a polygon, is there any way to reconstruct it(finding positions for its vertices in the plane)? The former problem is

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 19–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

known as recognizing visibility graphs and the latter one is known as reconstructing a polygon from a visibility graph. The computational complexity of these problems are widely open. The only known result about the computational complexity of these problems are that they belong to *PSPACE* [7] complexity class. More precisely, they belong to the class of *Existential theory of the reals* [15]. This means that it is not even known whether these problems are in *NP* or can be solved in polynomial time. Even, if we are given the Hamiltonian cycle of the visibility graph which determines the order of vertices on the boundary of the target polygon, the exact complexity classes of these problems are still unknown.

However, these problems have been solved efficiently for special cases of *tower* and *spiral polygons*. The recognizing and reconstruction problems have been solved for tower polygons [6] and spiral polygons [8] in linear time in terms of the size of the graph.

Although there is some progress on recognizing and reconstruction problems, there have been plenty of studies on characterizing visibility graphs. In 1988, Ghosh introduced three necessary conditions for visibility graphs and conjectured their sufficiency [9]. In 1990, Everett proposed a graph that rejects Ghosh's conjecture [7]. She also refined Ghosh's third necessary condition to a new stronger one [10]. In 1992, Abello *et al.* built a graph satisfying Ghosh's conditions and the stronger version of the third condition which was not the visibility graph of any simple polygon [1], disproving the sufficiency of these conditions. In 1997, Ghosh added his forth necessary condition and conjectured that this condition along with his first two conditions and the stronger version of the third condition are sufficient for a graph to be a visibility graph. Finally, in 2005 Streinu proposed a counter example for this conjecture [18].

Existential theory of the reals ($\exists\mathbb{R}$) is a complexity class that was implicitly introduced in 1989 [3], introduced by Shor in 1991 [17] and explicitly defined by Schaefer in 2009[16]. It is the complexity class of problems which can be reduced to the problem of deciding, whether there is a solution for a Boolean formula $\phi : \{True, False\}^n \to \{True, False\}$ in propositional logic, in the form $\phi(F_1(X_1, X_2, ..., X_N), F_2(X_1, X_2, ..., X_N), ..., F_n(X_1, X_2, ..., X_N))$, where each $F_i : \mathbb{R}^N \to \{True, False\}$ consists of a polynomial function $G_i : \mathbb{R}^N \to \mathbb{R}$ on some real variables, compared to 0 with one of the comparison operators in $\{<, \leq, =, >, \geq\}$ (for example $G_i(X_1, X_2) = X_1^3X_2^2 - X_1X_2^3$ and $F_i(X_1, X_2) \equiv G_i(X_1, X_2) < 0$). Clearly, satisfiability of quantifier free Boolean formulas belong to $\exists\mathbb{R}$. Therefore, $\exists\mathbb{R}$ includes all NP problems. In addition, $\exists\mathbb{R}$ belongs to $PSPACE$ [5] and we have $NP \subseteq \exists\mathbb{R} \subseteq PSPACE$. Many other decision problems, especially geometric problems, belong to $\exists\mathbb{R}$ and some are complete for this complexity class. Recognizing *LineArrangement* (*Stretchability*), simple order type, intersection graphs of segments, recognizing visibility graphs of a point set, and intersection graphs of unit disks in the plane are some problems which are complete for $\exists\mathbb{R}$ or simply $\exists\mathbb{R}$-*Complete* [5]. The computational complexity of these problems was open for years and after proving $\exists\mathbb{R}$-*Completeness*, the study of the $\exists\mathbb{R}$ class and $\exists\mathbb{R}$-*Complete* problems gets more attention in computational geometry literature. We discuss the problem, Recognizing *LineArrangement* (*Stretchability*), in more details in this paper in Section 2.

In this paper, we show that recognizing a visibility graph of polygon with holes is $\exists\mathbb{R}$-*Complete*. In this problem we assume that the sequence of vertices corresponding to the boundary of the polygon and its holes, is given as input[1]

[1] While (in Dec-2017) we submitted this result to SOCG2018 and later submitted it to arXiv in Apr-2018[4], in an independent work by Hoffmann and Merckx[13] in Jan-2018 they used another technique to prove the $\exists\mathbb{R}$-*Completeness* of recognizing the visibility graphs of polygon with holes. First, they proved the $\exists\mathbb{R}$-*Completeness* of recognizing the *AllowableSequences* and then reduced this problem to recognizing the visibility graphs of polygon with holes.

2 Preliminaries and Definitions

2.1 Line arrangement and stretchability

Considering a set of lines in the plane, the problem of describing their arrangement is called *LineArrangement*. This is an important and fundamental problem in combinatorics and a well-studied problem in computational geometry. This description for a set of lines $l_1, l_2, ..., l_n$ consists of their vertical order with respect to a vertical line to the left of all their intersections, and for each line l_i, the order of lines that are intersected by l_i when we traverse l_i from left to right (we assume that none of the input lines l_i is vertical). Recognizing whether there can be a set of lines in the plane with the given *LineArrangement*, is called *Recognizing LineArrangement* or simply *LineArrangement* problem. When the lines are in general position (all pairs of lines intersect and no 3 lines intersect at the same point) the problem is called *SimpleLineArrangement*. It has been proved that *SimpleLineArrangement* is $\exists\mathbb{R}$-*Complete* [5, 14].

A pseudo-line is a monotone curve with respect to the X axis. Assuming that no pair of pseudo-lines intersect each other more than once, we can describe an instance of recognizing *PseudoLineArrangement* problem in the same way as we did for *LineArrangement*. However, Recognizing *PseudoLineArrangement* belongs to the P complexity class and it can be decided with a Turing machine in polynomial time [12]. A pseudo code implementation and the details of this algorithm has been given in [4] and depicted in Figure 1.

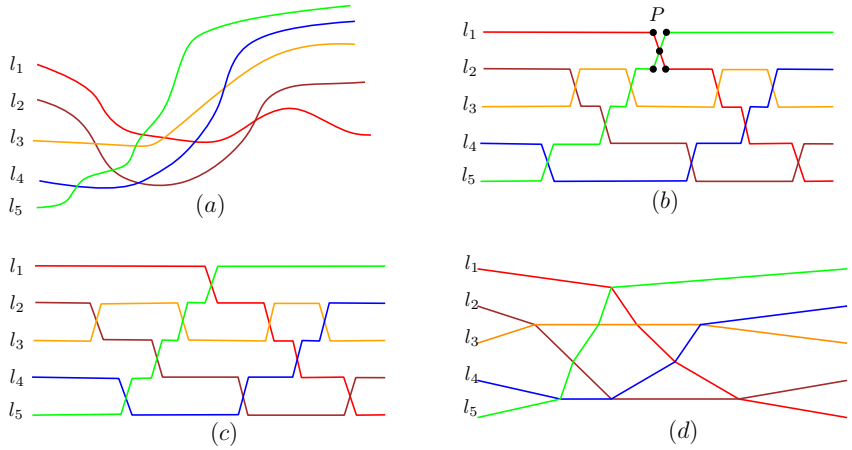


Figure 1 The reconstruction algorithm for *PseodoLineArrangement*.

Trivially, if an instance of the *LineArrangement* problem is realizable, it has a *PseudoLineArrangement* realization as well. On the other hand, if an instance of the *PseudoLineArrangement* problem has a realization in which all segments of each pseudo-line lie on the same line, the input instance has also a *LineArrangement* realization as well.

Therefore, we can describe the *LineArrangement* problem as follows:

- Is it possible to stretch a *PseudoLineArrangement* of a given line arragement description such that each pseudo-line lies on a single line?

This problem is known as *Stretchability*. As stated before, pseudo-line arragement belongs to the P complexity class and can be recognized and reconstructed efficiently. Therefore, $\exists\mathbb{R}$-*Completeness* of *LineArrangement* implies that *Stretchability* is $\exists\mathbb{R}$-*Complete*.

2.2 Visibility graph of a polygon with holes

A polygon with holes is a simple polygon that has a set of non-colliding areas (simple polygons) inside it. The internal areas of the holes belong to the outside area of the polygon. In these polygons, two vertices are visible from each other if their connecting segment completely lies inside the polygon. The visibility graph of a polygon with holes is a graph whose vertices correspond to the vertices of the polygon and the holes, and in this graph there is an edge between two vertices if and only if their corresponding vertices in the polygon are visible from each other (see Fig. 2). In this paper, we assume that along with the visibility graph, we have the cycles that correspond to the order of vertices on the boundary of the polygon and the holes. The cycle that corresponds to the external boundary of the polygon is called the external cycle(see Fig. 2).

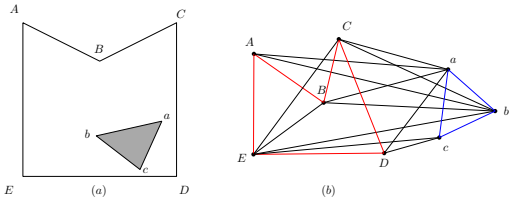


Figure 2 A polygon with one hole (a), and its visibility graph (b).

3 Complexity of Recognizing Visibility Graphs of Polygons with Holes

In this section, we show that recognizing a visibility graph of polygon with holes is $\exists\mathbb{R}$-*Complete*. This is done by reducing an instance of the stretchability problem to an instance of this problem.

In Section 2.1 we showed that we can describe the line arragement problem as an instance of stretchability of pseudo-lines in which each pseudo-line is composed of a chain of segments and the break-points of these chains(except the first and the last endpoints of the chains) correspond to the intersection points of the pseudo-lines. We build a visibility graph $\mathcal{G}$, an external cycle $\mathcal{P}$, and a set of boundary cycles $\mathcal{H}$ from an instance of such a stretchability problem, and prove that the pseudo-line arragement is stretchable in the plane if and only if there exists a polygon with holes whose visibility graph is $\mathcal{G}$, its external cycle is $\mathcal{P}$ and the set of boundary cycles of its holes is $\mathcal{H}$.

Assume that $(\mathcal{L}, \mathcal{S})$ is an instance of the stretchability problem where, as described in [4], $\mathcal{L} = \langle l_1, l_2, ..., l_n \rangle$ is the sequence of the pseudo-lines and $\mathcal{S} = \langle S_1, S_2, ..., S_n \rangle$ is the sequence of the intersections of these pseudo-lines in which $S_i = \langle l_{a(i,1),...,a(i,n-1)} \rangle$ is the order of lines intersected by l_i. Let denote by $(\mathcal{G}, \mathcal{P}, \mathcal{H})$ the corresponding instance of the visibility graph realization in which $\mathcal{G}$ is the visibility graph, $\mathcal{P}$ is the external cycle of the outer boundary of the polygon and $\mathcal{H} = \{H_1, H_2, ..., H_k\}$ is the set of boundary cycles of its holes. To build this instance, consider an example of such an $(\mathcal{L}, \mathcal{S})$ instance shown in Fig. 3-a. This figure shows a pseudo-line realization obtained from the pseudo-line reconstruction algorithm for an instance of four pseudo-lines. If this instance is stretchable, like the one shown in Fig. 3-b, we can build a polygon with holes like the one shown in Fig. 3-c. The outer boundary of this polygon and the boundary of its holes lie along a set of convex curves connecting the endpoints of each stretched pseudo-line. Precisely, for each stretched pseudo-line l_i, as in Fig. 3-b, there is a pair of convex chains on both of its sides which connect its endpoints. These pair of convex chains are sufficiently close to their corresponding stretched pseudo-lines, and their

break-points are the intersection points of these chains(like point o in Fig. 3-c). This pair of convex chains, for each pseudo-line l_i, makes a convex polygon which is called its channel and is denoted by $Ch(l_i)$. The outer boundary of the target polygon and the boundary of its holes are obtained by removing those segments of the chains that lie inside another channel (see Fig. 3-c). Note that, we do not have the stretched realization of $(\mathcal{L}, \mathcal{S})$ instance of the stretachability problem. But, from the pseudo-line realization, we can determine $\mathcal{G}$, $\mathcal{P}$ and $\mathcal{H}$ of the corresponding instance $(\mathcal{G}, \mathcal{P}, \mathcal{H})$ in polynomial time. As shown in Fig. 3-d, $\mathcal{P}$ and $\mathcal{H}$ are obtained by imaginary drawing a channel for each pseudo-line l_i. Finally, the vertex set of graph $\mathcal{G}$ is the set of all break-points of these convex chains, and, two vertices are connected by an edge if and only if they belong to the boundary of the same channel. The following theorem shows the relationship between $(\mathcal{L}, \mathcal{S})$ and $(\mathcal{G}, \mathcal{P}, \mathcal{H})$ problem instances. The detailed proof of the theorem is given in [4].

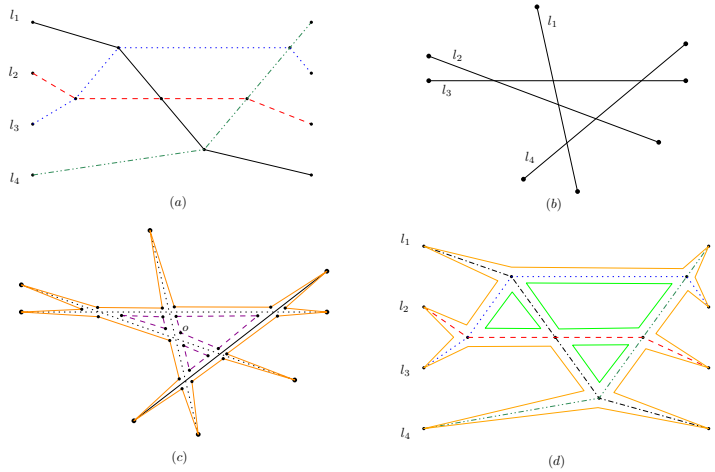


Figure 3 A polygon with holes which is constructed from an instance of the *PseudoLineArrangement* problem.

▶ **Lemma 1.** *An instance $(\mathcal{L}, \mathcal{S})$ of the stretchability problem is realizable if and only if its corresponding $(\mathcal{G}, \mathcal{P}, \mathcal{H})$ instance of the visibility graph is realizable.*

It is easy to show that recognizing a visibility graph of a polygon with holes belongs to $\exists\mathbb{R}$. It can be done by constructing a set of boolean formulas on a set of functions $F_i : \mathbb{R} \to \mathbb{R}$ on the set of vertices (a pair of two real numbers) of the polygon with hole, that verifies the visibility constrains in it. While the stretchability problem is $\exists\mathbb{R}$-*Complete* and our reduction is polynomial, Theorem 1 implies the $\exists\mathbb{R}$-*Hardness* of recognizing visibility graph of polygon with holes. Therefore, we have the following theorem.

▶ **Theorem 2.** *Recognizing visibility graph of polygon with holes is $\exists\mathbb{R}$-Complete.*

4 Conclusion

In this paper, we showed that the visibility graph recognition problem is $\exists\mathbb{R}$-*Complete* for polygons with holes.

References

1 James Abello, Hua Lin, and Sekhar Pisupati. On visibility graphs of simple polygons. *Congressus Numerantium*, pages 119–119, 1992.

2 Calin Belta, Volkan Isler, and George J Pappas. Discrete abstractions for robot motion planning and control in polygonal environments. *IEEE Transactions on Robotics*, 21(5):864–874, 2005.

3 Lenore Blum, Mike Shub, and Steve Smale. On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *Bulletin of the American Mathematical Society*, 21(1):1–46, 1989.

4 Hossein Boomari, Mojtaba Ostovari, and Alireza Zarei. Recognizing visibility graphs of polygons with holes and internal-external visibility graphs of polygons. *arXiv preprint arXiv:1804.05105*, 2018.

5 John Canny. Some algebraic and geometric computations in PSPACE. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 460–467. ACM, 1988.

6 Paul Colley, Anna Lubiw, and Jeremy Spinrad. Visibility graphs of towers. *Computational Geometry*, 7(3):161–172, 1997.

7 Hazel Everett. Visibility graph recognition - PhD thesis. 1990.

8 Hazel Everett and Derek G. Corneil. Recognizing visibility graphs of spiral polygons. *Journal of Algorithms*, 11(1):1–26, 1990.

9 Subir K. Ghosh. On recognizing and characterizing visibility graphs of simple polygons. In *Scandinavian Workshop on Algorithm Theory*, volume LNCS 318, pages 96–104. Springer, 1988.

10 Subir K. Ghosh. On recognizing and characterizing visibility graphs of simple polygons. *Discrete & Computational Geometry*, 17(2):143–162, 1997.

11 Subir K. Ghosh. *Visibility algorithms in the plane*. Cambridge University Press, 2007.

12 Jacob E Goodman and Richard Pollack. Allowable sequences and order types in discrete and computational geometry. In *New trends in discrete and computational geometry*, pages 103–134. Springer, 1993.

13 Udo Hoffmann and Keno Merckx. A universality theorem for allowable sequences with applications. *arXiv preprint arXiv:1801.05992*, 2018.

14 Jan Kratochvíl and Jirí Matousek. Intersection graphs of segments. *Journal of Combinatorial Theory, Series B*, 62(2):289–315, 1994.

15 Jürgen Richter-Gebert. Mnëv's universality theorem revisited. *Séminaire Lotaringien de Combinatorie*, 1995.

16 Marcus Schaefer. Complexity of some geometric and topological problems. In *International Symposium on Graph Drawing*, pages 334–344. Springer, 2009.

17 Peter Shor. Stretchability of pseudolines is NP-hard. *Applied Geometry and Discrete Mathematics-The Victor Klee Festschrift*, 1991.

18 Ileana Streinu. Non-stretchable pseudo-visibility graphs. *Computational Geometry*, 31(3):195–206, 2005.

19 Seth Teller and Pat Hanrahan. Global visibility algorithms for illumination computations. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 239–246. ACM, 1993.

Approximating the Sweepwidth of Polygons with Holes

Dorian Rudolph[1]

1 **Paderborn University**
dorian@mail.upb.de

Abstract

Consider a contaminated polygon P with the goal of decontaminating P by sweeping it with barrier curves, where the contaminant spreads instantly along all paths not blocked by a barrier. The maximum length of curves in a sweep needed to decontaminate P is defined as its *sweepwidth*. This problem was introduced Karaivanov et. al (2014) [8] who also proved that computing the sweepwidth of even simple, orthogonal polygons is $\mathcal{NP}$-hard. Therefore, we propose a polynomial time $O(\log n)$-approximation algorithm for the sweepwidth of n-vertex polygons with holes. We accomplish this by rasterizing the polygon into a grid which allows a reduction to the well known *node search problem* on graphs. In order to obtain a polynomially sized rasterization, we first apply a *compression* technique to the polygon.

1 Introduction

Karaivanov et al. [8] introduced the problem of decontaminating an initially contaminated planar region by sweeping it with moving barriers in the form of curves while the contaminant instantly spreads along any path that is not blocked by a barrier. It can be seen as an extension of the *node search problem* introduced by Kirousis and Papadimitriou [9] where a graph has to be decontaminated with as few searchers (or pebbles) as possible. Next, we will formally define these problems.

Sweepwidth [8]. Let P be a closed n-vertex polygon with holes and no intersecting edges. $P \subset \mathbb{R}^2$ shall also denote the set of points on the polygon including its boundary. Every point of P is either *contaminated* or *decontaminated*. We sweep P using a set of moving *barriers* $b\colon [0,1] \to 2^P$, where the barriers at any time $t \in [0,1]$ consist of the points $b(t)$. All points in $b(t)$ become decontaminated. Initially, all points of P (except $b(0)$) are contaminated. A decontaminated point q becomes *recontaminated* at time t if there exists a path from a contaminated point p to q not intersecting $b(t)$. We say b decontaminates P if all points in P are decontaminated at time 1 (see [8] for a more formal definition). We restrict barriers to piecewise continuously differentiable *barrier curves* with a finite number of pieces and barriers. This allows us to describe a sweep by a function $b\colon [0,1]^2 \to P$ such that $b(s,t)$ is piecewise continuous in both curve parameter s and time t, and for any t, $b(\cdot,t)$, is piecewise continuously differentiable. The function $s \mapsto b(s,t)$ describes the barriers at time t. We measure the total length of barriers at time t as the sum of the arc lengths of all pieces of $b(\cdot,t)$. The *bottleneck length* of b is defined as the supremum over time of the sum of the lengths of barriers in $b(\cdot,t)$. An exemplary sweep is depicted in Fig. 1. We refer to the minimum bottleneck length of all decontamination sweeps of P as *sweepwidth* of P, denoted $sw(P)$. Karaivanov et al. show that each decontamination sweep can be transformed into a *canonical* sweep without increasing its bottleneck length, i. e., a sweep where all barriers consist of one or two straight line segments connecting two points on the boundary of P.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

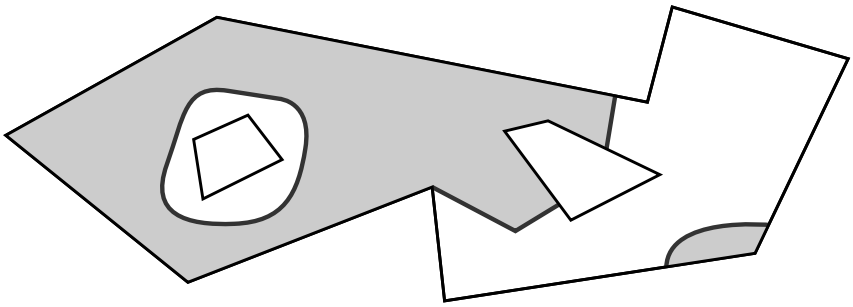


Figure 1 Incomplete sweep of a polygon with holes. The contaminated area is depicted as light gray, barriers as dark lines.

Node search [9]. Let $G = (V, E)$ be an undirected graph. All edges $e \in E$ are initially considered contaminated. To decontaminate G, we can *place* and *remove searchers* on nodes. We refer to this sequence of moves as *node-search strategy*. Nodes with a searcher are considered *guarded*. An edge $e = (v, w) \in E$ is *decontaminated* if v and w are guarded. e is recontaminated if there exists a path W of unguarded nodes from u or v to a node incident to a contaminated edge. Let the *node-search number* of G, $ns(G)$, be the minimum number of searchers needed to decontaminate all edges of G.

Related Work. For the polygon decontamination problem defined above, Karaivanov et al. [8] construct optimal sweeps for rather simple classes of polygons and prove that computing sweepwidth is $\mathcal{NP}$-hard for simple, orthogonal polygons. To the best of our knowledge, no approximation algorithms for the sweepwidth of general polygons are known. That problem is generalized by Markov et al. [12] to the *directed sweepwidth* where barriers must start and end on predefined parts of the boundary. They also give a rather involved lower bound for the sweepwidth based on the sweepwidth of three non-intersecting subshapes. Various other search problems for polygons have been analyzed in literature, including observing the entire polygon (*art gallery problem*) [13], or agents having to catch [10] or spot [1] an intruder. Another related problem is sweeping terrains with aerial vehicles [3]. If restricted to a single curve and simple polygons, sweepwidth is equivalent to the *elastic ring-width* [14], solved in time $O(n^2 \log n)$ [6]. Hence, ring-width constitutes an upper bound on sweepwidth. However, that bound can be arbitrarily bad, e. g., for an arbitrarily narrow T-shaped polygon.

Regarding different search problems on graphs, we refer the reader to the survey [4].

Our Contribution. We develop a polynomial time $O(\log n)$-approximation algorithm for the sweepwidth of n-vertex polygons with holes. We do this by rasterizing the polygon as a grid graph and computing its node-search number. It will follow that $O(1)$-approximation of sweepwidth is at most as hard as $O(1)$-approximation of the node-search number.

2 Algorithm

We will construct a sweep of P with a bottleneck length of $O(\log n \cdot sw(P))$ by rasterizing P using hexagonal cells (each cell is a closed set). Their adjacency graph G_P is an induced sub-

graph of the infinite triangular lattice graph [1]. First, we will argue $sw(P) = \Theta(ns(G_P))$ (using an appropriate scale). Afterwards, we describe a *compression* technique to construct a polygon P' with $sw(P) = \Theta(sw(P'))$ and polynomially sized $G_{P'}$. We can compute an $O(\log n)$-approximation of $ns(G_{P'})$ in polynomial time, also yielding an $O(\log n)$-approximation of $sw(P)$.

2.1 Rasterization

Cell size. Let R_P be the diameter of the largest inscribed circle in P. We define the *size* of cells, i. e., the length of edges in the lattice, as $r_P := R_P/n$. As $sw(P) \geq R_P$ [8], we can guard $O(n)$ cells throughout the entire sweep using curves of length $O(sw(P))$. To compute R_P, we construct the Voronoi diagram of the edges of P in time $O(n \log n)$ with Fortune's algorithm [5]. One of the Voronoi nodes must be the center of the largest inscribed circle.

Cell categories. We can now describe the rasterization process. There are different types of cells that will need to be treated differently by the algorithm. To that end, we introduce categories for cells intersecting P. Each cell not completely outside P belongs to exactly one of the following categories:

(a) *Blocked cell*: cell that either contains a vertex of P, or is completely inside P and is adjacent to two cells on opposing sides that are both intersected by at least one edge (dark gray in Fig. 2).
(b) *Full cell*: cell fully inside P that is not a blocked cell (white).
(c) *Empty cell*: any other cell (light gray).

All cells between the outermost blocked cells belonging to the same edge pair that do not contain a vertex shall be *redefined* as *empty cells* (see striped cells in Fig. 3).

If m cells intersect P, then we can easily compute the categories in time polynomial in m. Let G_P be the graph induced by full cells. We will argue that there are $O(n)$ blocked cells and that placing barriers around them separates cells belonging to different connected components of G_P.

▶ **Lemma 1.** *There are $O(n)$ blocked cells.*

▶ **Theorem 2.** *It holds that $sw(P) = \Theta(ns(G_P) \cdot r_P)$.*

Proof sketch. In order to prove $sw(P) = O(ns(G_P) \cdot r_P)$, we construct a sweep by first placing barriers at a distance of two around all blocked cells and decontaminate the inside of these barriers. One can show that this separates cells belonging to different connected components of G_P. Hence, we can decontaminate the resulting regions of P separately. Regions without full cells can easily be decontaminated with a curve of length $O(r_P)$. For each component of G_P, consider an optimal node-search strategy. Whenever a searcher is on a node of G_P, we place barriers at a distance of 2 around the corresponding cell. This can be shown to decontaminate regions of full cells including adjacent empty cells.

G_P has a connected component C such that $ns(G_P) = ns(C) = \Omega(n)$, as separate connected components can be decontaminated one after another. $ns(C) = \Omega(n)$ follows from the fact that it takes $\Omega(n)$ searchers to decontaminate an $\Omega(n) \times \Omega(n)$ parallelogram of cells, which is contained in the largest inscribed circle. Let $P_C \subset P$ be the polygon of cells in C. It is straightforward to prove $sw(P_C) = \Theta(ns(P_C) \cdot r_P)$. $sw(P) = \Omega(ns(G_P) \cdot r_P)$ follows. ◀

[1] We use hexagonal cells since then G_P is planar, which would not be the case for square cells due to diagonal adjacencies.

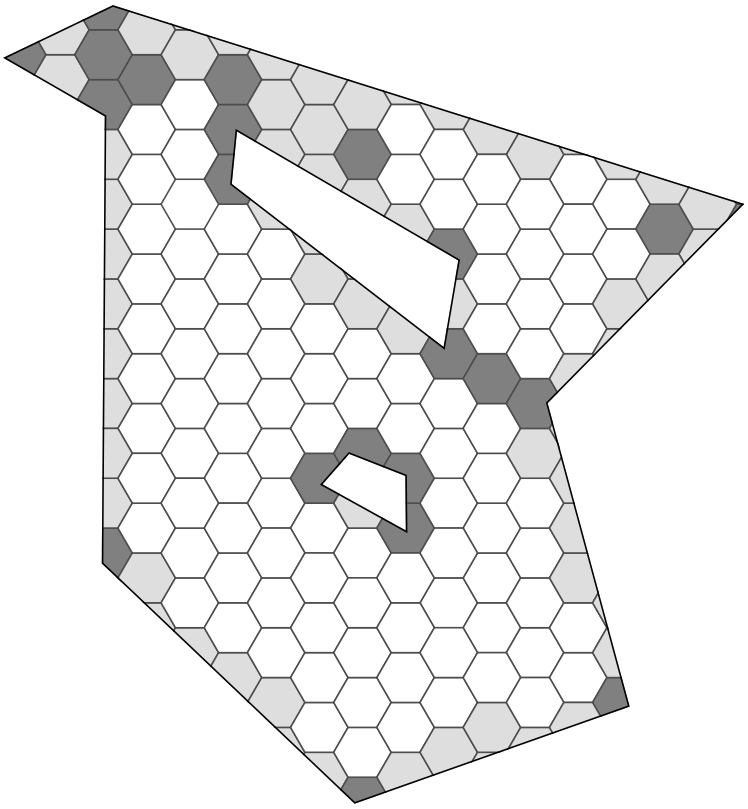


Figure 2 Polygon rasterized into a hexagonal grid. Blocked cells are depicted dark gray, empty cells light gray, and full cells white.

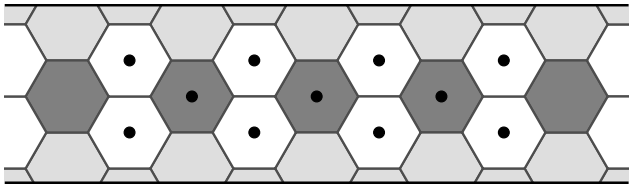


Figure 3 Cells between outermost blocked cells of an edge pair are redefined as empty (marked with dots).

Since G_P is planar, we can compute an $O(1)$-approximation of its *treewidth* $tw(G_P)$ in time $O(m \log^4 m)$ [7] if G_P has m nodes. As $ns(G_P) = \Omega(tw(G_P))$ and $ns(G_P) = O(tw(G_P) \log m)$ [2], we have an $O(\log m)$-approximation for $ns(G_P)$.

2.2 Polygon Compression

G_P may still contain arbitrarily many cells if P contains long, narrow sections. Thus, we will *compress* intervals along the x- and y-axes such that the resulting polygon P' can be rasterized using a polynomial number of cells. In the following, we will describe how to compress an interval along the x-axis where P has no vertices. We only consider maximal such intervals with a length greater than $(n+6)R_P$ and compress them down to that length, thereby bounding the distance between vertices. Compressed intervals will not overlap.

Let $I' := [x'_0, x'_1] \subset \mathbb{R}$, $x'_1 - x'_0 > (n+6)R_P$ be maximal such that no vertices of P have

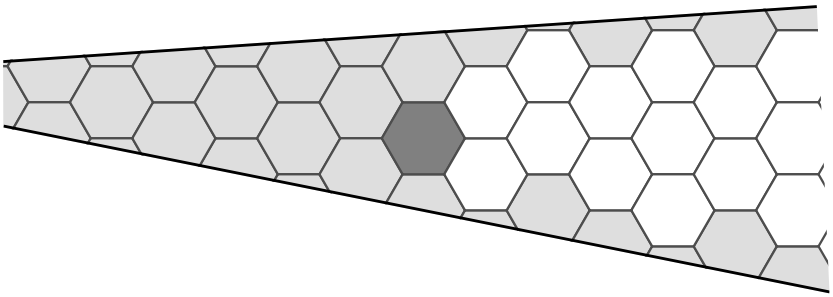


Figure 4 Blocked cells between two polygon edges.

their x-coordinate inside I'. That interval will look like the top of Fig. 5, i. e., there are k pairs of subsegments of edges, bordering part of the polygon. For each pair, we compute their minimum distances $d_1, \ldots, d_k$ which are bounded by R_P and the distance of the two points of the intersection of the edge with the line $x = x_0$ or $x = x_1$. Then, we cut out $I := [x_0, x_1] := [x'_0 + 2R_P, x'_1 - 2R_P]$, and replace that part as illustrated at the bottom of Fig. 5. More specifically, each pair of edges is replaced by a rectilinear path of width d_i from left to right, beginning with the lowest edge. The opening between the edges is constricted to d_i on both sides. For each pair of edges i, let y_i (y'_i) be the y-coordinate of the intersection of the lower edge with the left (right) boundary of I. W.l.o.g., we may assume $y_i \leq y'_i$. We then replace the lower edge by a path constructed as follows. Begin in (x_0, y_i) and move right until either reaching x_1 (see d_1 in Fig. 5) or an edge of pair $i-1$ (see d_3). We can clearly move right until at least $x_1 - \sum_{j=1}^{i-1} d_i \geq x_1 - n \cdot R_P$. Then we move straight up to y'_i and continue to (x_1, y'_i). Since $x_1 - x_0 \geq (n+2) \cdot R_P$, we move a distance of at least R_P right before moving up. The upper edge is replaced analogously such that that the distance between parallel segments is d_i. There will be an interval I'' with a size of at least $x_1 - x_0 - (n+1) \cdot R_p \geq R_P$ inside I where all edges are parallel to the x-axis. We compress I'' to a length of R_P. We obtain a polygon P' in polynomial time by performing the above steps on P both in x- and y-direction.

▶ **Lemma 3.** *P' has $O(n^4)$ vertices and intersects a polynomial number of cells of size $r_{P'} = O(R_P/n^4)$.*

Consequently, the size of $G_{P'}$ is polynomial in n. What remains to show is that the compression steps did not change sweepwidth by much.

▶ **Lemma 4.** *Let $\tilde{P}$ be the result of compressing P along one axis. Then $sw(\tilde{P}) = \Theta(sw(P))$.*

Proof sketch. Given a canonical decontamination sweep of P, we construct a decontamination sweep b of P which does not maintain barriers inside each interval $\hat{I} := [x_0 - R_P, x_1 + R_P]$, where x_0, x_1 are defined as above. Instead, b only uses barriers inside $\hat{I}$ for sweeping the region between edge pairs and subsequently places barriers just outside $\hat{I}$ to block off that region if necessary. We can show that this construction increases bottleneck length by at most a constant factor. Next, we construct a sweep $\tilde{b}$ of $\tilde{P}$ from b, where b and $\tilde{b}$ have the same barriers outside compressed regions and sweep corresponding compressed regions at the same time. This allows us to show $sw(\tilde{P}) = O(sw(P))$. $sw(P) = O(sw(\tilde{P}))$ follows analogously. ◀

The above lemmas directly imply the final theorem.

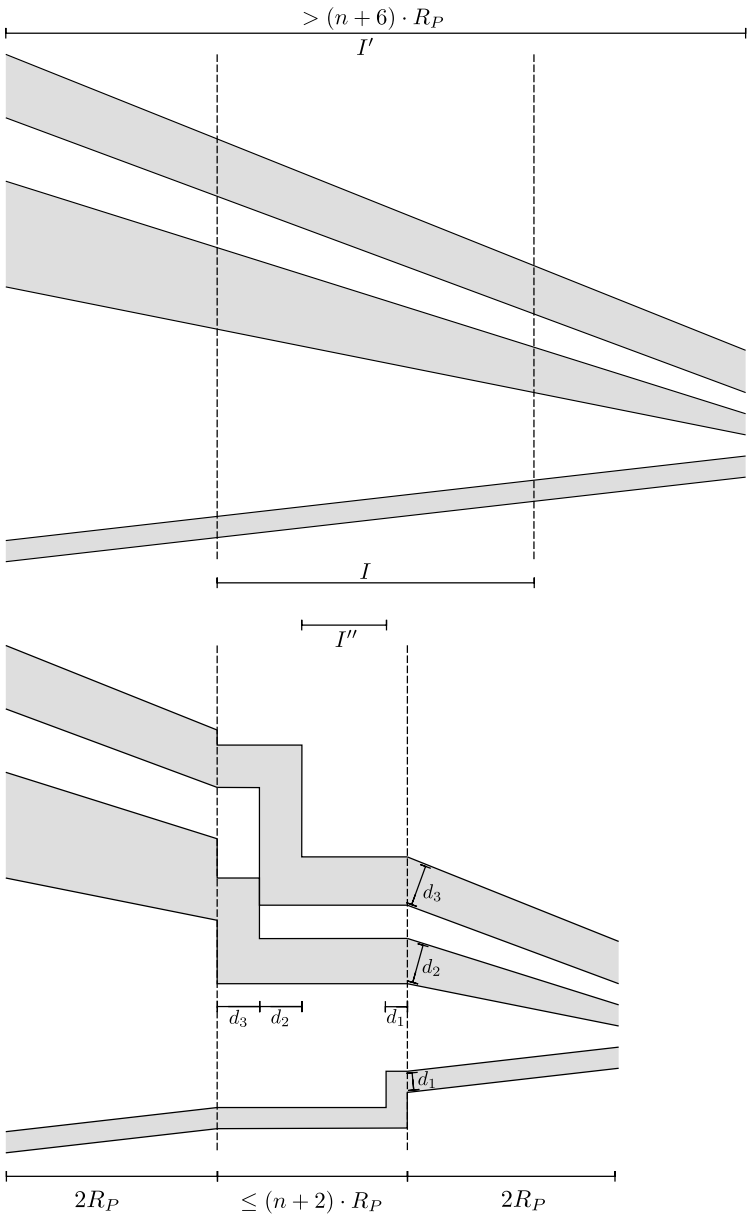


Figure 5 Illustration of polygon compression. d_1, d_2, d_3 are the minimum distance between their respective lines. The area inside P is depicted gray.

▶ **Theorem 5.** *There exists a polynomial time $O(\log n)$-approximation algorithm for computing $sw(P)$.*

The exact computation of $ns(G_P)$ is in $\mathcal{NP}$ [11], which implies the following corollary.

▶ **Corollary 6.** *$O(1)$-approximation of the sweepwidth of is in $\mathcal{NP}$.*

▶ Remark. If P is simple, it can be shown that $sw(P) = O(R_P \log n)$, which immediately gives an $O(\log n)$-approximation.

3 Conclusion

We constructed a polynomial time approximation algorithm for the relatively novel problem of computing a polygon's sweepwidth. Our proofs imply explicit constructions of sweeps. However, the runtime can be considered prohibitively bad. Improving approximation factors and runtime, potentially also for simple polygons, might therefore be interesting future work.

Acknowledgments. I thank Christian Scheideler and Kristian Hinnenthal for their help and advice.

References

1 Binay Bhattacharya, Tsunehiko Kameda, and John Z. Zhang. Surveillance of a polygonal area by a mobile searcher from the boundary: Searchability testing. In *2009 IEEE International Conference on Robotics and Automation*, pages 2461–2466, May 2009.

2 Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1):1 – 45, 1998.

3 Alon Efrat, Mikko Nikkilä, and Valentin Polishchuk. Sweeping a terrain by collaborative aerial vehicles. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL'13, pages 4–13, New York, NY, USA, 2013. ACM.

4 Fedor V. Fomin and Dimitrios M. Thilikos. An annotated bibliography on guaranteed graph searching. *Theoretical Computer Science*, 399(3):236 – 245, 2008. Graph Searching.

5 Steven Fortune. A sweepline algorithm for voronoi diagrams. *Algorithmica*, 2(1):153, Nov 1987.

6 Jacob E. Goodman, János Pach, and Chee K. Yap. Mountain climbing, ladder moving, and the ring-width of a polygon. *The American Mathematical Monthly*, 96(6):494–510, 1989.

7 Qian-Ping Gu and Gengchun Xu. Near-linear time constant-factor approximation algorithm for branch-decomposition of planar graphs. In Dieter Kratsch and Ioan Todinca, editors, *Graph-Theoretic Concepts in Computer Science*, pages 238–249, Cham, 2014. Springer International Publishing.

8 Borislav Karaivanov, Minko Markov, Jack Snoeyink, and Tzvetalin S. Vassilev. Decontaminating planar regions by sweeping with barrier curves. In *26th Canadian Conference on Computational Geometry, CCCG 2014*, pages 206–211, 2014.

9 Lefteris M. Kirousis and Christos H. Papadimitriou. Searching and pebbling. *Theoretical Computer Science*, 47:205–218, 1986.

10 Kyle Klein and Subhash Suri. Catch me if you can: Pursuit and capture in polygonal environments with obstacles. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, AAAI'12, pages 2010–2016. AAAI Press, 2012.

11 Andrea S. LaPaugh. Recontamination does not help to search a graph. *J. ACM*, 40:224–245, 1993.

12 Minko Markov, Vladislav Haralampiev, and Georgi Georgiev. Lower bounds on the directed sweepwidth of planar shapes. 2015.

13 Jorge Urrutia. Chapter 22 - art gallery and illumination problems. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 973 – 1027. North-Holland, Amsterdam, 2000.

14 Chee-Keng Yap. How to move a chair through a door. *IEEE Journal on Robotics and Automation*, 3(3):172–181, June 1987.

Probabilistic smallest enclosing ball in high dimensions via subgradient sampling*

Amer Krivošija[1] and Alexander Munteanu[2]

1 **Department of Computer Science, TU Dortmund, Germany**
amer.krivosija@tu-dortmund.de
2 **Department of Computer Science, TU Dortmund, Germany**
alexander.munteanu@tu-dortmund.de

Abstract

We study a variant of the median problem for a collection of point sets in high dimensions. This generalizes the geometric median as well as the (probabilistic) smallest enclosing ball (pSEB) problems. Our main objective and motivation is to improve the previously best algorithm for the pSEB problem by reducing its exponential dependence on the dimension to linear. This is achieved via a novel combination of sampling techniques for clustering problems in metric spaces with the framework of stochastic subgradient descent. As a result, the algorithm becomes applicable to shape fitting problems in Hilbert spaces of unbounded dimension via kernel functions. We present an exemplary application by extending the support vector data description (SVDD) shape fitting method to the probabilistic case. This is done by simulating the pSEB algorithm implicitly in the feature space induced by the kernel function.

1 Introduction

The (probabilistic) smallest enclosing ball (pSEB) problem in $\mathbb{R}^d$ is to find a center that minimizes the (expected) maximum distance to the input points (see Definition 3.1). It occurs often as a building block for complex data analysis and machine learning tasks like estimating the support of high dimensional distributions, outlier detection, novelty detection, classification and robot gathering [5, 15, 16, 18]. It is thus very important to develop efficient algorithms for the base problem. This involves reducing the number of points but also keeping the dependence on the dimension as low as possible. We study both objectives and focus on a small dependence on the dimension. This is motivated as follows. Kernel methods are a common technique in machine learning. These methods implicitly project the d-dimensional input data into much larger dimension D where simple linear classifiers or spherical data fitting methods can be applied to obtain a non-linear separation or non-convex shapes in the original d-dimensional space. The efficiency of kernel methods is usually not harmed since inner products and thus distances in the D-dimensional space can be evaluated in $O(d)$ time.

In some cases, however, a proper approximation relying on sampling and discretizing the ambient solution space may require a polynomial or even exponential dependence on D. The algorithm of Munteanu et al. [11] is the only fully polynomial time approximation scheme (FPTAS) and the fastest algorithm to date for the pSEB problem in fixed dimension. However, it suffers from the stated problems. In particular, the number of realizations sampled by their algorithm had a linear dependence on D stemming from a ball-cover decomposition of the solution space. The actual algorithm made a brute force evaluation (on the sample)

* The full version of this paper will appear at SoCG 2019. This work was supported by the German Science Foundation (DFG) Collaborative Research Center SFB 876 "Providing Information by Resource-Constrained Analysis", projects A2 and C4.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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 all centers in a grid of exponential size in D. This is prohibitive in the setting of kernel methods since the implicit feature space may have infinite dimension. Even if it is possible to exploit the up to n-dimensional subspace spanned by n points in infinite dimensions, we would still have $D = n \gg d$ leading to exponential time algorithms. To make the pSEB algorithm viable in the context of kernel methods and generally in high dimensions, it is desirable to reduce the dependence on the dimension to a small polynomial occurring only in evaluations of inner products and distances of low dimensional vectors.

Related work: The study of *probabilistic clustering problems* was initiated by Cormode and McGregor [7]. They developed approximation algorithms for the probabilistic settings of k-means, k-median as well as k-center clustering. Munteanu et al. [11] gave the first fully polynomial time $(1+\varepsilon)$-approximation scheme (FPTAS) for the pSEB problem, in fixed dimensions, in time $O(nd/\varepsilon^{O(1)} + 1/\varepsilon^{O(d)})$. We reduce its exponential dependence on d to linear, using sampling techniques. The stochastic subgradient descent from convex optimization [6, 12] is a quite popular and often only implicitly used technique in the core-set literature [2, 4, 10]. Indyk and Thorup [8, 17] showed that a uniform sample of size $O(\log n/\varepsilon^2)$ is sufficient to approximate the discrete metric 1-median within a factor of $(1+\varepsilon)$. We adapt these ideas to find a $(1+\varepsilon)$-approximation to the best center in our setting.

Kernel functions simulate an inner product space in large or even unbounded dimensions but can be evaluated via simple low dimensional vector operations in the original dimension of input points [14]. This enables simple spherical shape fitting via a smallest enclosing ball algorithm in the high dimensional feature space, which implicitly defines a more complex and even non-convex shape in the original space. The smallest enclosing ball problem in kernel spaces is well-known as the support vector data description (SVDD) problem [16, 18].

1.1 Contributions and outline

We extend the geometric median in Euclidean space to the more general problem of finding a center $c \in \mathbb{R}^d$ that minimizes the sum of maximum distances to sets of points in a given collection of N point sets. We show how to solve this problem via estimation and sampling techniques combined with a stochastic subgradient descent algorithm, see Theorem 2.2.

The elements in the collection are sets of n points in $\mathbb{R}^d$. In [11] they were summarized via strong coresets of size $1/\varepsilon^{\Theta(d)}$. This is not an option in high dimensions. Reviewing the techniques of [1] we show that no reduction below $\min\{n, \exp(d^{1/3})\}$ is possible unless sacrificing an additional approximation factor of roughly $\sqrt{2}$, see Theorem 2.3. However, it is possible to achieve roughly a $(\sqrt{2}+\varepsilon)$-approximation in streaming via the *blurred-ball-cover* [1] of size $O(1/\varepsilon^3 \cdot \log 1/\varepsilon)$, and in an off-line setting via weak coresets [2, 3] of size $O(1/\varepsilon)$.

We show in Theorem 3.2 how Theorem 2.2 improves the previously best FPTAS for the pSEB problem from $O(dn/\varepsilon^3 \cdot \log 1/\varepsilon + 1/\varepsilon^{O(d)})$ to $O(dn/\varepsilon^4 \cdot \log^2 1/\varepsilon)$. In particular the dependence on the dimension d is reduced from exponential to linear and more precisely occurs only in distance evaluations between points in d-dimensional Euclidean space, but not in the number of sampled points nor in the number of candidate centers to evaluate.

This enables working in very high D-dimensional Hilbert spaces whose inner products and distances are given implicitly via positive semidefinite kernel functions. These functions can be evaluated in $O(d)$ time although D is large or even unbounded. We extend the well-known SVDD method to the probabilistic case, see Theorem 3.3.

1.2 General notation

We denote the set of positive integers up to $n \in \mathbb{N}$ by $[n] = \{1, \ldots, n\}$. For any convex function $f\colon \mathbb{R}^d \to \mathbb{R}$ we denote by $\partial f(x) = \left\{g \in \mathbb{R}^d \mid \forall y \in \mathbb{R}^d\colon f(x) - f(y) \leq \langle g, x - y\rangle\right\}$ the set of subgradients of f at x. We assume the error parameter satisfies $0 < \varepsilon < 1/9$.

2 A generalized median problem

The pSEB problem can be reduced to two different types of 1-median problems [11]. One of them is defined on the set of all non-empty locations in $\mathbb{R}^d$ where probabilistic points may appear, equipped with the Euclidean distance. The other is defined on the collection of all possible realizations of probabilistic point sets, and the distance measure between a center $c \in \mathbb{R}^d$ and a realization $P_i \subset \mathbb{R}^d$ is $m(c, P_i) = \max_{p \in P_i} \|c - p\|$. We state a generalized median problem that we call the *set median problem* and covers both of these cases.

▶ **Definition 2.1** (set median problem)**.** Let $\mathcal{P} = \{P_1, \ldots, P_N\}$ be a family of finite non-empty sets where $\forall i \in [N]\colon P_i \subset \mathbb{R}^d$ and $n = \max\{|P_i| \mid i \in [N]\}$. The set median problem on $\mathcal{P}$ consists in finding a center $c \in \mathbb{R}^d$ that minimizes the cost function

$$f(c) = \sum_{i=1}^{N} m(c, P_i).$$

Note that in case of singleton sets, the set median problem is equivalent to the well-known Fermat-Weber problem (a.k.a. 1-median or geometric median). Also, for $N = 1$ it coincides with the smallest enclosing ball or 1-center problem. For both of these problems there are known algorithms based on the subgradient method from convex optimization [2, 6].

The Lipschitz constant of the function f can be bounded by N. We want to minimize f via the subgradient method, see [12]. For that sake we need to compute a subgradient $g(c_i) \in \partial f(c_i)$ at the current center c_i. The subgradient computation takes $O(dnN)$ time to calculate, since in each of the N terms of the sum we maximize over $|P_i| \leq n$ distances in d dimensions to find a point in P_i that is furthest away from c. To remove the dependence on N we replace the exact subgradient $g(c_i)$ by a uniform sample of only one nonzero term which points into the right direction in expectation. Then we can adapt the deterministic subgradient method from [12] using the random unbiased subgradient in such a way that the result is in expectation a $(1 + \varepsilon)$-approximation to the optimal solution. Given an initial center c_0, a fixed step size s, and a number of iterations ℓ, our algorithm iteratively picks a set $P_j \in \mathcal{P}$ uniformly at random and chooses a point $p_j \in P_j$ that attains the maximum distance to the current center. This point is used to compute an approximate subgradient. The algorithm finally outputs the best center found in all iterations.

To bound in expectation the quality of the output of our algorithm to be a $(1 + \varepsilon)$-approximation, we choose the values of parameters c_0, s, and ℓ in an appropriate way. It suffices to run our algorithm for $\ell \in O(1/\varepsilon^2)$ iterations, and to choose c_0 to be an arbitrary point in a randomly chosen input set from $\mathcal{P}$. We estimate the average cost on a sample of size $1/\varepsilon$ [9], which bounds the value of s. It remains to describe how to find the best center out of all iterations of our algorithm efficiently. We cannot do this exactly since evaluating the cost even for one single center takes time $O(dnN)$. However, we use a sampling technique [8, 17] (cf. the related work above), adapted here to work in our setting, to find a point that is a $(1 + \varepsilon)$-approximation of the best center in a finite set of candidate centers. The main difference is that in the original work the set of input points and the set of candidate solutions are identical. In our setting, however, we have that the collection of input sets and

the set of candidate solutions may be completely distinct. Putting all pieces together we have the following Theorem.

▶ **Theorem 2.2.** *Consider an input $\mathcal{P} = \{P_1, \ldots, P_N\}$, where for every $i \in [N]$ we have $P_i \subset \mathbb{R}^d$ and $n = \max\{|P_i| \mid i \in [N]\}$. There exists an algorithm that computes a center $\tilde{c}$ that is with constant probability a $(1+\varepsilon)$-approximation to the optimal solution c^* of the set median problem (see Definition 2.1). Its running time is $O(dn/\varepsilon^4 \cdot \log^2 1/\varepsilon)$.*

The removal of the linear dependence on n for the maximum distance computations was achieved in [11] via a grid based strong coreset of size $1/\varepsilon^{\Theta(d)}$. However, here we focus on reducing the dependence on d, and exponential is not an option if we want to work in high dimensions. It turns out that without introducing an exponential dependence on d, we would have to lose a constant approximation factor. We adapted the techniques of [1] to show that no small data structure can exist for answering maximum distance queries to within a factor of less than roughly $\sqrt{2}$. In comparison to the previous results, it is not limited to the streaming setting, as in [1], and it is not restricted to subsets of the input, as in [13].

▶ **Theorem 2.3.** *Any data structure that, with probability at least $2/3$, α-approximates maximum distance queries on a set $S \subset \mathbb{R}^d$ of size $|S| = n$, for $\alpha < \sqrt{2}\left(1 - 2/d^{1/3}\right)$, requires $\Omega\left(\min\{n, \exp\left(d^{1/3}\right)\}\right)$ bits of storage.*

3 Applications

3.1 Probabilistic smallest enclosing ball

We apply our result to the pSEB problem, as given in [11]. In such a setting, the input is a set $\mathcal{D} = \{D_1, \ldots, D_n\}$ of n discrete and independent probability distributions. The i-th distribution D_i is defined over a set of z possible locations $q_{i,j} \in \mathbb{R}^d \cup \{\perp\}$, for $j \in [z]$, where $\perp$ indicates that the i-th point is not present in a sampled set, i.e., $q_{i,j} = \perp \Leftrightarrow \{q_{i,j}\} = \emptyset$. We call these points *probabilistic points*. Each location $q_{i,j}$ is associated with the probability $p_{i,j}$, such that $\sum_{j=1}^{z} p_{i,j} = 1$, for every $i \in [n]$. Thus the probabilistic points can be considered as independent random variables X_i. A probabilistic set X of probabilistic points is also a random variable.

▶ **Definition 3.1.** ([11]) Let $\mathcal{D}$ be a set of n discrete distributions, where each distribution is defined over z locations in $\mathbb{R}^d \cup \{\perp\}$. The pSEB problem is to find a center $c^* \in \mathbb{R}^d$ that minimizes the expected smallest enclosing ball cost: $c^* \in \operatorname{argmin}_{c \in \mathbb{R}^d} \mathbb{E}\left[m(c, X)\right]$, where the expectation is taken over the randomness of $X \sim \mathcal{D}$.

Our pSEB algorithm adapts the framework of [11], but plugging in Theorem 2.2 it differs mainly in three points. First, the number of samples had a dependence on d hidden in the O-notation. This is not the case any more. Second, the sampled realizations are not sketched via coresets of size $1/\varepsilon^{\Theta(d)}$ any more. Third, the running time of the actual optimization task is reduced instead of an exhaustive grid search.

▶ **Theorem 3.2.** *Let $\mathcal{D}$ be a set of n discrete distributions, where each distribution is defined over z locations in $\mathbb{R}^d \cup \{\perp\}$. Let $\tilde{c} \in \mathbb{R}^d$ be the output of our pSEB algorithm on input $\mathcal{D}$. Let $\varepsilon < 1/9$. With constant probability $\tilde{c}$ is a $(1+\varepsilon)$-approximation for the pSEB problem,*

$$\mathbb{E}_X\left[m(\tilde{c}, X)\right] \leq (1+\varepsilon) \min_{c \in \mathbb{R}^d} \mathbb{E}_X\left[m(c, X)\right].$$

The running time of our pSEB algorithm is $O(dn \cdot (z/\varepsilon^3 \cdot \log 1/\varepsilon + 1/\varepsilon^4 \cdot \log^2 1/\varepsilon))$.

Comparing to the result of [11], the running time is reduced from $O(dnz/\varepsilon^{O(1)} + 1/\varepsilon^{O(d)})$ to $O(dnz/\varepsilon^{O(1)})$. The factor of d plays a role only in computations of distances between two points in $\mathbb{R}^d$. Further the sample size and the number of centers that need to be evaluated do not depend on the dimension d any more. This is crucial in the following.

3.2 Probabilistic support vector data description (pSVDD)

We consider the SVDD problem, i.e., the SEB problem in kernel spaces, and show how to extend it to its probabilistic version. Let $K\colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a positive semidefinite kernel function with feature map $\varphi\colon \mathbb{R}^d \to \mathcal{H}$, where $\mathcal{H}$ is a high dimensional Hilbert space, say $\mathbb{R}^D$, where $D \gg d$ [14].

▶ **Theorem 3.3.** *Let $\mathcal{D}$ be a set of n discrete distributions, where each distribution is defined over z locations in $\mathbb{R}^d \cup \{\perp\}$. There exists an algorithm that implicitly computes $\tilde{c} \in \mathcal{H}$ that with constant probability is a $(1+\varepsilon)$-approximation for the probabilistic SVDD problem. It is*

$$\mathbb{E}_X\left[m(\tilde{c}, \varphi(X))\right] \leq (1+\varepsilon)\min_{c\in\mathcal{H}} \mathbb{E}_X\left[m(c, \varphi(X))\right],$$

where the expectation is taken over the randomness of $X \sim \mathcal{D}$, and $\varphi(X) = \{\varphi(x_i) \mid x_i \in X\}$. The running time of the algorithm is $O(dn \cdot \left(z/\varepsilon^3 \cdot \log 1/\varepsilon + 1/\varepsilon^8 \cdot \log^2 1/\varepsilon\right))$.

References

1 P. K. Agarwal and R. Sharathkumar. Streaming algorithms for extent problems in high dimensions. *Algorithmica*, 72(1):83–98, 2015.

2 M. Bădoiu and K. L. Clarkson. Smaller core-sets for balls. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 801–802, 2003.

3 M. Bădoiu and K. L. Clarkson. Optimal core-sets for balls. *Comput. Geom*, 40(1):14–22, 2008.

4 M. Bădoiu, S. Har-Peled, and P. Indyk. Approximate clustering via core-sets. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC)*, pages 250–257, 2002.

5 M. Cieliebak, P. Flocchini, G. Prencipe, and N. Santoro. Distributed computing by mobile robots: Gathering. *SIAM J. Comput.*, 41(4):829–879, 2012.

6 M. B. Cohen, Y. T. Lee, G. L. Miller, J. Pachocki, and A. Sidford. Geometric median in nearly linear time. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 9–21, 2016.

7 G. Cormode and A. McGregor. Approximation algorithms for clustering uncertain data. In *Proceedings of the 27th Symposium on Principles of Database Systems (PODS)*, pages 191–200, 2008.

8 P. Indyk. *High-dimensional Computational Geometry.* PhD thesis, Stanford University, 2000.

9 A. Kumar, Y. Sabharwal, and S. Sen. Linear-time approximation schemes for clustering problems in any dimensions. *J. ACM*, 57(2):5:1–5:32, 2010.

10 A. Munteanu and C. Schwiegelshohn. Coresets-methods and history: A theoreticians design pattern for approximation and streaming algorithms. *KI*, 32(1):37–53, 2018.

11 A. Munteanu, C. Sohler, and D. Feldman. Smallest enclosing ball for probabilistic data. In *30th Annual Symposium on Computational Geometry (SoCG)*, pages 214–223, 2014.

12 Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course.* Applied Optimization. Springer, New York, 2004.

13 R. Pagh, F. Silvestri, J. Sivertsen, and M. Skala. Approximate furthest neighbor with application to annulus query. *Inf. Syst.*, 64:152–162, 2017.

14 B. Schölkopf and A. J. Smola. *Learning with Kernels: support vector machines, regularization, optimization, and beyond.* Adaptive computation and machine learning series. MIT Press, 2002.

15 M. Stolpe, K. Bhaduri, K. Das, and K. Morik. Anomaly detection in vertically partitioned data by distributed core vector machines. In *Proceedings of Machine Learning and Knowledge Discovery in Databases - European Conference, (ECML/PKDD) Part III*, pages 321–336, 2013.

16 D. M. J. Tax and R. P. W. Duin. Support vector data description. *Machine Learning*, 54 (1):45–66, 2004.

17 M. Thorup. Quick k-median, k-center, and facility location for sparse graphs. *SIAM J. Comput.*, 34(2):405–432, 2005.

18 I. W. Tsang, J. T. Kwok, and P. Cheung. Core vector machines: Fast SVM training on very large data sets. *Journal of Machine Learning Research*, 6:363–392, 2005.

Practical volume estimation by a new annealing schedule for cooling convex bodies

Apostolos Chalkis[1], Ioannis Z. Emiris[2], and Vissarion Fisikopoulos[3]

1 **Department of Informatics & Telecommunications**
National & Kapodistrian University of Athens, Greece
achalkis@di.uoa.gr
2 **Department of Informatics & Telecommunications**
National & Kapodistrian University of Athens, Greece, and
ATHENA Research & Innovation Center, Greece
emiris@di.uoa.gr
3 **Department of Informatics & Telecommunications**
National & Kapodistrian University of Athens, Greece
vfisikop@di.uoa.gr

Abstract

We experimentally study the problem of estimating the volume of convex bodies, focusing on H- and V-polytopes, as well as zonotopes. Although a lot of effort is devoted to practical algorithms for H-polytopes there is no such method for the latter two representations. We propose a new, practical method for all representations which also improves upon the performance of existing methods on H-polytopes.

1 Introduction

Volume computation of a convex body in general dimension is a fundamental problem in discrete geometry. In the past 28 years randomized algorithms, for this problem, have made great progress. The two existing [5], [2] practical methods and the corresponding implementations are based on theoretical results, but they make some practical adjustments and show experimentally that they estimate volumes with small errors and high probability. Our new practical method can be used for general convex bodies but in this paper we focus on convex polytopes. A convex polytope P can be given as (a) an intersection of q halfspaces (H-polytope), (b) a convex hull of a set of points (V-polytope) and (c) a Minkowski sum of k segments (zonotope). We assume that an H-polytope is given by a matrix $A \in \mathbb{R}^{q\times d}$ and a vector $b \in \mathbb{R}^q$, s.t. $P = \{x \mid Ax \leq b\}$ and a zonotope by a matrix $G \in \mathbb{R}^{d\times k}$ which contains the k segments column-wise.

Exact volume computation is #P-hard for H- and V-polytopes, including zonotopes [6]. There are several implementations in packages such as `VINCI` or `qHull` but, as expected, they do not scale beyond, say, $d \geq 15$ dimensions. The first approximation algorithm, is given in [4] with complexity $O^*(d^{23})$.

The main approach relies on a Multiphase Monte Carlo (MMC) sequence of convex bodies $P_0 \subseteq \cdots \subseteq P_m = P$ such that rejection sampling would efficiently estimate $\mathrm{vol}(P_i)/\mathrm{vol}(P_{i-1})$, i.e. sample uniform points from P_i and reject/accept points in P_{i-1}. Assuming P is well-rounded and also that the unit ball B_d is the largest inscribed ball in P, defining P_1. Then each convex body P_i is defined by the intersection of a scaled copy of B_d with P while the largest ball, which defines $P_m = P$, is an enclosing ball of P. Then we estimate $\mathrm{vol}(P)$ through the telescopic product (2). The critical complexity issue is to minimize the length of the sequence in MMC, called m, while each ratio remains large enough to use rejection sampling. In [7] the sequence of balls in MMC is defined deterministically for each instance,

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

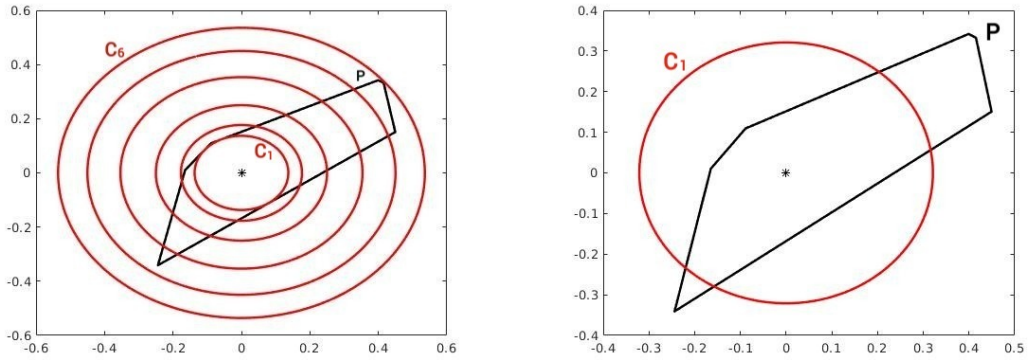


Figure 1 Balls in MMC in [7] left and from the annealing schedule right (r=0.25, δ=0.05)

while $m = O(d \lg d)$. Our method uses a new annealing schedule to define a sparser sequence of balls, (Fig. 1), and a new practical convergence criterion for each ratio in the MMC in order to minimize the number of sampled points to estimate each ratio. Moreover for zonotopes of order ≤ 4 we do not use balls in MMC but we define a centrally symmetric convex polytope which reduces the number of bodies (or phases) significantly.

Current state-of-the-art software is based on the above paradigms and, for H-polytopes, typically uses Hit-and-Run (HnR). `VolEsti` [5], which scales up to hundreds of dimensions, uses Coordinate-Direction HnR. We shall also juxtapose the software of [2] (for H-polytopes), which implements [1] with an annealing schedule.

These implementations can not handle efficiently zonotopes or V-polytopes as they require an inscribed ball (ideally the largest one). Additionally the software of [2] requires the number of facets which is typically exponential in the dimension for both zonotopes and V-polytopes. Our software outperforms for $d \leq 100$ software for H-polytopes by [2] and [5]. Moreover we provide the first practical method for V-polytopes and zonotopes that scales to high dimensions (currently 100 for V-polytopes and low-order zonotopes).

We introduce some notions from statistics and refer to [3] for details. Given a random sample of size ν from a random variable $X \sim \mathcal{N}(\mu, \sigma^2)$ with unknown variance σ^2, the (one tailed) t-test checks the null hypothesis that the population mean exceeds a specified value μ_0 using the statistic $t = \frac{\bar{x}-\mu_0}{s/\sqrt{\nu}} \sim t_{\nu-1}$, where $\bar{x}$ is the sample mean, s the sample standard deviation and $t_{\nu-1}$ is the t-student distribution with $\nu - 1$ degrees of freedom. Given a significance level $\alpha > 0$ we test the null hypothesis for the mean value of the population, $H_0 : \mu \leq \mu_0$. We reject H_0 if,

$$t \geq t_{\nu-1,\alpha} \Rightarrow \bar{x} \geq \mu_0 + t_{\nu-1,\alpha} s/\sqrt{n}, \tag{1}$$

where $t_{\nu-1,\alpha}$ is the critical value of the t-student distribution. Inequality 1 implies $\Pr(H_0 \text{ true} \mid \text{reject } H_0) = \alpha$. Otherwise we fail to reject H_0.

2 Volume algorithm

Our method introduces some new algorithmic features. The MMC of the algorithm constructs a sequence of convex bodies $C_1 \supseteq \cdots \supseteq C_m$ intersecting the given polytope P; we introduce a new annealing schedule in order to minimize m. A typical choice for the C_i's is a sequence of co-centric balls but any set of convex bodies can be used in our method. We re-write the

telescopic product in [7] as follows:

$$\mathrm{vol}(P) = \frac{\frac{\mathrm{vol}(P_m)}{\mathrm{vol}(C_m)}}{\frac{\mathrm{vol}(P_1)}{\mathrm{vol}(P_0)}\frac{\mathrm{vol}(P_2)}{\mathrm{vol}(P_1)}\cdots\frac{\mathrm{vol}(P_m)}{\mathrm{vol}(P_{m-1})}}\mathrm{vol}(C_m), \quad \text{where } P_0 = P, P_i = C_i \cap P. \tag{2}$$

The behavior of our method is parametrized by: the error of approximation ϵ, cooling parameters $0 < r < 1$, $\delta > 0$, s.t. $0 < r + \delta < 1$ which are used in the schedule, significance level $\alpha > 0$ of the statistical tests, ν the degrees of freedom for the t-student used in t-tests, parameter N that controls the number of points νN generated in P_i, and n the length of the sliding window. From the telescopic product (2) it is clear that in practical estimations C_m has to be a convex body whose volume is computed much faster than $\mathrm{vol}(P)$ (ideally by a closed formula) and which can be sampled efficiently.

The annealing schedule specifies $C_1 \supseteq \cdots \supseteq C_m$ using the following two statistical tests:

testL$(P_1, P_2, r, \delta, \alpha, \nu, N)$:	**testR**$(P_1, P_2, r, \alpha, \nu, N)$:
$H_0:\ \mathrm{vol}(P_2)/\mathrm{vol}(P_1) \geq r + \delta$	$H_0:\ \mathrm{vol}(P_2)/\mathrm{vol}(P_1) \leq r$
Successful if H_0 is **rejected**	**Successful** if H_0 is **rejected**

These tests are being used by annealing schedule to restrict each ratio $r_i = \mathrm{vol}(P_{i+1})/\mathrm{vol}(P_i)$ in the interval $[r, r + \delta]$ with high probability in order to avoid unnecessarily big ratios in MMC. Then we can use rejection sampling to estimate efficiently each ratio. Given P_i, `testL` is used to define $P_{i+1} \subseteq P_i$ s.t. ratio $\mathrm{vol}(P_{i+1})/\mathrm{vol}(P_i)$ is not too large, while `testR` is used so that the ratio is not too small, with high probability. In general, if we sample N uniform points from a body P_i then random variable X that counts points in P_{i+1}, follows $X \sim b(N, r_i)$, the binomial distribution, and random variable $Y = X/N \sim \mathcal{N}(r_i, r_i(1-r_i)/N)$ is Gaussian. If we sample νN points from P_i and split the sample into ν sublists of length N, the corresponding ν ratios are experimental values that follow $\mathcal{N}(r_i, r_i(1 - r_i)/N)$ and can be used to check both null hypotheses for r_i in `testL` and `testR`. Using the mean μ_0 of the ν ratios, r_i is restricted to $[r, r + \delta]$ with high probability when the following holds:

$$r + \delta - t_{\nu-1,\alpha}\frac{s}{\sqrt{\nu}} > \mu_0 > r + t_{\nu-1,\alpha}\frac{s}{\sqrt{\nu}}, \tag{3}$$

Initialization of the annealing schedule is to compute the body C' s.t. the volume of $C' \cap P$ could be efficiently estimated using rejection sampling, i.e. sampling from C' and accepting points in $C' \cap P$. Body C' is also used for the stopping criterion: the annealing schedule stops in the i-th step if `testR`$(P_i, C' \cap P)$ succeeds, which means that the $\mathrm{vol}(P_i)$ is close enough to $\mathrm{vol}(C' \cap P)$, so that rejection sampling can be used. Then set $m = i + 1$ and $C_m = C'$, $P_m = C_m \cap P$. When balls are used in the MMC, the smallest ball C_m is not an inscribed ball and the largest one, C_1, is not an enclosing ball as in [7]. Hence in practice the number of phases in [5] is an upper bound, with high probability, on the number of phases of our method, when $0 < r + \delta < 1/2$. Fig. 1 shows the sequence of balls for a given polytope P with our method (m=1) and in [7] (m=6). The ratios our method estimates are: $\mathrm{vol}(P_1)/\mathrm{vol}(P_0)$ and $\mathrm{vol}(P_1)/\mathrm{vol}(C_1)$, where $P_0 = P$, $P_1 = C_1 \cap P$.

The annealing schedule returns m bodies and we estimate $m + 1$ volume ratios. For fixed step i and each sample point generated in P_i, we keep the value of the i-th ratio. We store the last n such values in a queue called sliding window denoted by W whose length is n. We update W for each new point by inserting the new ratio and by popping out the oldest ratio in W. For each ratio r_i, we bound error by ϵ_i s.t. $\sum_{i=0}^{m} \epsilon_i^2 = \epsilon^2$ then, from standard error propagation analysis, (2) estimates $\mathrm{vol}(P)$ with error at most ϵ.

At step i, let $\hat{\mu}$ be the mean, s the st.d. of W and $Pr = 3/4$. Using $p = (1 + Pr)/2$, where $z_p = \sqrt{2} \cdot \text{erf}^{-1}(2p - 1)$, we consider the interval $[\hat{\mu} - z_p s, \hat{\mu} + z_p s]$, where erf is the Gauss error function. The values that the sliding window contains are not independent, so we can not define a confidence interval using t-student distribution, but we experimentally show that the following practical criterion is very efficient:

$$\text{if } \frac{2z_p s}{\hat{\mu} - z_p s} \leq \epsilon_i/2, \text{ then declare convergence.} \tag{4}$$

In practice we set $n = O(d^2)$ so that our method estimates volumes with error $\leq \epsilon$ with high probability.

We use Hit-and-Run (HnR) with uniform target distribution for sampling from P_i at step i of the annealing schedule. One step of HnR is described below. For a value t we return a point after t iterations.

Hit-and-Run(P, p): Convex polytope P, current point $p \in P$
Pick a uniformly distributed line ℓ through p
Return a uniform point on the chord $\ell \cap P$

For zonotopes each step in both Coordinate-Directions HnR and Random-Directions HnR solves the following LP to compute one extreme point on $\ell \cap P$: minimize α, $s.t. \, p + \alpha v = \sum_{i=1}^{k} \lambda_i g_i$, $-1 \leq \lambda_i \leq 1$. For the second extreme point, keep the same constraints and minimize $-\alpha$. This LP uses the basic feasible solution of the first one.

Moreover, for zonotopes we study different types of convex bodies than ball for the MMC sequence. $G^T G$ has $k - d$ zero eigenvalues; the corresponding eigenvectors form matrix $Q \in \mathbb{R}^{k \times (k-d)}$. The intersection of the hypercube $[-1, 1]^k$ with the d-dimensional affine subspace defined by $Q^T = 0$ equals a d-dimensional polytope C in $\mathbb{R}^k$. SVD yields an orthonormal basis for the linear constraints, and its orthogonal complement $W_\perp$:

$$Q = USV^T = \begin{bmatrix} W \\ W_\perp \end{bmatrix}^T \begin{bmatrix} S_1 & 0 \\ 0 & 0 \end{bmatrix} V^T.$$

Let $Ay \leq b$, $A \in \mathbb{R}^{2k \times k}$ be an H-representation of $[-1, 1]^k$, then $Mx \leq b$, $M = AW_\perp^T (GW_\perp^T)^{-1} \in \mathbb{R}^{d \times d}$ is an H-representation of the full-dimensional, centrally symmetric polytope $C \subseteq P$ with $\leq 2k$ facets to be used in MMC. Each C_i arises from parallel shifting of the facets of C. This C improves the schedule when order is low, i.e. ≤ 4.

3 Implementation and experiments

We perform extended experiments analyzing various aspects of our method such as practical complexity and how is affected by the bodies used in MMC and we compare our implementation with the `matlab` code of [2] and `C++` package `VolEsti` [5]. Our `C++` software is open source[1]. When we use balls in MMC we call our implementation `CoolingBall` and when we use the H-polytope for zonotopes we call it `CoolingHpoly`. We call the implementation of [2] `CoolingGaussian` and that of [5] `SeqOfBalls`.

Set $r = 0.1$ and $\delta = 0.05$ in order for the next convex body in MMC to have about 10% of the volume of the previous one; let s.l. be $\alpha = 0.10$. We set the number of points sampled in P_i per step to be $\nu N = 1200 + 2d^2$ and $\nu = 10$. Set the length of the sliding window $n = 2d^2 + 250$ and the step of HnR $t = 1$.

[1] `https://github.com/TolisChal/volume_approximation/tree/v_poly`

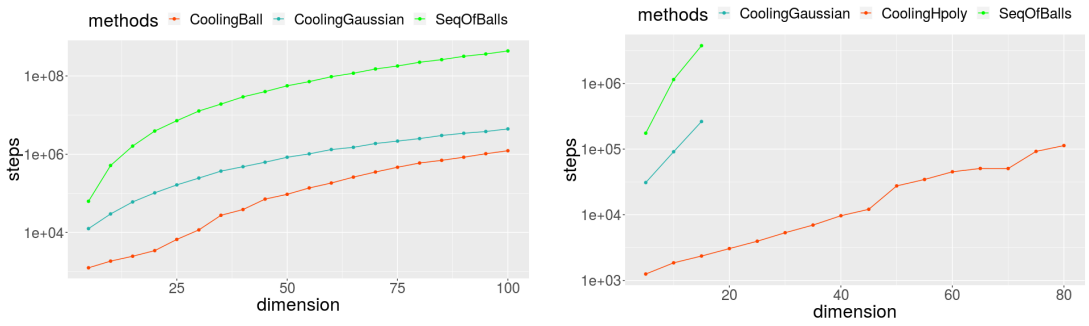


Figure 2 Left: the number of steps for for unit cubes in H-representation, $d = 5, 10, \ldots, 100$. Right: the number of steps for random zonotopes of order 2, $d = 5, 10, \ldots, 80$. In both plots we use log_{10} scale for the y-axis

To study the practical complexity of our method we experimentally correlate the total number of HnR steps with the dimension. In the right plot in Fig. 2 we compare the number of steps for random zonotopes. `CoolingGaussian` fails to estimate volumes for $d > 15$ as the upper bound for the number of facets is the bottleneck for this implementation while `SeqOfBalls` takes $> 1hr$ for $d > 15$. In the left plot we notice that our method is faster than both `CoolingGaussian` and `SeqOfBalls` for $d \leq 100$. In Table 1 we estimate the volumes of random zonotopes. The number of phases for high-order zonotopes is $m = 1$ as our methods defines just an enclosing ball and applies rejection sampling, whereas for low-order zonotopes the H-polytope we defined reduces significantly the number of phases and run-time. The maximum number of phases for zonotopes (up to what our software computes in $<$ 10hr) is $m \leq 3$. None of the volumes of the zonotopes in Table 1 can be computed using exact computations in practice. To define a random zonotope z-d-k we choose a random direction for each segment $s \in S$, where $\sum_{s \in S} s$, and pick a random length in the interval $[0, \sqrt{d}]$.

z-d-k	*Body*	*order*	Vol	m	e	*steps*	*time*
z-10-1000	Ball	100	2.62e+29	1	0.1	0.1400e+04	130.1
z-15-1500	Ball	100	5.00e+45	1	0.1	0.1650e+04	506.1
z-20-2000	Ball	100	2.79e+62	1	0.1	0.2000e+04	1428
z-60-90	Hpoly	1.5	5.81e+82	2	0.1	5.355e+04	943.9
z-80-120	Hpoly	1.5	8.48e+114	3	0.1	12.35e+04	4180
z-100-150	Hpoly	1.5	2.32+149	3	0.1	15.43e+04	10060
z-80-160	Hpoly	2	2.01e+131	3	0.2	11.31e+04	5356
z-100-200	Hpoly	2	5.27e+167	3	0.2	15.25e+04	34110

Table 1 *Body* the type of body in MMC; *order* is k/d, Vol the estimated volume; m the number of phases in MMC; ϵ the requested error; *time* the time in seconds; e the input value for error.

Acknowledgements. AC and IE acknowledge support by the EU Horizon 2020 research and innovation programme under grant agreement No 734242 (Project LAMBDA). They are both members of team AROMATH joint between NKUA and INRIA Sophia-Antipolis (France).

References

1 B. Cousins and S. Vempala. Bypassing KLS: Gaussian cooling and an $o^*(n^3)$ volume algorithm. In *Proc. ACM STOC*, pages 539–548, 2015.

2 B. Cousins and S. Vempala. A practical volume algorithm. *Mathematical Programming Computation*, 8, June 2016.

3 H. Cramer. *Mathematical methods of statistics*. Princeton University Press, 1946.

4 M. Dyer, A. Frieze, and R. Kannan. A random polynomial-time algorithm for approximating the volume of convex bodies. *J. ACM*, 38(1):1–17, 1991. URL: `http://doi.acm.org/10.1145/102782.102783`.

5 I.Z. Emiris and V. Fisikopoulos. Practical polytope volume approximation. *ACM Trans. Math. Soft.*, 44(4):38:1–38:21, 2018. Prelim. version: Proc. Symp. Comp. Geometry, 2014. URL: `http://doi.acm.org/10.1145/3194656`, `doi:10.1145/3194656`.

6 E. Gover and N. Krikorian. Determinants and the volumes of parallelotopes and zonotopes. *Linear Algebra and its Applications*, 413:28–40, 2010.

7 L. Lovász, R. Kannan, and M. Simonovits. Random walks and an $o^*(n^5)$ volume algorithm for convex bodies. *Random Structures and Algorithms*, 11:1–50, 1997.

On the Complexity of Nesting Polytopes

Michael G. Dobbins[1], Andreas F. Holmsen[2], and Tillman Miltzow[3]

1 **Department of Mathematical Sciences, SUNY Binghampton**
2 **Department of Mathematical Sciences, KAIST**
3 **Department of Information and Computing Sciences, Utrecht University**

Abstract

Given two rational convex polytopes $A \subseteq B \subset \mathbb{R}^d$ and a number k where A is given by vertices and B is given by halfspaces, the Nested Polytope Problem asks whether there exists a polytope X with k vertices such that $A \subseteq X \subseteq B$. We prove that Nested Polytope Problem is $\exists\mathbb{R}$-complete, which implies that Nested Polytope Problem is not contained in the complexity class NP, unless $\exists\mathbb{R} = \text{NP}$. Although this result was, to the best of our knowledge, never pointed out explicitly, it follows from some known results easily, as we will explain [17, 8].

1 Introduction

Given two rational convex polytopes $A \subseteq B \subset \mathbb{R}^d$ and a number k where A is given by vertices and B is given by halfspaces, the Nested Polytope Problem asks whether there exists a polytope X with k vertices such that $A \subseteq X \subseteq B$. The earliest mention of this problem that we know of is by Silio in 1979 [18], who found an $O(nm)$ time algorithm for nesting a triangle between an n-gon and m-gon. Independently Victor Klee suggested the same problem as was pointed out in several papers [11, 2, 9, 16, 10]. Gillis and Glineur showed that the Nested Polytope Problem is polynomial time equivalent to a variant of the Non-negative Matrix Factorization (NMF) problem called Restricted NMF [13]. These problems respectively generalize the Intermediate Simplex problem, where the polytopes A and B are required to be full dimensional and $k = d + 1$, and a special case of NMF called Exact NMF. Vavasis showed that these two problems are polynomial time equivalent to each other, and are NP-hard [19]. Yannakakis showed that NMF is a generalization of the extension complexity problem for polytopes. More specifically, the non-negative rank of the slack-matrix of a polytope corresponds precisely to the extension complexity of the polytope defined by the set of defining linear constraints. Thereby he gave lower bounds on the size of *symmetric* linear programs needed to describe certain combinatorial problems such as the Traveling Salesman problem [20], see also [12] for the asymmetric case. Yannakakis's paper may be celebrated for showing that a swath of fruitless attempts to prove P= NP are untenable. This situation is laid out in Figure 1. Our main contribution is an independent proof that the Nested Polytope Problem is $\exists\mathbb{R}$-complete by a simple geometric construction.

Note that although this seems never to be pointed out explicitly, the result is **not novel**. In 2016, it was shown elegantly by Shitov that NMF is $\exists\mathbb{R}$-complete [17]. Cohen and Rothblum described already in 1993 a simple polynomial reduction from NMF to the Nested Polytope Problem [8].

▶ **Theorem 1.1.** *The* Nested Polytope Problem *is* $\exists\mathbb{R}$*-complete.*[1]

[1] Our method of proof also implies a universality theorem similar to Mnëv's theorem for oriented matroids, but we do not include it in this abstract.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

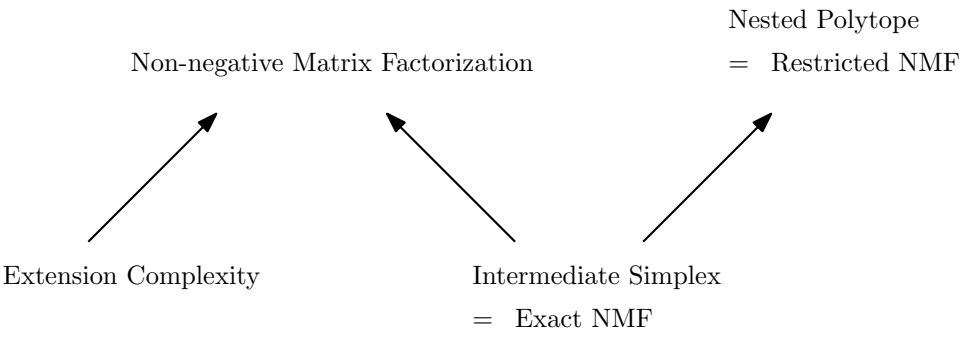


Figure 1 The long and winding road from extension complexity to nested polytopes.

For more work on Nested Polytope Problem and NMF, we refer the reader to the following literature [2, 5, 7, 14, 4, 13, 15, 3, 6].

The proof works in two steps. As a first step, we introduce a variant of the existential theory of the reals, denoted ETR-INV-array, and we show this variant is $\exists\mathbb{R}$-complete. This is described in Section 2. It ensures that we only have to encode algebraic relations that have a specific form. In the second step, we define gadgets, which are small Nested Polytope Problem instances where the coordinates of certain vertices of the nested polytope X are forced to satisfy the algebraic relations from the first step, and then we assemble these small gadgets to define a Nested Polytope Problem instance corresponding to each ETR-INV-array instance.

2 Encoding ETR

As a first step to encode an instance of the existential theory of the reals as an instance of the Nested Polytope Problem, we first simplify the algebraic structure.

An instance $\mathcal{A}$ of ETR-INV-array of size $m \times n$ is an m-by-n matrix A of variables $\alpha_{i,j}$ together with a system of equations of the form

$$\alpha_{i,j} + \alpha_{i,k} = \tfrac{5}{2} \quad , \quad \alpha_{i,j} + \alpha_{i,k} + \alpha_{i,l} = \tfrac{5}{2} \quad , \quad \alpha_{i,k} \cdot \alpha_{j,k} = 1.$$

Note that the linear equations relate variables in the same row and the quadratic equations relate variables in the same column. A solution to $\mathcal{A}$ is an assignment of values $\alpha_{i,j} \in [\frac{1}{2}, 2]^{m \times n}$ to each variable that satisfies each equation of $\mathcal{A}$. The corresponding decision problem asks whether an instance of $\mathcal{A}$ has a solution.

▶ **Lemma 2.1.** ETR-INV-array *is* $\exists\mathbb{R}$*-complete.*

This can be proven by introducing intermediate variables. For example, the relation $\alpha_{i,j} + \alpha_{i,k} = \alpha_{i,l}$ could be obtained by introducing a variable $\alpha_{i,m}$ and using the equations $\alpha_{i,j} + \alpha_{i,k} + \alpha_{i,m} = \frac{5}{2}$ and $\alpha_{i,m} + \alpha_{i,l} = \frac{5}{2}$. A similar reduction is given in [1, Lemma 12].

3 Building the polytopes

This section is devoted to show the following lemma. Together, Lemma 2.1 and Lemma 3.1 establish Theorem 1.1.

▶ **Lemma 3.1.** *Let* $\mathcal{A}$ *be an* ETR-INV-*array of size* $m \times n$*. There exists convex polytopes* $A \subset B \subset \mathbb{R}^{2+n+m}$ *such that there exists a nested polytope* $A \subset X \subset B$ *with* $k = mn + 2m + 2$ *vertices, if and only if* $\mathcal{A}$ *has a solution.*

▶ Remark. In fact we will show something stronger. In our construction, the polytopes $A \subset B$ in Theorem 3.1 will have precisely $2m + 2$ vertices in common. It follows that any nested polytope $A \subset X \subset B$ must contain these common vertices. Thus X will have $m \cdot n$ remaining vertices, and our construction will force these remaining vertices to be contained in certain segments of some of the edges of the outer polytope B. By parametrizing each of these segments by the interval $[\frac{1}{2}, 2]$ we obtain a correspondence between the remaining $m \cdot n$ vertices and a subset of $[\frac{1}{2}, 2]^{m \cdot n}$. The key step in the proof of Lemma 3.1 is to show that a positioning of the remaining vertices of X gives us $A \subset X \subset B$, if and only if those vertex positions correspond to a solution of $\mathcal{A}$.

3.1 Two geometric observations

Here we state two simple geometric observations that are used for the "gadgets" needed in our construction of the polytopes of Lemma 3.1. The proofs are simple calculations and left to the reader.

Let $\{v_0, v_1, \ldots, v_k\}$ be a set of linearly independent points in $\mathbb{R}^d$. For $1 \leq i \leq k$ let $w_i = v_i + v_0$ and define the prism P as

$$P = \text{conv}(\{v_1, \ldots, v_k, w_1, \ldots, w_k\}).$$

For $t \in [0, 1]$ define the point $q_t \in P$ as

$$q_t = (1 - t)(\tfrac{1}{k}v_1 + \cdots + \tfrac{1}{k}v_k) + t(\tfrac{1}{k}w_1 + \cdots + \tfrac{1}{k}w_k) = \tfrac{1}{k}v_1 + \cdots + \tfrac{1}{k}v_k + tv_0.$$

Finally, for $1 \leq i \leq k$ define points p_i as

$$p_i = (1 - \lambda_i)v_i + \lambda_i w_i = v_i + \lambda_i v_0,$$

where $\lambda_i \in [0, 1]$. We have the following.

▶ **Observation 3.2.** *$q_t \in conv(\{p_1, \ldots, p_k\})$ if and only if $\sum_{i=1}^{k} \lambda_i = tk$.*

▶ **Observation 3.3.** *Let $\alpha_1, \alpha_2 \in [\frac{1}{2}, 2]$ and $p_1 = (\alpha_1, -1)$ and $p_2 = (-1, \alpha_2)$. Then it holds that the origin $(0, 0) \in conv(\{p_1, p_2\})$ if and only if $\alpha_1 \cdot \alpha_2 = 1$.*

3.2 A basic outline of the construction

We now give an outline of the construction of the polytopes in Lemma 3.1, without giving explicit coordinates, and rather focusing on the three "gadgets" that will be used to encode the three types of equations in $\mathcal{A}$.

3.2.1 The outer polytope

The outer polytope B is a product of an orthogonal simplex of dimension m with a regular simplex of dimension $n + 1$. That is, we start with an *"orthogonal frame"* spanning $\mathbb{R}^m$, consisting of m mutually orthogonal segments of length 3 all meeting in a common point. Note that the convex hull of these segments form an m-dimensional orthogonal simplex. We now take $n + 2$ distinct copies of the orthogonal frame, U_1, U_2, $V_1, \ldots, V_n$, each one translated into "independent dimensions" so that their union now lives in $\mathbb{R}^{2+n+m}$. We label

the segments of these orthogonal frames as

$$\begin{aligned} U_1 &= \{\tau_{1,1}, \ldots, \tau_{m,1}\} \\ U_2 &= \{\tau_{1,2}, \ldots, \tau_{m,2}\} \\ V_1 &= \{\sigma_{1,1}, \ldots, \sigma_{m,1}\} \\ &\vdots \\ V_n &= \{\sigma_{1,n}, \ldots, \sigma_{m,n}\} \end{aligned}$$

such that the segments $\tau_{i,1}, \tau_{i,2}, \sigma_{i,1}, \ldots, \sigma_{i,n}$ are all parallel.

We now take the outer polytope B to be the convex hull of $U_1 \cup U_2 \cup V_1 \cup \cdots \cup V_n$. It is straight-forward to show that B is an $n+m+1$-dimensional polytope with $(n+2)(m+1)$ vertices. In what follows, for each $1 \leq i \leq m$ and $1 \leq j \leq n$, the "second half" of segment $\sigma_{i,j}$, parametrizing the interval $[\frac{1}{2}, 2]$, will correspond to the variable $\alpha_{i,j}$ in ETR-INV-array $\mathcal{A}$. The segments $\tau_{i,j}$ will play an auxiliary role which we now describe.

3.2.2 Building the inner polytope: Enforcing vertices to segments

The first step in building the inner polytope A is to enforce the following.

▶ Property 3.4. Let X be a nested polytope, with $k = mn + 2m + 2$ vertices and $A \subset X \subset B$. For every $1 \leq i \leq m$ and $1 \leq j \leq n$, the segment $\sigma_{i,j} \in V_j$ contains exactly one vertex of X, which we denote by $x_{i,j}$.

(More specifically, each segment of the orthogonal frame V_i will contain exactly one vertex from X in its "second half", thus encoding a value in the interval $[\frac{1}{2}, 2]$.) This can be done as follows. Fix indices $1 \leq i \leq m$ and $1 \leq j \leq n$, and consider segment $\tau_{i,1} \in U_1$ and its parallel copy $\sigma_{i,j} \in V_j$, which are edges of a 2-dimensional face of the outer polytope B. Define the point $y_{i,j}$ to be the unique point in this 2-face such that segment $\tau_{i,1} \in U_1$ is mapped to the second half of its parallel copy $\sigma_{i,j} \in V_j$ by central projection through $y_{i,j}$. Similarly, we define the analogous point $z_{i,j}$ in the 2-face of A spanned by the segment $\tau_{i,2} \in U_2$ and its parallel copy $\sigma_{i,j} \in V_j$. (See Figure 2.)

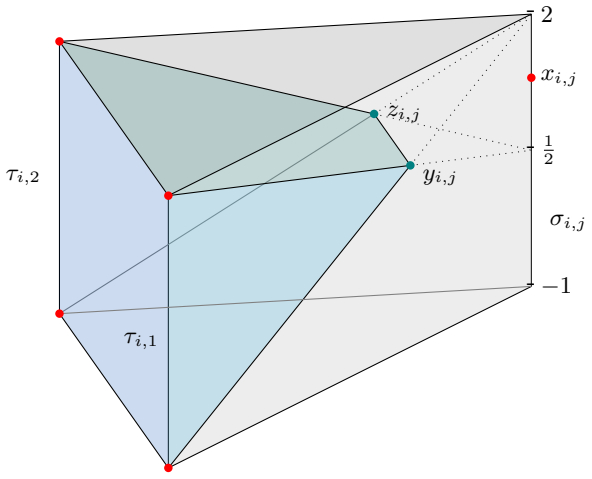


Figure 2 The vertices of any nested polytope $A \subset X \subset B$ (marked in red) must include the endpoints of segments $\tau_{i,1} \in U_1$ and $\tau_{i,2} \in U_2$, while the last vertex, $x_{i,j}$, must be contained in the segment $\sigma_{i,j} \in V_j$ restricted to the interval $[\frac{1}{2}, 2]$.

At this stage of the construction the inner polytope A will consist of the orthogonal frames U_1 and U_2 together with the points $\{y_{i,j}, z_{i,j}\}$ for all $1 \leq i \leq m$ and $1 \leq j \leq n$. Moreover, if X is a nested polytope, with $mn + 2m + 2$ vertices and $A \subset X \subset B$, then X must contain the orthogonal frames U_1 and U_2 (which accounts for $2m + 2$ of the vertices) and one vertex in each of the segments of the orthogonal frames $V_1, \ldots, V_n$ (accounting for the remaining $m \cdot n$ vertices). Thus Property 3.4 is satisfied, and we let $x_{i,j}$ denote the unique vertex of X which is contained in the (second half of the) segment $\sigma_{i,j} \in V_j$, which we associate with the variable $\alpha_{i,j} \in [\frac{1}{2}, 2]$.

3.2.3 Building the inner polytope: Encoding the linear equations

In order to enforce the relation $\alpha_{i,j} + \alpha_{i,k} = \frac{5}{2}$, we add a new vertex $p_{i,j,k}$ to the inner polytope A as follows. We consider the rectangular 2-face of the outer polytope B spanned by the segments $\sigma_{i,j} \in V_j$ and $\sigma_{i,k} \in V_k$. Define $p_{i,j,k}$ to be the point in this 2-face such that $p_{i,j,k}$ is contained in the convex hull of the vertices $x_{i,j}$ and $x_{i,k}$ of the nested polytope X (satisfying Property 3.4) if and only if the associated variables $\alpha_{i,j} + \alpha_{i,k} = \frac{5}{2}$. (The unique point $p_{i,j,k}$ exists by Observation 3.2. See Figure 3.)

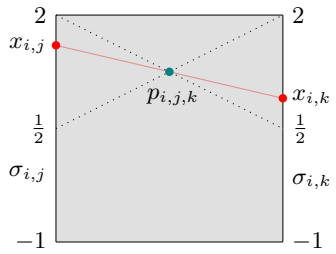


Figure 3 The vertices $x_{i,j}$ and $x_{i,k}$ contain the point $p_{i,j,k}$ in their convex hull if and only if the associated variables satisfy the equation $\alpha_{i,j} + \alpha_{i,k} = \frac{5}{2}$

By adding the vertex $p_{i,j,k}$ to A, it follows that for any nested polytope X satisfying Property 3.4, the associated variables satisfy the equation $\alpha_{i,j} + \alpha_{i,k} = \frac{5}{2}$.

Enforcing the relation $\alpha_{i,j} + \alpha_{i,k} + \alpha_{i,l} = \frac{5}{2}$ is similar to the previous case, and we add a new vertex $q_{i,j,k,l}$ to the inner polytope A as follows. We consider the triangluar prism spanned by the segments $\sigma_{i,j} \in V_j$, $\sigma_{i,k} \in V_k$, and $\sigma_{i,l} \in V_l$, which is a 3-face of the outer polytope B.

Define $q_{i,j,k,l}$ to be the point in this 3-face such that $q_{i,j,k,l}$ is contained in the convex hull of the vertices $x_{i,j}$, $x_{i,k}$, and $x_{i,l}$ of the nested polytope X (satisfying Property 3.4) if and only if the associated variables $\alpha_{i,j} + \alpha_{i,k} + \alpha_{i,l} = \frac{5}{2}$. (The unique point $q_{i,j,k,l}$ exists by Observation 3.2. See Figure 4.)

By adding the vertex $q_{i,j,k,l}$ to A, it follows that for any nested polytope X satisfying Property 3.4, the associated variables satisfy the equation $\alpha_{i,j} + \alpha_{i,k} + \alpha_{i,l} = \frac{5}{2}$.

3.2.4 Building the inner polytope: Encoding the quadratic equations

In order to enforce the relation $\alpha_{i,k} \cdot \alpha_{j,k} = 1$ we add a new vertex $r_{i,j,k}$ to the inner polytope A as follows. Consider the triangular 2-face of B spanned by segments $\sigma_{i,k} \in V_k$ and $\sigma_{j,k} \in V_k$. We can coordinatize the plane containing this 2-face such that the segment $\sigma_{i,k}$ is parametrized by $\{(x, -1) : -1 \leq x \leq 2\}$ and the segment $\sigma_{j,k}$ is parametrized by $\{(-1, y) : 1 \leq y \leq 2\}$. We then define $r_{i,j,k}$ to be the origin with respect to this coordinate

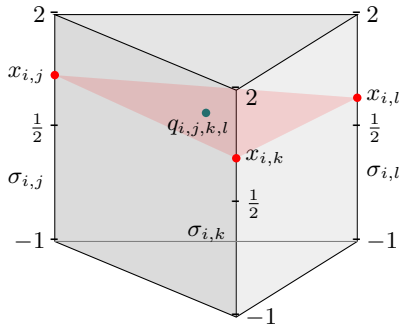


Figure 4 The vertices $x_{i,j}$, $x_{i,k}$, and $x_{i,l}$ contain the point $q_{i,j,k,l}$ if and only if the associated variables satisfy the equation $\alpha_{i,j} + \alpha_{i,k} + \alpha_{i,l} = \frac{5}{2}$.

system. It follows from Observation 3.3 that the vertices $x_{i,k}$ and $x_{j,k}$ contain the point $r_{i,j,k}$ in their convex hull if and only if the associated coordinates satisfy the equation $\alpha_{i,k} \cdot \alpha_{j,k} = 1$. (See Figure 5.)

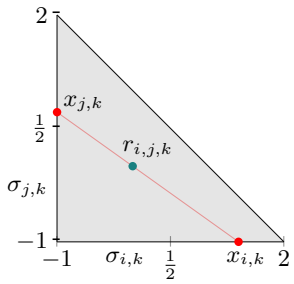


Figure 5 The vertices $x_{i,k}$ and $x_{j,k}$ contain the point $r_{i,j,k}$ if and only if the associated variables satisfy the equation $\alpha_{i,k} \cdot \alpha_{j,k} = 1$.

By adding the vertex $r_{i,j,k}$ to A, it follows that for any nested polytope X satisfying Property 3.4, the associated variables satisfy the equation $\alpha_{i,k} \cdot \alpha_{j,k} = 1$.

References

1 Mikkel Abrahamsen, Anna Adamaszek, and Tillmann Miltzow. The art gallery problem is $\exists\mathbb{R}$-complete. In *Symposium on Theory of Computing, STOC 2018*, pages 65–73, 2018. arxiv 1704.06969. URL: `http://doi.acm.org/10.1145/3188745.3188868`, `doi:10.1145/3188745.3188868`.

2 Alok Aggarwal, Heather Booth, Joseph O'Rourke, Subhash Suri, and Chee K. Yap. Finding minimal convex nested polygons. *Information and Computation*, 83(1):98–110, 1989. also appeared at the first symposium on Computational geometry in 1985.

3 Sanjeev Arora, Rong Ge, Ravi Kannan, and Ankur Moitra. Computing a nonnegative matrix factorization - provably. *SIAM J. Comput.*, 45(4):1582–1611, 2016. a preliminary version appeared at STOC 2012. URL: `https://doi.org/10.1137/130913869`, `doi:10.1137/130913869`.

4 A Berman. Rank factorization of nonnegative matrices. *SIAM Review*, 15(3):655, 1973.

5 Hervé Brönnimann and Michael T. Goodrich. Almost optimal set covers in finite vc-dimension. *Discrete & Computational Geometry*, 14(4):463–479, 1995.

6 Dmitry Chistikov, Stefan Kiefer, Ines Marusic, Mahsa Shirmohammadi, and James Worrell. Nonnegative matrix factorization requires irrationality. *SIAM Journal on Applied Algebra and Geometry*, 1(1):285–307, 2017. previous versions appeared at SODA 2017 and ICALP 2016.

7 Kenneth L. Clarkson. Algorithms for polytope covering and approximation. In *Workshop on Algorithms and Data Structures*, pages 246–252. Springer, 1993.

8 Joel E. Cohen and Uriel G. Rothblum. Nonnegative ranks, decompositions, and factorizations of nonnegative matrices. *Linear Algebra and its Applications*, 190:149–168, 1993.

9 Gautam Das. *Approximation schemes in computational geometry*. PhD thesis, The University of Wisconsin-Madison, 1990.

10 Gautam Das and Michael T. Goodrich. On the complexity of optimization problems for 3-dimensional convex polyhedra and decision trees. *Comput. Geom.*, 8(3):123–137, 1997.

11 Gautam Das and Deborah Joseph. The complexity of minimum convex nested polyhedra. In *Proc. 2nd Canad. Conf. Comput. Geom*, pages 296–301, 1990.

12 Samuel Fiorini, Serge Massar, Sebastian Pokutta, Hans Raj Tiwary, and Ronald De Wolf. Linear vs. semidefinite extended formulations: exponential separation and strong lower bounds. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 95–106. ACM, 2012.

13 Nicolas Gillis and François Glineur. On the geometric interpretation of the nonnegative rank. *Linear Algebra and its Applications*, 437(11):2685–2712, 2012.

14 Joseph S.B. Mitchell and Subhash Suri. Separation and approximation of polyhedral objects. *Computational Geometry*, 5(2):95–114, 1995.

15 Ankur Moitra. An almost optimal algorithm for computing nonnegative rank. *SIAM J. Comput.*, 45(1):156–173, 2016. URL: `https://doi.org/10.1137/140990139`, `doi:10.1137/140990139`.

16 Joseph O'Rourke. The computational geometry column# 4. *ACM SIGGRAPH Computer Graphics*, 22(2):111–112, 1988.

17 Yaroslav Shitov. A universality theorem for nonnegative matrix factorizations. *Preprint, https://arxiv.org/abs/1606.09068*, 2016.

18 Charles B. Silio Jr. An efficient simplex coverability algorithm in E^2 with application to stochastic sequential machines. *IEEE Trans. Computers*, 28(2):109–120, 1979.

19 Stephen A. Vavasis. On the complexity of nonnegative matrix factorization. *SIAM Journal on Optimization*, 20(3):1364–1377, 2009.

20 Mihalis Yannakakis. Expressing combinatorial optimization problems by linear programs. *Journal of Computer and System Sciences*, 43(3):441–466, 1991.

Green-Wins Solitaire Revisited — Simultaneous Flips that Affect Many Edges*

Michael Hoffmann[†1], **János Pach**[2], **and Miloš Stojaković**[‡3]

1 **Department of Computer Science, ETH Zürich**
`hoffmann@inf.ethz.ch`
2 **Department of Mathematics, EPF Lausanne and Rényi Institute, Budapest**
`pach@cims.nyu.edu`
3 **Department of Mathematics and Informatics, University of Novi Sad**
`milos.stojakovic@dmi.uns.ac.rs`

Abstract

We study the Green-Wins Solitaire game, which is a single player edge flipping game played on a given edge-colored geometric triangulation. An edge is flippable if it is a diagonal of a convex quadrilateral, and a flip replaces it by the other diagonal of that quadrilateral. Initially all edges are colored black. A move consists of flipping a black edge and coloring the resulting new edge along with all four edges of the surrounding convex quadrilateral green. The goal is to maximize the number of green edges. We show that in every triangulation on n vertices, for n sufficiently large, at least a fraction of $5/18 \approx 0.277$ edges can be colored green. On the other hand, there exist infinitely many triangulations in which no more than a $1/3$ fraction of edges can be colored green. These results improve earlier bounds of Aichholzer et al. [1].

1 Introduction

In this paper, the term *triangulation* denotes a maximal geometric plane graph: all edges are realized as straight-line segments and all bounded faces are triangles. Conversely, a *triangle* in a triangulation is a bounded facial triangle. Aichholzer et al. [1] studied various games related to triangulations, in particular, the *Green-Wins Solitaire* game. This game is played on a given triangulation, which we consider as edge-colored.

Edge flips. An edge in a triangulation is *flippable* if the union of the two incident faces forms a convex quadrilateral. *Flipping* a flippable edge amounts to replacing said edge by the other diagonal of the surrounding convex quadrilateral; see Figure 1. We say that these five edges, the flipped edge and the four edges of the surrounding quadrilateral, are *affected* by the flip. Edge flips are among the most prominent and well-studied operations for local modification of triangulations and, more generally, planar subdivisions. They serve as a crucial tool in many applications, such as counting and sampling, or optimization, for instance, to compute Delaunay triangulations; see, e.g., the survey by Bose and Hurtado [2].

* This work was started at the 13th Gremo's Workshop on Open Problems (GWOP), June 1–5, 2015, in Feldis (GR), Switzerland. We thank May Szedlák and Antonis Thomas for helpful discussions.
† Research partly supported by the Swiss National Science Foundation within the collaborative DACH project *Arrangements and Drawings* as SNSF Project 200021E-171681.
‡ Research partly supported by Ministry of Education, Science and Technological Development, Republic of Serbia, and Provincial Secretariat for Higher Education and Scientific Research, Province of Vojvodina.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

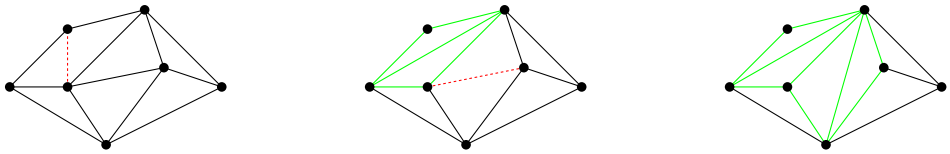


Figure 1 A sequence of moves in Green-Wins Solitaire. The flipped edge is shown red dotted.

Green-Wins Solitaire. Initially all edges are colored black. A move consists of picking a flippable black edge. This edge is flipped, and then all edges that are affected by the flip are colored green; see Figure 1. As green edges cannot be picked anymore, the set of edges flipped over the course of the game is *simultaneously flippable* [3], that is, the convex quadrilaterals surrounding the flipped edges are pairwise interior-disjoint. It also follows that the order of edge flips in a game is irrelevant, and we can describe every strategy as a set of (simultaneously flippable) edges.

For a triangulation T let $\gamma(T)$ denote the ratio of edges of T that can be colored green. Similarly, let $\gamma := \inf_T \gamma(T)$, where T is sufficiently large so as to exclude trivial cases like a single triangle or a K_4 (where no edge can be flipped). Aichholzer et al. [1] show that $1/6 \leq \gamma \leq 5/9$ and specifically ask whether $\gamma \geq 1/2$. They also show that an optimal set of edges to flip can be computed in linear time for convex point sets.

Improvements. The lower bound $\gamma \geq 1/6$ uses a lower bound for the number of simultaneously flippable edges in any triangulation by Galthier et al. [3]. Later Souvaine et al. [7] improved this bound by showing that in any (geometric) triangulation on n vertices at least $(n-4)/5$ edges are simultaneously flippable, which is best possible in general [3]. Plugging this bound into the argument of Aichholzer et al. [1] immediately gives $\gamma \geq 1/5 = 0.2$. Our goal in the following is to further improve this lower bound to $5/18 \approx 0.277 > 1/4$.

▶ **Theorem 1.** *In every triangulation of n points, for n sufficiently large, there exists a simultaneously flippable set of edges that affects at least a $5/18$ fraction of all edges.*

Before attacking the lower bound, let us note that the upper bound $\gamma \leq 5/9$ can be easily improved by considering families of triangulations for which the lower bound on the number of simultaneously flippable edges is tight (see Figure 2). It must have been an oversight that this was not observed by Aichholzer et al. [1] because the upper bound on the number of simultaneously flippable edges [3] was already known at that time.

▶ **Observation 2.** *For infinitely many $n \in \mathbb{N}$, there exists a triangulation on n points such that every simultaneously flippable set of edges affects at most $n-4$ out of $3n-6$ edges.*

In summary we have $5/18 \leq \gamma \leq 1/3$. The rest of the paper is devoted to derive the lower bound and prove Theorem 1.

2 Preliminaries

Sets of triangles. Consider a triangulation T on $n \geq 5$ vertices and a set U of triangles in T. The unbounded face is bounded by a cycle of r vertices that form the convex hull, where $3 \leq r \leq n$. By the Euler Formula, T has $3n-r-3$ edges and $2n-r-1$ faces, all but one of which are triangles. A vertex or edge of T is *interior* if it is not on the convex hull.

A vertex or edge of T is (1) *internal* to U if all incident faces are in U, (2) *external* to U if no incident face is in U, or (3) on the *boundary* of U if it is incident to at least one face

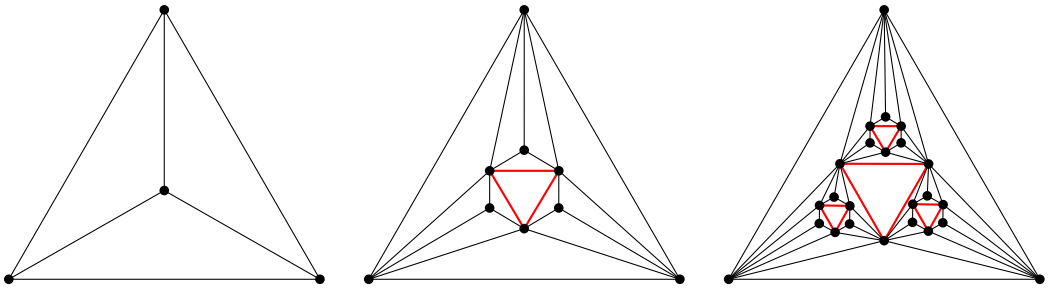


Figure 2 The upper bound construction for the number of simultaneously flippable edges by Souvaine et al. [7]. Recursively the central vertex of K_4 subconfigurations is replaced by a hexagon, connected as shown in (b). All flippable edges are shown in red; they appear in triangles. In each such triangle, no more than one edge can be selected for any simultaneous flip.

in U and at least one face that is not in U. Note that a vertex or edge on the convex hull is not internal to U by definition. See Figure 3a for illustration.

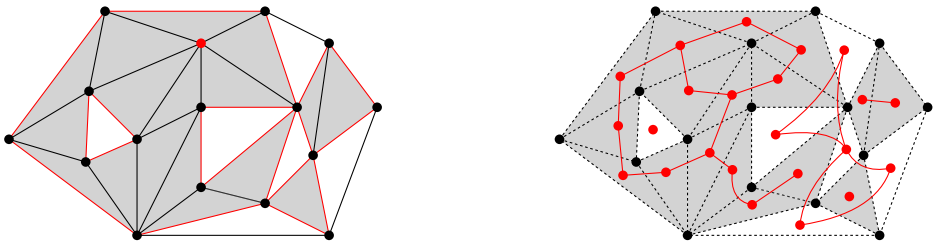


Figure 3 (a) A set U of gray triangles with 1 internal vertex (red), 16 internal edges, 3 components, 1 hole, and 19 boundary edges (red). Confirming Alpaca 3 we have $3\cdot 1+3\cdot 1+19 = 16+3\cdot 3$.

Let $T^*|_U$ denote the following graph on the triangles of T: Two triangles of T are connected in $T^*|_U$ if (1) they share an edge and are both in U or (2) they share a vertex and are both not in U. A *component* of U is a component of U in $T^*|_U$. A *hole* in U is a component of $T^*|_U \setminus U$ that is contained inside a cycle of triangles from U in $T^*|_U$. We obtain the following variation of the Euler Formula for sets of triangles in a triangulation.[1]

▶ **Alpaca 3.** *Consider a triangulation T and a subset U of triangles of T. Then*

$$3h + 3v_i + e_b = e_i + 3c,$$

where h denotes the number of holes in U, v_i denotes the number of internal vertices of U, e_b denotes the number of boundary edges of U, e_i denotes the number of internal edges of U and c denotes the number of components of U in $T^*|_U$.

Strategy. Let F denote a maximum size set of simultaneously flippable edges in T, and let U denote the set of triangles in T that are incident to an edge in F. We flip the $f := |F|$

[1] It is named after the cute animals that the authors watched when working on this problem and they realized: It is not a lama but an alpaca...

edges in F and color them green. For every flip, the four edges of the bounding quadrilateral are also colored green. However, each of these edges may be colored green twice overall in case the edge is incident to two triangles from U. Exactly the e_b edges on the boundary of U are colored once only. Therefore, the number of green edges after flipping F is

$$f + 4f/2 + e_b/2 = 3f + e_b/2. \tag{GF}$$

In order to bound this quantity, we want to obtain a good lower bound for e_b to combine with the known lower bound $f \geq (n-4)/5$.

▶ **Lemma 4.** *We have $e_b \geq 4c$.*

Proof. As T is a triangulation, every component of U in $T^*|_U$ is bounded by at least three edges. Furthermore, every component of U consists of pairs of triangles and hence an even number of triangles. A triangulation whose outer boundary forms a triangle corresponds to a maximal planar graph, which has $2n-5$ bounded faces—an odd number. Therefore, no component of U has a triangle as an outer boundary. ◀

3 Counting black edges: Proof of Theorem 1

We color the edges of T as follows. An edge e of T is colored *green* if it is incident to a triangle from U; otherwise, e is colored *black*. Note that we work with the original triangulation T, no edges are flipped. If a black edge was flippable, then we could flip it to color more edges green, contradicting the maximality of F. Hence no black edge is flippable. As a first step we observe that not too many vertices, relatively speaking, are on the convex hull.

▶ **Lemma 5.** *For every $\alpha \in (0,1)$ and $r \geq \frac{3\alpha}{1+\alpha}n$, at least an $(\alpha - \varepsilon)$ fraction of the edges in T are green, where $\varepsilon = \varepsilon(n)$ tends to zero when $n \to \infty$.*

Proof. At least $r-3$ edges are flippable in T [4]. Clearly, every flippable edge can be colored green: if it is not, then flip it. Hence, at least a fraction of $(r-3)/(3n-r-3)$ edges is colored green. This expression is monotonically increasing in r, for $3 \leq r \leq n$. Therefore,

$$\frac{r-3}{3n-r-3} \geq \frac{\frac{3\alpha}{1+\alpha}n - 3}{3n - \frac{3\alpha}{1+\alpha}n - 3} = \frac{\alpha n - (1+\alpha)}{n-(1+\alpha)} \geq \alpha - \frac{2}{n-2}$$

which converges to α, for $n \to \infty$. ◀

In the following, we investigate the subgraph B of T that is induced by the black edges. The vertices of B fall into three groups: vertices on the convex hull, vertices on the boundary of U, and *internal* vertices (vertices that are not on the convex hull and not incident to any green edge). An edge of B is *internal* if it is incident to two triangles in B. In particular, an internal edge is not a convex hull edge.

Following standard terminology [6], we call an edge $e = uv$ of T *separable* at its endpoint u if there exists a line ℓ through u so that e is the only edge of T incident to u on one side of ℓ. In other words, u is *pointed* (has a free angle $\geq \pi$) in $T \setminus e$. We observe (cf. [5, 6]):

(S1) Every unflippable edge is separable at one of its endpoints and only the convex hull edges are separable at both endpoints.

(S2) At an interior vertex v of degree ≥ 4, at most two incident edges are separable; if two incident edges are separable, then they are consecutive in the circular order around v.

In particular, (S1) implies that an edge that is incident to two convex hull vertices is either a convex hull edge or flippable. So no internal edge of B connects two convex hull vertices. By the following lemma, all internal vertices of B have degree 3 (in B and T).

▶ **Lemma 6** ([5, Lemma 4.2]). *In a triangulation, every interior (not on the convex hull) vertex of degree four or higher is incident to a flippable edge.*

In a triangulation, no two interior degree three vertices are adjacent. Let us bound the number of black edges. Every black edge connects two vertices from the abovementioned three groups: At most r edges are convex hull edges, and every edge that is incident to an internal vertex is incident to exactly one internal vertex (because every internal vertex has degree three). All remaining edges are incident to at least one vertex on the boundary of U but not on the convex hull (because there is no internal black edge between any two convex hull vertices). Denote the number of these remaining edges by $\overline{e_i}$, denote by d_3 the number of internal (hence degree-3) vertices of B, and denote by e_C the number of convex hull edges on the boundary of U. We select F so that the number d_3 is smallest over all maximum size sets of simultaneously flippable edges in T.

▶ **Lemma 7.** *We have* $3d_3 + e_C \leq 2e_b$.

Let us derive a bound for $\overline{e_i}$ by applying Alpaca 3 to $T \setminus U$. The boundary of $T \setminus U$ is formed by edges that are on the convex hull or on the boundary of U, and there are $(e_b - e_C) + (r - e_C) = e_b + r - 2e_C$ such edges. The edges incident to the d_3 internal vertices of B are not counted in $\overline{e_i}$, and so we have no internal vertices to consider. Every hole in $T \setminus U$ corresponds to at least one separate component of U, which in turn corresponds to a separate component of U in T, and so the number of holes is upper bounded by c. Altogether we obtain

$$3c + e_b + r - 2e_C \geq \overline{e_i}. \tag{1}$$

▶ **Lemma 8.** *There are no more than* $3e_b + 3c + 2r - 4e_C$ *black edges.*

Proof. We have exactly $r - e_C$ black convex hull edges, exactly $3d_3$ edges that are incident to an interior vertex, and exactly $\overline{e_i}$ other edges. Using Lemma 7 and (1), we can bound the number of black edges as $(r - e_C) + 3d_3 + \overline{e_i} \leq (r - e_C) + (2e_b - e_C) + (3c + e_b + r - 2e_C)$. ◀

▶ **Lemma 9.** *There are at least* $3(n - r - e_b - c - 1) + 4e_C$ *green edges.*

Proof. The total number of edges in T is $3n - r - 3$. Thus, by Lemma 8, we have at least $(3n - r - 3) - (3e_b + 3c + 2r - 4e_C) = 3(n - r - e_b - c - 1) + 4e_C$ green edges. ◀

If we head for a bound of $\gamma \geq \alpha$, then we may assume that

$$c \leq 3n\left(\frac{\alpha}{2} - \frac{1}{10}\right) - \frac{\alpha}{2}r.$$

Otherwise, by Lemma 4 and the lower bound $f \geq (n-4)/5$, we have

$$3f + \frac{1}{2}e_b \geq \frac{3}{5}n + 2c \geq \frac{3}{5}n + 3n\left(\alpha - \frac{1}{5}\right) - \alpha r \geq \alpha(3n - r - 3),$$

and we are done. In combination with (GF) and Lemma 9 we get

$$\begin{aligned} 3f + \frac{1}{2}e_b &\geq 3(n - r - e_b - c) \\ \frac{7}{2}e_b &\geq 3(n - r - f - c) \\ \frac{1}{2}e_b &\geq \frac{3}{7}(n - r - f - c) \end{aligned}$$

and, therefore,

$$\begin{aligned}
3f + \frac{1}{2}e_b &\geq 3f + \frac{3}{7}(n - r - f - c) \\
&= \frac{18}{7}f + \frac{3}{7}(n - r - c) \\
&\geq \frac{18}{35}n + \frac{3}{7}(n - r - c) \\
&= \frac{33}{35}n - \frac{3}{7}(r + c) \\
&\geq \frac{33}{35}n - \frac{3}{7}r - \frac{3}{7}\left(3n\left(\frac{\alpha}{2} - \frac{1}{10}\right) - \frac{\alpha}{2}r\right) \\
&= \frac{1}{70}\Big((75 - 45\alpha)n - (30 - 15\alpha)r\Big).
\end{aligned}$$

Thus, the fraction of green edges is at least

$$\frac{3f + \frac{1}{2}e_b}{3n - r} \geq \frac{(75 - 45\alpha)n - (30 - 15\alpha)r}{70(3n - r)} = \frac{1}{70}\left(30 - 15\alpha - \frac{15n}{3n - r}\right).$$

The fraction $\frac{-15n}{3n-r}$ is monotonically decreasing as a function of r and so it is minimized for r maximal, that is, $r = \frac{3\alpha}{1+\alpha}n$, due to Lemma 5. In this case, we obtain a fraction of

$$\frac{1}{70}\left(30 - 15\alpha - \frac{15n}{3n - \frac{3\alpha}{1+\alpha}n}\right) = \frac{1}{70}\Big(30 - 15\alpha - 5(1 + \alpha)\Big) = \frac{1}{14}(5 - 4\alpha).$$

Optimizing for α yields $\gamma \geq \alpha = 5/18 > 0.277$, which completes the proof of Theorem 1.

References

1 Oswin Aichholzer, David Bremner, Erik D. Demaine, Ferran Hurtado, Evangelos Kranakis, Hannes Krasser, Suneeta Ramaswami, Saurabh Sethia, and Jorge Urrutia. Games on triangulations. *Theoret. Comput. Sci.*, 343(1–2):42–71, 2005. URL: https://doi.org/10.1016/j.tcs.2005.05.007.

2 Prosenjit Bose and Ferran Hurtado. Flips in planar graphs. *Comput. Geom. Theory Appl.*, 42(1):60–80, 2009. URL: https://doi.org/10.1016/j.comgeo.2008.04.001.

3 Jérôme Galtier, Ferran Hurtado, Marc Noy, Stephane Perennes, and Jorge Urrutia. Simultaneous edge flipping in triangulations. *Internat. J. Comput. Geom. Appl.*, 13(2):113–133, 2003. URL: https://doi.org/10.1142/S0218195903001098.

4 Michael Hoffmann, André Schulz, Micha Sharir, Adam Sheffer, Csaba D. Tóth, and Emo Welzl. Counting plane graphs: Flippability and its applications. In János Pach, editor, *Thirty Essays on Geometric Graph Theory*, pages 303–325. Springer-Verlag, 2013. URL: https://doi.org/10.1007/978-1-4614-0110-0_16.

5 Ferran Hurtado, Marc Noy, and Jorge Urrutia. Flipping edges in triangulations. *Discrete Comput. Geom.*, 22(3):333–346, 1999. URL: https://doi.org/10.1007/PL00009464.

6 Micha Sharir and Emo Welzl. Random triangulations of planar point sets. In *Proc. 22nd Internat. Sympos. Comput. Geom.*, pages 273–281, 2006. URL: https://doi.org/10.1145/1137856.1137898.

7 Diane L. Souvaine, Csaba D. Tóth, and Andrew Winslow. Simultaneously flippable edges in triangulations. In *Computational Geometry—XIV Spanish Meeting on Computational Geometry*, volume 7579 of *Lecture Notes Comput. Sci.*, pages 138–145. Springer-Verlag, 2011. URL: https://doi.org/10.1007/978-3-642-34191-5_13.

Triangles in the colored Euclidean plane*

Oswin Aichholzer[1] and Daniel Perz[1]

1 **Institute of Software Technology, Graz University of Technology, Austria**
[oaich|daperz]@ist.tugraz.at

Abstract

We study a variation of the well known Hadwiger-Nelson problem on the chromatic number of the Euclidean plane. An embedding of a given triangle T into the colored plane is called monochromatic, if the three corners of the triangle get the same color. We provide a classification of triangles according to the number of colors needed to color the plane so that the triangle can not be embedded monochromatically. For example, we show that for near-equilateral triangles three colors are enough and that for almost all triangles six colors are sufficient.

1 Introduction

An *r-coloring* of the Euclidean plane is a partition such that every point of the plane gets one of r colors assigned. The famous Hadwiger-Nelson problem (see Soifer [10] for an extensive history) asks for the minimum r to r-color the Euclidean plane so that every two points at unit distance from each other have different colors. It was well known that this so-called chromatic number of the plane is at least 4, and that 7 colors are for sure sufficient. Most recently de Grey [1] constructed an explicit unit-distance graph with 1581 vertices with chromatic number 5. This was further optimized by Heule [5] to graphs with 'only' 553 vertices. So the chromatic number of the Euclidean plane is now 5, 6 or 7.

To shed more light on the Hadwiger-Nelson problem, Graham [3] posed the following question.

▶ Problem 1. What is the smallest number c, so that for every triangle T, there exists a c-coloring of the Euclidean plane so that it is not possible to embed T in such a way that all three vertices of T have the same color?

If a triangle T, or its reflected copy, can be embedded in the colored plane such that all three vertices have the same color, we call the embedding and the triangle monochromatic, and non-monochromatic otherwise. If the color of all three vertices is for example red, we also call the triangle red.

As for a lower bound on c it has been shown that some triangles exist monochromatically in every 2-coloring of the plane [2, 9]. For example in [2] they show that a triangle with smallest side of length 1 and angles 30°, 60°, and 90° is monochromatic in every 2-coloring. The following argumentation for this statement is based on Soifer [10]. As in any 2-coloring an equilateral triangle has two vertices with the same color, there always exist two points with the same color, say blue, with distance 2 (or any other fixed distances). Consider the corners of a regular hexagon where these two blue points are antipodal, see Figure 1. If any other corner of the hexagon is also blue, we are done. Otherwise, the remaining 4 corners are all red, and we have the required triangle in red. Surprisingly, this example constitutes the best known lower bound, and thus Graham conjectured that $c = 3$ could be the true bound.

* Partially supported by the Austrian Science Fund within the collaborative DACH project *Arrangements and Drawings* as FWF project I 3340-N35. Work based on the master thesis [7] of the second author.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

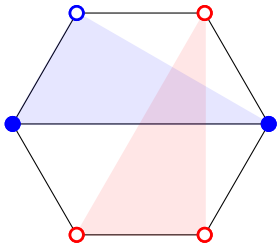


Figure 1 A triangle with 30°, 60°, and 90° always exists monochromatically in every 2-coloring.

Obviously an upper bound for the chromatic number of the plane is also an upper bound for c, and thus we have $c \in \{3, 4, 5, 6, 7\}$.

In this work we concentrate our considerations on specific triangles and how many colors guarantee their non-monochromatic embedding.

▶ Problem 2. Given a triangle T. What is the smallest number $c(T)$, so that we can $c(T)$-color the Euclidean plane, such that T can not be embedded monochromatically.

An upper bound on c in Problem 1 is also an upper bound for all $c(T)$ in Problem 2, and if we can show an upper bound for $c(T)$ for all triangles T then this implies an upper bound for c. For example in [6] it has been shown that the equilateral triangle with side length s is non-monochromatic in a 2-coloring with halfopen horizontal strips of height $\frac{\sqrt{3}}{2}s$. We call such a coloring *zebra* coloring. Also other colorings for equilateral triangles are provided in [6]. Pritikin [8] used colorings, where points with the same color and distance 1 only occur in one color. Modifying his 7-coloring so that it becomes a proper 6-coloring of the plane leads to results which are slightly worse than what we will go to present. In addition, in his 5-coloring almost all triangles occure monochromatic. Graham and Tressler [4] show bounds for degenerated triangles and mentioned that zebra colorings avoid a large class of triangles, without giving explicit bounds.

1.1 Results

In Section 2 we will consider triangles for which a 6-coloring exists such that a monochromatic embedding is not possible. We show that this holds for almost all triangles, thus lowering the upper bound for Problem 1 from 7 to 6 with the exception of almost isosceles triangles with a short base. In Section 3 we categorize triangles according to the number of colors which are needed to guarantee that a monochromatic embedding is not possible. We show that for near equilateral triangles 3 colors are sufficient. The flatter a triangle gets, the more colors are needed. Due to space limitations several proofs will be omitted in this note.

2 Non-monochromatic triangles in the 6-colored plane

Let T be a triangle with vertices A, B and C. We write AB for the line segment AB and $\overline{AB}$ for its length. Without loss of generality we assume that AB is the longest side of T.

▶ **Definition 2.1.** We denote the heights of T as depicted in Figure 2. We call T a normed triangle, if the longest side AB of T has length 1, and $\overline{BC} \leq \overline{AC}$ holds.

Observe that a triangle T with side lengths a, b and c is non-monochromatic in some r-coloring, if and only if there exists an r-coloring F, so that the triangle T_a with side lengths 1, $\frac{b}{a}$ and $\frac{c}{a}$ is non-monochromatic in F. Therefore we can restrict our investigations

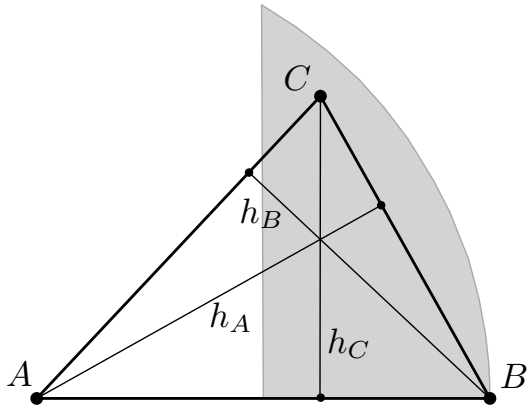


Figure 2 Normed triangle with $\overline{BC} \leq \overline{AC} \leq \overline{AB} = 1$. The grey area shows all possible locations for vertex C.

to normed triangles. The grey area in Figure 2 shows all possible locations of vertex C to be considered.

2.1 Zebra colorings

In a *zebra coloring* the plane is cyclically colored with horizontal strips, all of the same height. The strips are halfopen, that is, the boundary between two neighboured strips has the same color as the strip above it. For a normed triangle T we have that h_C is the shortest height of T. Thus the height of the strips has to be at most h_C, as otherwise T fits into one strip and obviously would be monochromatic. So assume that the height of each strip is exactly h_C. This implies that the vertices of the triangle have to lie in at least two different strips.

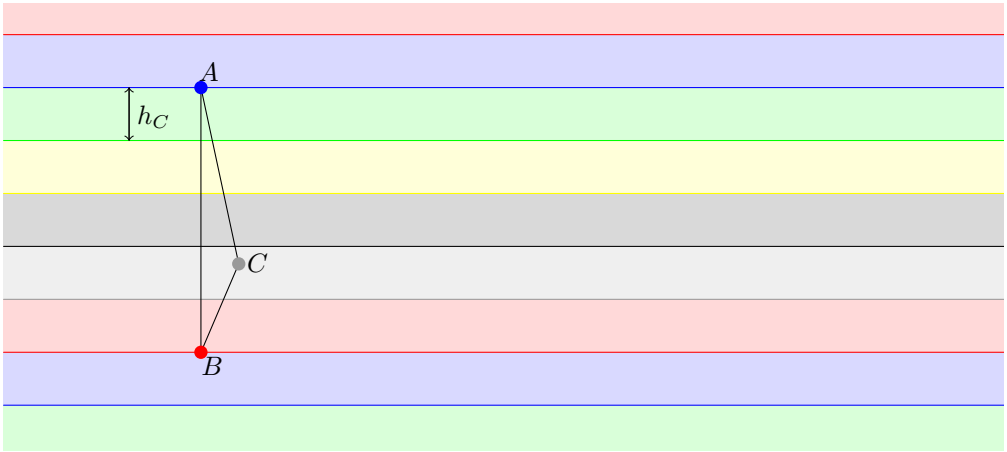


Figure 3 Zebra coloring with 6 colors with halfopen strips of height h_C

For 6 colors the distance between two points in two different strips with the same color is strictly larger than $5h_C$, because there are five strips between two strips with the same color. So if two points have the same color and have distance at most $5h_C$, then these two points have to lie in the same strip. Similarly, if a point of the triangle is the only one in a strip, then the distances to both other points are at least $5h_C$ if all the points are of the

same color. Therefore a normed triangle T is non-monochromatic if the second longest side $\overline{AC} \leq 5h_C$ and we obtain the following lemma.

▶ **Lemma 2.2.** *All normed triangles with $\overline{AC} \leq 5h_C$ are non-monochromatic in a zebra coloring with 6 colors, where all strips have height h_C.*

For triangles where every angle is at most 90° we can slightly improve this result. The basic idea is to place the triangle between two strips of the same color such that the longest height h_A is vertical. If we rotate the triangle around the vertex A until one of the sides AB or AC is vertical, then one of the two vertices B and C moves upwards while the other one moves downwards, and eventually one of them has to leave the strip. The details of the proof are omitted.

▶ **Theorem 2.3.** *Every normed triangle, in which every angle is at most 90° and $h_A \leq 5h_C$ is non-monochromatic in a zebra coloring with 6 colors, where all strips have height h_C.*

By comparing different expressions of the area of the triangle, we get the following corollary.

▶ **Corollary 2.4.** *Every normed triangle, in which every angle is at most 90°, and $\overline{BC} \geq \frac{1}{5}$ is non-monochromatic in a zebra coloring with 6 colors, where all strips have height h_C.*

2.2 Hexagon colorings

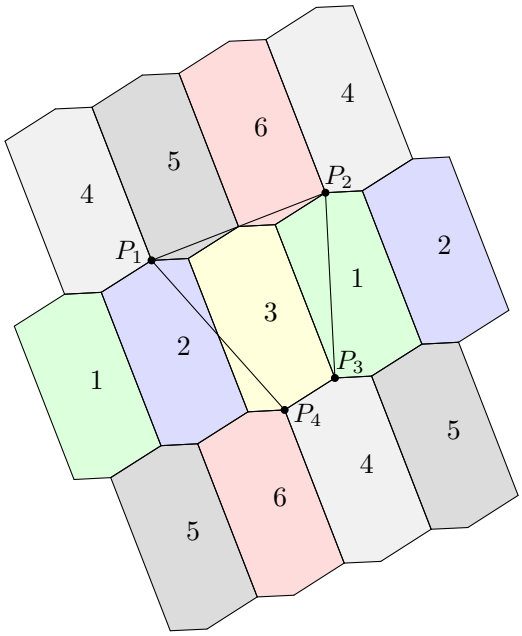
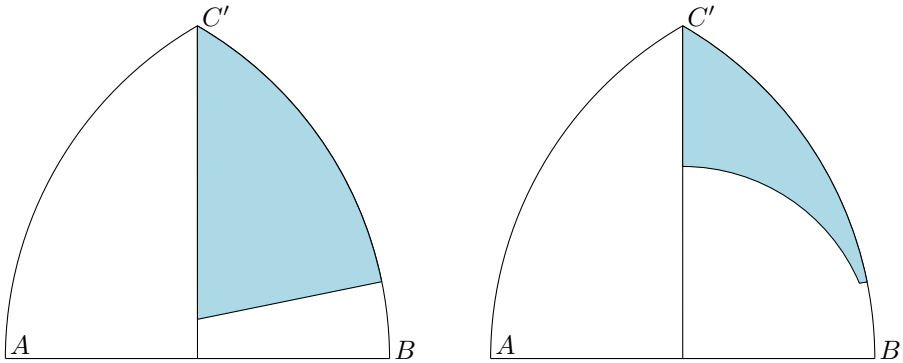


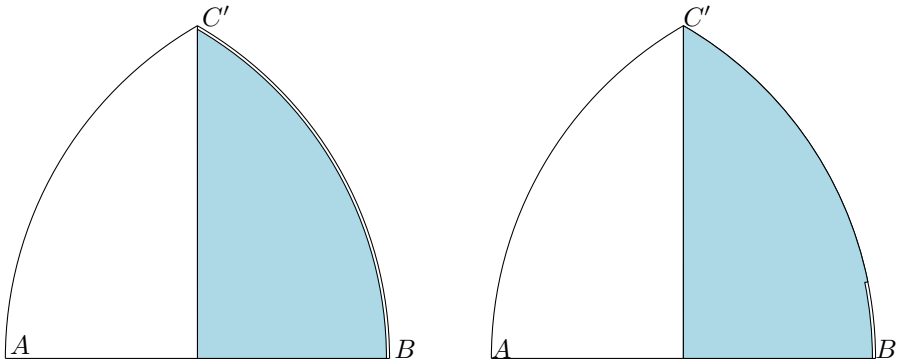
Figure 4 6-coloring with hexagons, where all diagonals have length 1 and opposing sides are parallel

Consider the hexagonal 6-coloring shown in Figure 4. The vertices of a hexagon lie on a circle with radius $\frac{1}{2}$ and are center symmetric. Thus the three central diagonals have length 1 and opposing sides are parallel. The hexagons are halfopen in the sense that we color the two lowest vertices of a hexagon as well as the three sides, which are incident to these



a Locations covered by Lemma 2.2

b Locations covered by Corollary 2.4



c Locations covered by Theorem 2.5

d Total area covered by our results

Figure 5 Possible locations for vertex C so that there exists a 6-coloring, such that the triangle ABC is non-monochromatic.

vertices, with the same color as the hexagon. For example in Figure 4 the vertex P_4 and side P_3P_4 have color 3, but P_3 has color 1.

Our goal is to maximize the shortest of the lengths $\overline{P_1P_2}$, $\overline{P_2P_3}$ and $\overline{P_1P_4}$, as they are also the lower bounds for the distance of two points in different hexagons with the same color. Details on how to compute the exact lengths of the hexagons will be given in the full version of this work.

▶ **Theorem 2.5.** *All normed triangles T with $\overline{AC} \leq 0.992076$ are non-monochromatic in the hexagon coloring in Figure 4 with specific lengths.*

2.3 Summarizing bounds

Let us look which types of normed triangles are covered by the previous results. Recall that the grey area in Figure 2 shows the relevant region of vertex C. Figures 5a to 5c show which locations are covered by our results, and Figure 5d gives their union.

For almost all triangles T there exists a 6-coloring so that T is non-monochromatic. Only for near-isosceles triangles with $0.992076\overline{AB} < \overline{AC} \leq \overline{AB}$ and $\overline{BC} < \frac{1}{5}\overline{AB}$ we have no such 6-coloring. This has to be seen in context of the Hadwiger-Nelson problem. For these triangles the vertex C is rather close to the end of the unit length segment AB, meaning that in the extreme case these triangles approach the segment.

Furthermore, Figure 5c shows that Theorem 2.5 covers almost all possible choices of C. For locations of C which are not covered by Theorem 2.5 we see that Corollary 2.4 covers

slightly more than Lemma 2.2. Actually Lemma 2.2 is not needed for 6-colorings, as for almost isosceles triangles, Corollary 2.4 is better than Lemma 2.2.

3 Non-monochromatic triangles with fewer colors

We have shown that for most triangles there exists a 6-coloring that prevents a monochromatic embedding. A natural question is to ask for which triangles less colors are sufficient. We can use zebra colorings to generalize Lemma 2.2 and Corollary 2.4. Similar to Lemma 2.2 we get.

▶ **Theorem 3.1.** *All normed triangles with* $\overline{AC} \leq (k-1)h_C$ *are non-monochromatic in a zebra coloring with* k *colors,* $3 \leq k \leq 6$*, where all strips have height* h_C*.*

Corollary 2.4 generalizes to

▶ **Theorem 3.2.** *All normed triangles for which every angle is at most* $90°$ *and* $\overline{BC} \geq \frac{1}{k-1}$ *are non-monochromatic in a zebra coloring with* k *colors,* $2 \leq k \leq 6$*. where all strips have height* h_C*.*

Since our arguments for the proofs of Lemma 2.2, Theorem 2.3 and Corollary 2.4 were about the distance between two points with the same color in different strips, we can prove these statements in a similar way. Actually we just need to replace 5 by $k-1$. Note that Theorem 3.2 also works for 2 colors, whereas Theorem 3.1 only works for 3 or more colors.

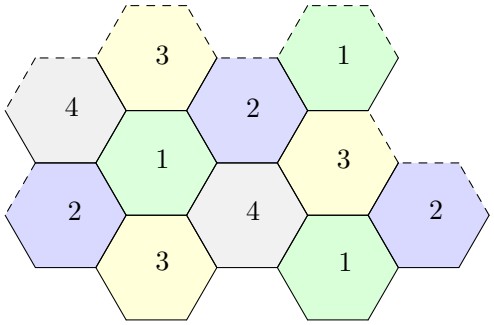
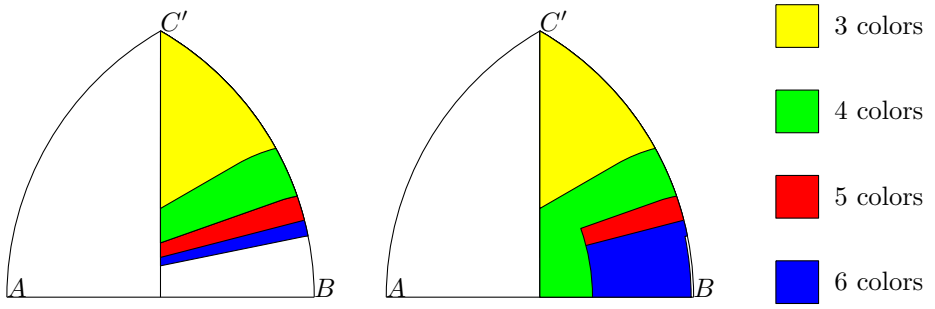


Figure 6 4-coloring with regular hexagons of diameter 1

In the 4-coloring in Figure 6 all hexagons have diameter 1. The three sides and two vertices below have the same color as the hexagon. This gives us the following theorem (proof omitted).

▶ **Theorem 3.3.** *In the 4-coloring in Figure 6, all normed triangles with* $\overline{AC} \leq \frac{\sqrt{3}}{2}$ *are non-monochromatic.*

As before we summarize the results in a diagam, see Figure 7. For all possible locations of the third vertex C of a normed triangle the colors indicate which k is sufficient so that a k-coloring exists such that the triangle is non-monochromatic in this coloring. For example, if C is in the yellow shaded area, then there exists a 3-coloring such that the triangle ABC is non-monochromatic. This includes all triangles where the length of the three edges does not differ too much. For Figure 7a only zebra colorings are used, while for Figure 7b also hexagonal colorings are allowed.



a Only zebra colorings **b** All colorings **c** Color code

Figure 7 Location of vertex C of triangles for which a k-coloring, $3 \leq k \leq 6$ exists so that they are non-monochromatic.

4 Conclusion

We have shown that for almost all triangles there exists a 6-coloring, such that the triangle is non-monochromatic in this coloring. To be precise, for every normed triangle with $\overline{AC} \leq 0.992076$ or $\overline{BC} \geq \frac{1}{5}$ we can give such a 6-coloring. We have also seen that some triangles are non-monochromatic in colorings with less than 6 colors. It remains an open problem, if for every triangle there exists a 6-coloring of the plane, such that the triangle is non-monochromatic. Or more generally, what is the smallest c so that there exists for every triangle a c-coloring, such that the triangle is non-monochromatic.

References

1. A. de Grey. The chromatic number of the plane is at least 5, 2018. `arXiv:1804.02385`.
2. P. Erdős, R. Graham, P. Montgomery, B. Rothschild, J. Spencer, and E. Straus. Euclidean Ramsey theorems. i. *Journal of Combinatorial Theory, Series A*, 14(3):341–363, 1973.
3. R. Graham. Problem 58: Monochromatic Triangles. http://cs.smith.edu/~orourke/TOPP/P58.html#Problem.58.
4. R. Graham and E. Tressler. Open problems in Euclidean Ramsey Theory. In A. Soifer, editor, *Ramsey Theory: Yesterday, Today, and Tomorrow*, pages 115–120. Birkhäuser Boston, Boston, MA, 2011.
5. M.J.H. Heule. Computing Small Unit-Distance Graphs with Chromatic Number 5, 2018. `arXiv:1805.12181`.
6. V. Jelínek, J. Kyncl, R. Stolar, and T. Valla. Monochromatic triangles in two-colored plane. *Combinatorica*, 29:699–718, 2009.
7. D. Perz. Triangles in the colored Euclidean plane.
8. D. Pritikin. All Unit-Distance Graphs of Order 6197 Are 6-Colorable. *Journal of Combinatorial Theory, Series B*, 73:159–163, 1998.
9. L. Shader. All right triangles are Ramsey in E^2. *Journal of Combinatorial Theory, Series A*, 20:385–389, 1976.
10. A. Soifer. *The Mathematical Coloring Book*. Springer, 2009.

A Wavefront-Like Strategy for Computing Multiplicatively Weighted Voronoi Diagrams

M. Held[1] and S. de Lorenzo[2]

1 **University of Salzburg**
`held@cs.sbg.ac.at`
2 **University of Salzburg**
`slorenzo@cs.sbg.ac.at`

Abstract

We study multiplicatively weighted Voronoi diagrams (MWVDs) of point sites in the Euclidean plane and present a wavefront-like approach for computing the MWVD of n points in near-optimal $\mathcal{O}(n^2 \log n)$ time and $\Theta(n^2)$ space. The key advantage of our algorithm is its simplicity. Furthermore, it can be extended to handle additive weights at no additional computational cost.

1 Introduction

Let S denote a finite set of n distinct weighted points in $\mathbb{R}^2$, so-called *sites*. A weight function $w\colon S \to \mathbb{R}^+$ assigns a strictly positive weight $w(s)$ to every site $s \in S$. It is common to regard the weighted Euclidean distance $d_w(p, s)$ from an arbitrary point p in $\mathbb{R}^2$ to a site $s \in S$ as the standard Euclidean distance $d(p, s)$ from p to s divided by the weight of s:

$$d_w(p, s) := \frac{1}{w(s)} \cdot d(p, s).$$

For $s \in S$, the *(weighted) Voronoi region* $\mathcal{VR}_w(s, S)$ of s relative to S is the set of all points of the plane that are not farther to s than to any other site s' in S, that is

$$\mathcal{VR}_w(s, S) := \{p \in \mathbb{R}^2 : d_w(p, s) \leq d_w(p, s') \text{ for all } s' \in S \text{ with } s \neq s'\}.$$

The multiplicatively weighted Voronoi diagram (MWVD), $\mathcal{VD}_w(S)$, of S is simply defined as

$$\mathcal{VD}_w(S) := \bigcup_{s \in S} \partial\, \mathcal{VR}_w(s, S).$$

An example of a MWVD is shown in Figure 1.

A connected component of a Voronoi region is called a *face*. Voronoi region. For two distinct sites s, s' of S, the *bisector* $b(s, s')$ of s, s' models the set of points of the plane that are at the same weighted distance from s and s'. Hence, a non-empty intersection of two Voronoi regions is a subset of the bisector of the two defining sites. Following common terminology, a connected component of such a set is called a *(Voronoi) edge* of $\mathcal{VD}_w(S)$. An endpoint of an edge is called a *(Voronoi) node*. It is known that the bisector between two unequally weighted sites forms a circle[1]. Hence, the Voronoi edges of $\mathcal{VD}_w(S)$ are given by straight-line segments and circular arcs. In contrast to the standard Voronoi diagram, a MWVD may include a quadratic number of Voronoi nodes, edges, and faces [2].

In the sequel we present work in progress on computing MWVDs. Our current algorithm operates entirely in the plane and runs in $\mathcal{O}(n^2 \log n)$ time and $\Theta(n^2)$ space. It is based on a

[1] Apollonius of Perga defined a circle as a set of points that have a specific distance ratio to two foci.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

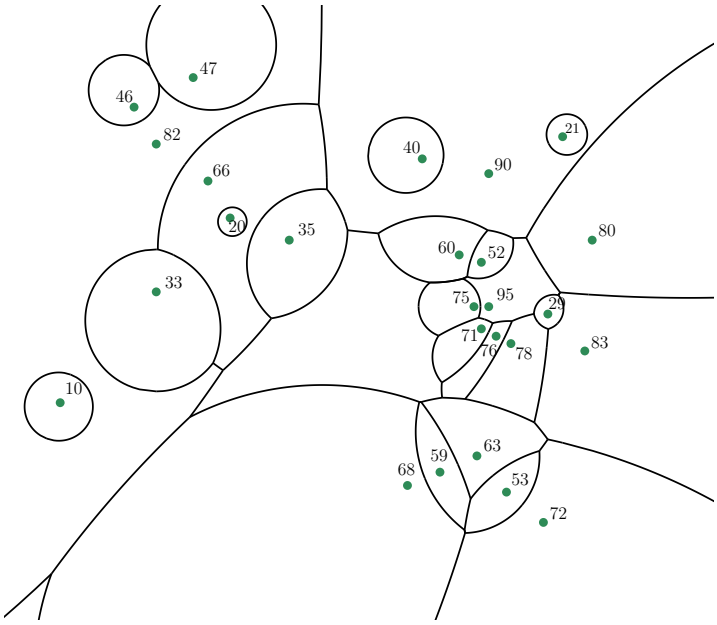


Figure 1 The MWVD (shown in black) of a set of input points (in green). The numbers next to the points indicate their weights.

wavefront-like expansion of weighted circles. As time progresses the wavefront $\mathcal{WF}(t)$ covers a growing portion of $\mathbb{R}^2$. Certain events mark topological changes of $\mathcal{WF}(t)$ and aid us in finding the individual Voronoi nodes. Although it is no wavefront propagation in the strict meaning of the word, we will (for the sake of simplicity) omit the qualifier "like" from here on. Our approach can be extended to handle both additive and multiplicative weights at no additional cost. (Details are omitted due to lack of space.) We developed a prototype implementation of our algorithm in C++ which uses standard IEEE 754 floating-point arithmetic.

2 Related Work

MWVDs were initially studied by Boots [3] in terms of market area analysis. In 1984, Aurenhammer and Edelsbrunner [2] presented an optimal algorithm for constructing the MWVD of a set of n points in $\mathbb{R}^2$ in $\mathcal{O}(n^2)$ time and space. They define spheres on the bisector circles and convert them into half-planes using a spherical inversion. We are not aware of an implementation of their algorithm, though. Later Aurenhammer uses divide&conquer to obtain an $\mathcal{O}(n \log n)$ time and $\mathcal{O}(n)$ space algorithm for the one-dimensional weighted Voronoi diagram [1], where all weighted input points lie on a line. Har-Peled and Raichel [4] show that MWVDs have a slightly super-linear expected combinatorial complexity if the weights are chosen randomly. Their result provides the motivation for working on an algorithm whose running time is output-sensitive.

Vyatkina and Barequet [5] present a wavefront-like strategy to compute the weighted Voronoi diagram of a set of lines in the plane in $\mathcal{O}(n^2 \log n)$ time. The Voronoi nodes are computed based on edge and break-through events. An edge event takes place when an active arc vanishes. A break-through event happens whenever a new wavefront arc appears.

3 Algorithm

For the sake of descriptional simplicity we assume that no point in $\mathbb{R}^2$ has the same weighted distance to more than three input sites of S. For time $t \in \mathbb{R}_0^+$, each site $s \in S$ is associated with an expanding *offset circle* $o(s,t)$ which is centered at s and whose radius equals $t \cdot w(s)$. We find it convenient to regard $o(s,t)$ as a function of either time or distance since at time t every point on $o(s,t)$ is at Euclidean distance $t \cdot w(s)$ from s, i.e., at weighted distance t.

The wavefront $\mathcal{WF}(t)$ at time t is the set of all points p of the plane whose weighted distance from S equals t: We have $p \in \mathcal{WF}(t)$ if and only if $\min_{s \in S} d_w(p,s) = t$. The wavefront is formed by parts of offset circles which we will refer to as *wavefront arcs*. Every wavefront arc starts and ends at a moving *wavefront vertex*, i.e., a specific intersection point with another offset circle. These vertices will trace out the MWVD, see Figure 2.

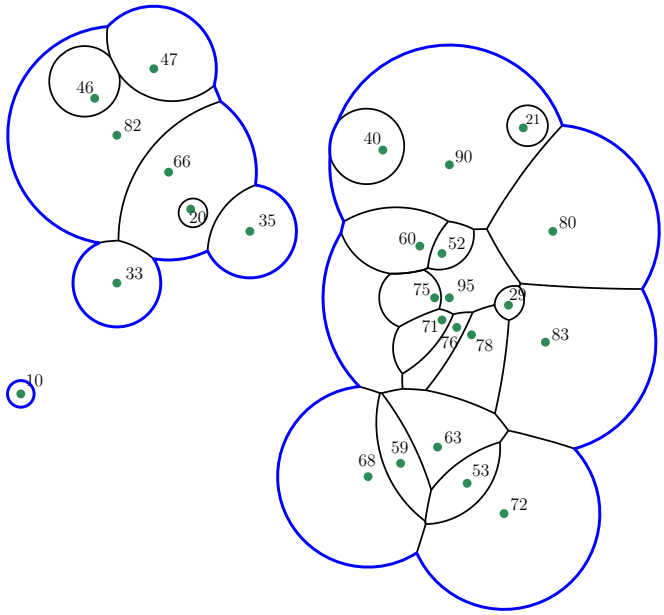


Figure 2 A wavefront $\mathcal{WF}(t)$ (in blue) and the MWVD traced out till some time t for the setting of Figure 1.

We now consider two sites $s_1, s_2 \in S$, with $w(s_1) < w(s_2)$, and assume that $o(s_1,t)$ and $o(s_2,t)$ intersect in the points i_1 and i_2. These two *moving intersection points* trace out $b(s_1,s_2)$. (Of course, the moving intersection point i_1 depends on time t but we simply write i_1 instead of $i_1(t)$.) Since $w(s_1) < w(s_2)$, it is easy to see that the arc of $o(s_1,t)$ which is inside $o(s_2,t)$ will not belong to $\mathcal{WF}(t')$ for any $t' > t$. We refer to such an arc of an offset circle as *inactive*. All other arcs in the arrangement of all offset circles are called *active*, see Figure 3. Thus, it is necessary (but not sufficient) for an arc of an offset circle to be active for all times t' with $0 \leq t' < t$ if it is active at time t.

We now describe an event-handling scheme that allows to trace out the Voronoi diagram by simulating the expansion of all active arcs. During the wavefront propagation process, *collision events* mark the initial contact of two offset circles, *domination events* happen as soon as the offset circle of a higher-weighted site fully contains the offset circle of a lower-weighted one, and *edge events* as well as *break-through events* take place whenever an active arc vanishes or appears. These events capture topological changes of the wavefront and determine the corresponding Voronoi nodes of $\mathcal{VD}_w(S)$. The latter two events are triggered

Figure 3 The arrangement of all active arcs that corresponds to the wavefront depicted in Figure 2.

whenever one or more active arcs vanish, i.e., shrink to zero length.

Every site s keeps track of the active arcs of its expanding offset circle $o(s,t)$ by storing them in a self-balancing binary search tree $\mathcal{T}(s)$ that is updated whenever events occur. It maintains these arcs in sorted angular order as they appear when $o(s,t)$ is traversed counter-clockwise.

We start with computing the collision times for every pair of offset circles and insert them into a priority queue $\mathcal{Q}$. The initial wavefront $\mathcal{WF}(t)$, for $t > 0$ small enough, contains the full offset circles of all sites, and every offset circle forms one active arc. Every active arc is also marked to belong to $\mathcal{WF}(t)$.

As soon as the initialization phase is completed, the events are successively popped from $\mathcal{Q}$. Let e be the current event at time t_e. Note that the number and order of the active arcs along a specific offset circle cannot change between two consecutive events, but their extents, i.e., the portions of the offset circle which they occupy, may change. Hence it is important to recompute the positions of the moving intersection points on the fly whenever we perform a search in one of our search trees at time t_e.

Collision event: If e is a collision event then two offset circles $o(s_1,t_e)$ and $o(s_2,t_e)$ meet at a single point q for the first time; see Figure 4. We search $\mathcal{T}(s_1)$ and $\mathcal{T}(s_2)$, and determine the arcs a_1 and a_2 which contain q. If either of them is inactive then this event requires no further processing. Otherwise we create two moving intersections i'_{12} and i''_{12} at q and we split both a_1 and a_2 at q. W.l.o.g., $w(s_1) < w(s_2)$. As time progresses, i'_{12} and i''_{12} limit a new active arc on $o(s_2,t)$ and an inactive arc on $o(s_1,t)$. (If $w(s_1) = w(s_2)$ then both new arcs are inactive.)

Domination Event: If e is a domination event then the offset circle $o(s_1,t_e)$ fully contains the offset circle $o(s_3,t_e)$ for the first time at time t_e, with $w(s_3) < w(s_1)$; see Figure 5. That

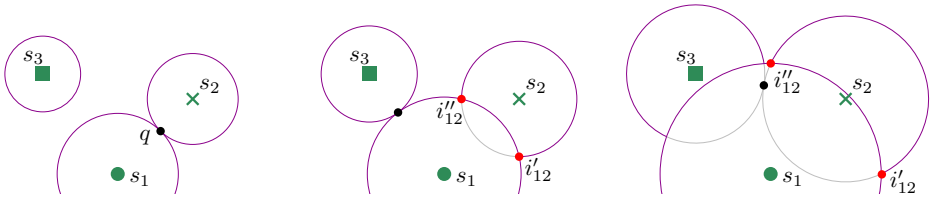


Figure 4 The collision events (depicted by black dots) of the offset circles that correspond to the three input sites s_1, s_2, and s_3, where $w(s_1) := 5$, $w(s_2) := 6$, and $w(s_3) := 8$. The active arcs of the offset circles are drawn in magenta whereas the inactive arcs are shown in gray. The red dots indicate the moving intersection points i'_{12} and i''_{12}.

is, the two offset circles touch at a point q. Let a_1 and a_3 be the corresponding arcs that contain q. If one of them is inactive then we ignore this event. Otherwise, all arcs of $o(s_3, t_e)$ become inactive and a_1 is merged with its neighboring arcs.

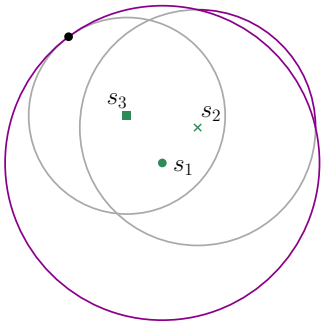


Figure 5 A domination event that involves s_1 and s_3 takes place.

Edge Event: An edge event occurs at time t_e if at least one active arc along an offset circle shrinks to a point q, i.e., to zero length; see Figure 6. If an entire circular-arc triangle shrinks to q at time t_e then all active arcs involved can be removed from their corresponding offset circles. Otherwise, a single active arc a_1 on one of the two higher weighted offset circles $o(s_1, t_e)$ just disappeared. The two other sites s_2 and s_3, where s_3 is w.l.o.g. the site with the lowest weight, whose offset circles cause a_1 to vanish can be derived from the moving intersection points i_{12} and i_{13} that limit a_1. We remove a_1 from $o(s_1, t_e)$ and add a new active arc which is bound by i_{12} and i_{23} to $o(s_2, t_e)$. Additionally, the moving intersection point i_{13} that bounds an active arcs along $o(s_3, t_e)$ needs to be replaced by i_{23}.

Break-Through Event: A break-through event is a special kind of edge event and can be treated similarly. It also occurs if an active arc a_1 arc on $o(s_2, t_e)$ shrinks to a point q. Let, again, s_1 and s_3 be the other two sites that participate in this event, where s_3 is associated with the lowest weight; see Figure 7. The arc a_1 is deleted from $o(s_2, t_e)$ and a new active arc a'_1 spawns between the corresponding moving intersection points i_{12} and i_{13} on $o(s_1, t_e)$, and i_{23} that bounds a neighboring active arc along $o(s_3, t_e)$ is replaced by i_{13}.

Common to all these events is the necessity to compute and store a future edge event for every newly created active arc. Furthermore, every inactive arc is deleted from its search data structure, while every new active arc is inserted into the search data structure of its site.

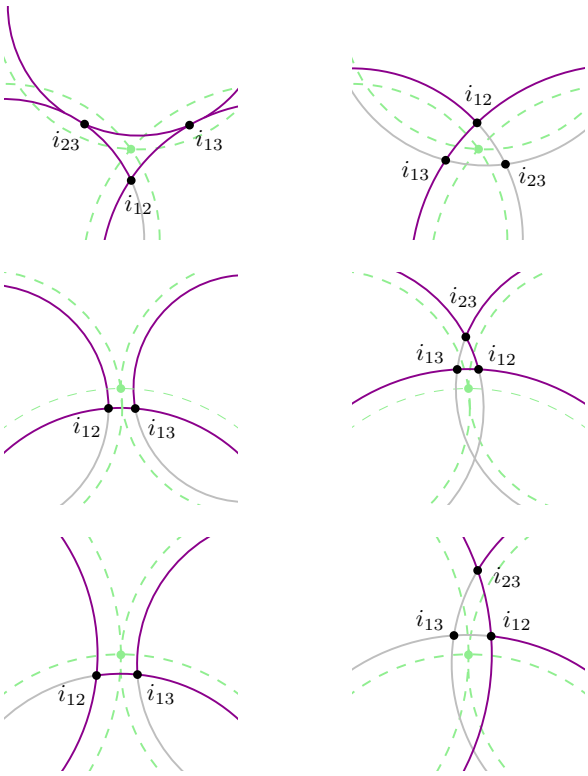


Figure 6 In the top-most subfigure the configuration before and after the collapse of an entire circular-arc triangle is displayed. The remaining subfigures illustrate the collapse of a single active arc on the offset circle of the highest-weighted (medium-weighted, resp.) site.

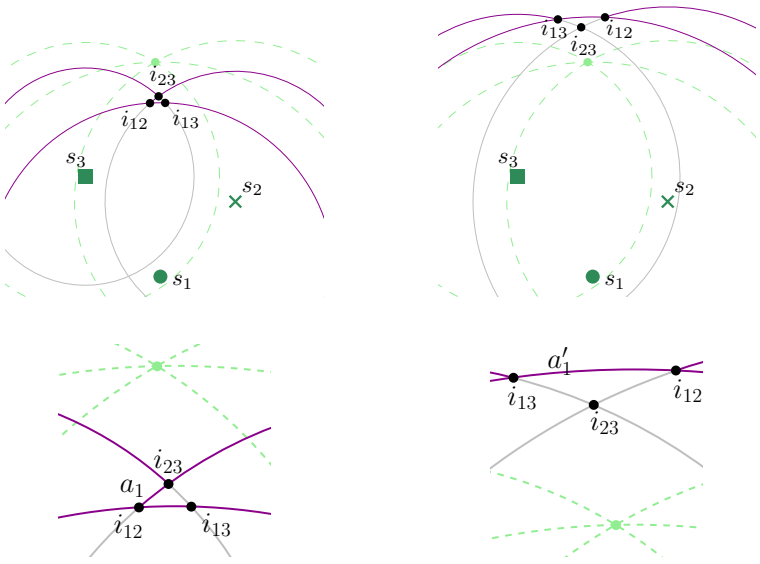


Figure 7 An active arc a_1 disappears and a new active wavefront arc a_1' spawns between the moving intersection points i_{12} and i_{13} at a break-through event.

We process events until the priority queue $\mathcal{Q}$ is empty. If the largest weight is associated with a strict subset $S_{\max}$ of S then $\mathcal{Q}$ will be empty once the wavefront contains only arcs of offset circles of sites of $S_{\max}$. If all sites have identical weight then $\mathcal{VD}_w(S)$ equals the standard Voronoi diagram and $\mathcal{Q}$ will be empty once the wavefront contains only arcs of offset circles of sites which lie on the convex hull of S.

4 Analysis

All topological changes of the wavefront are properly detected by our algorithm, as they coincide with the collapse of at least one active arc of the wavefront. It remains to determine an upper bound on the number of events that may take place during the wavefront propagation.

▶ **Lemma 4.1.** *During the wavefront propagation $\Theta(n^2)$ many collision and $\mathcal{O}(n^2)$ many domination events are computed.*

Proof. Recall that every pair of input sites corresponds to at most one collision and at most one domination event, with all collision events being computed a priori. ◀

▶ **Lemma 4.2.** *During the wavefront propagation $\mathcal{O}(n^2)$ many break-through events occur.*

Proof. Let s_1, s_2 and s_3 be the sites whose expanding offset circles $o(s_1, t)$, $o(s_2, t)$ and $o(s_3, t)$ are involved in a break-through event at time t_e. These offset circles define the three moving intersection points i_{12}, i_{13} and i_{23}, where i_{23} is shared by active arcs of $o(s_2, t)$ and $o(s_3, t)$ for $t < t_e$; recall Figure 7. At the time of the event, the offset circle $o(s_1, t_e)$ passes through i_{23} and this moving intersection point will be contained inside of $o(s_1, t)$ for all $t > t_e$. Hence, for $t > t_e$, no active arc of $o(s_2, t)$ can share a common vertex with an active arc of $o(s_3, t)$. (Note that otherwise the MWVD of $\{s_1, s_2, s_3\}$ would potentially include multiply-connected Voronoi regions.) This implies that a break-through event can occur at most once for each pair of input sites. ◀

▶ **Lemma 4.3.** *During the wavefront propagation $\mathcal{O}(n^2)$ many edge events occur.*

Proof. At every collision and domination event a constant number of new active arcs is generated, and every break-through event results in exactly one new active arc, resulting in a total of $\mathcal{O}(n^2)$ new active arcs during the entire run of the wavefront propagation. Every edge event either reduces the number of active arcs by at least one or, if this number stays constant, then it is coupled to exactly one of the at most quadratically many break-through events, recall the right part of Figure 6. ◀

Summarizing, $\Theta(n^2)$ events take place during the wavefront propagation. Each of these events consumes up to $\mathcal{O}(\log n)$ time, since every event requires a constant number of lookups, insertions, and/or deletions in a self-balancing binary search tree of size $\mathcal{O}(n)$ or in a priority queue of size $\Theta(n^2)$. Thus, we get an overall runtime of $\mathcal{O}(n^2 \log n)$. Additionally, this algorithm requires $\Theta(n^2)$ memory because our current approach computes all quadratically many collision events a priori. Avoiding this computational burden is work in progress.

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

References

1 Franz Aurenhammer. The One-Dimensional Weighted Voronoi Diagram. *Information Processing Letters*, 22(3):119–123, 1986. doi:10.1016/0020-0190(86)90055-4.

2 Franz Aurenhammer and Herbert Edelsbrunner. An Optimal Algorithm for Constructing the Weighted Voronoi Diagram in the Plane. *Pattern Recognition*, 17(2):251–257, 1984. doi:10.1016/0031-3203(84)90064-5.

3 Barry N Boots. Weighting Thiessen Polygons. *Economic Geography*, 56(3):248–259, 1980. doi:10.2307/142716.

4 Sariel Har-Peled and Benjamin Raichel. On the Complexity of Randomly Weighted Multiplicative Voronoi Diagrams. *Discrete & Computational Geometry*, 53(3):547–568, 2015. doi:10.1007/s00454-015-9675-0.

5 Kira Vyatkina and Gill Barequet. On Multiplicatively Weighted Voronoi Diagrams for Lines in the Plane. *Trans. Computational Science*, 13:44–71, 2011. doi:10.1007/978-3-642-22619-9_3.

Linear-size farthest color Voronoi diagrams: conditions and algorithms*

Ioannis Mantas[1], Evanthia Papadopoulou[1], Vera Sacristán[2], and Rodrigo I. Silveira[2]

1 **Faculty of Informatics, Università della Svizzera italiana, Lugano, Switzerland**
{ioannis.mantas,evanthia.papadopoulou}@usi.ch
2 **Departament de Matemàtiques, Universitat Politècnica de Catalunya, Barcelona, Spain**
{vera.sacristan,rodrigo.silveira}@upc.edu

Abstract

The *farthest-color Voronoi diagram* (FCVD) is a *farthest-site* Voronoi diagram defined on a family of m clusters (sets) of points in the plane. Its combinatorial complexity in the worst case is $\Theta(mn)$, where n is the total number of points. In this paper we give structural properties of the FCVD and list sufficient conditions under which this diagram has $O(n)$ combinatorial complexity. For such cases we present efficient construction algorithms.

1 Introduction

The *Voronoi diagram* is a well-known geometric partitioning structure, defined by a set of simple geometric objects in a space, called sites. The ordinary *(nearest-neighbor) Voronoi diagram* of a set of points in two dimensions is a subdivision of the plane into maximal regions such that all points in one region share the same nearest site. In the *farthest-site Voronoi diagram*, points in a single region have the same farthest site. Many generalizations of this simple concept have been considered for different types of sites, metrics and spaces. For a comprehensive list of results see [2].

We are interested in *color Voronoi diagrams*, where each site is a *cluster* (a set) of points in $\mathbb{R}^2$, identified by a distinct color. The distance between a point $x \in \mathbb{R}^2$ and a cluster P is realized by the nearest point in P, i.e., $d_c(x, P) = \min_{p \in P} d(x, p)$. The *nearest-color Voronoi diagram (NCVD)* of a family $\mathcal{P}$ of clusters, is a *min-min* diagram that can be easily derived from the ordinary Voronoi diagram of all points in $\mathcal{P}$: the region of a cluster P is the union of the Voronoi regions of points belonging to P (see Fig. 1a). Its farthest counterpart, the *farthest-color Voronoi diagram (FCVD)* of $\mathcal{P}$ is a *max-min* diagram and its properties have not been extensively looked into (see Fig. 1b).

The FCVD was first studied by Huttenlocher et al. [9], showing that the combinatorial complexity of the diagram in the worst case is $\Omega(mn)$ and $O(mn\alpha(mn))$, where m is the number of clusters and n is the overall number of points. This was later settled to $\Theta(mn)$ by Abellanas et al. [1]. Using a geometric transformation in 3D, the diagram can be computed in $O(mn \log n)$ time [9]: for every cluster P, each point in the plane is lifted in 3 dimensions, with height equal to the distance from the nearest point in P, yielding a surface; the upper

* This research and in particular the work of I. M. and E. P. has been supported in part by the Swiss National Science Foundation, under the DACH project VORONOI++, SNF 200021E-154387. R. S. was supported by MINECO through the Ramón y Cajal program. V. S. 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.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 19–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

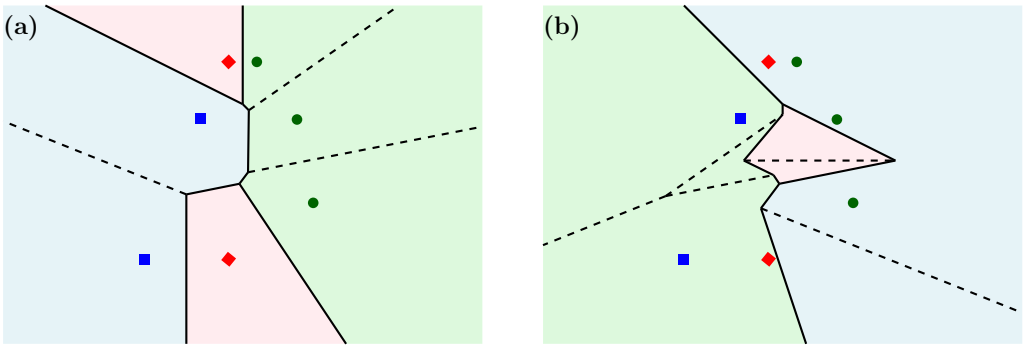


Figure 1 A family $\mathcal{P}$ of clusters along with (a) NCVD($\mathcal{P}$) and (b) FCVD($\mathcal{P}$).

envelope of these surfaces projected back onto the plane gives the FCVD. Instances of linear-size diagrams have been considered by Bae [3], Claverol et al. [6] and Iacono et al. [10]. Applications of the FCVD include facility location problems [1], variants of the *Steiner tree* problem [4], sensor deployment problems [13] and finding *stabbing circles* for line segments [6].

Closely related to the FCVD is the *Hausdorff Voronoi diagram* (HVD) of a family of point clusters. The HVD is a *min-max* diagram: the distance from a point $x \in \mathbb{R}^2$ to a cluster is the farthest distance, $d_f = \max_{p \in P} d(x, p)$, and the plane is subdivided into maximal regions with the same nearest cluster. The HVD has been extensively studied, see e.g. [8, 15], and many algorithmic paradigms have been considered for its construction, see e.g. [7, 11, 15, 16]. Interestingly, the algorithm presented in [8] can be adapted to also yield an $O(n^2)$-time algorithm for the FCVD. This has been remarked in [6] for point clusters of cardinality two. In the worst case, this is optimal as the diagram may have complexity $\Theta(n^2)$. However, the algorithm remains $\Theta(n^2)$ even if the diagram has only $O(n)$ structural complexity.

In this work, we study structural properties of the FCVD, give sufficient conditions under which the diagram has $O(n)$ structural complexity and present efficient algorithms to construct it when these conditions are met.

2 Definitions and basic properties

Let $\mathcal{P} := \{P_1, ..., P_m\}$ be a family of m *clusters of points* in $\mathbb{R}^2$, where no two clusters share a point. We assume that $m > 1$ and let $\sum_{i=1...m} |P_i| = n$. We define the following diagrams.

▶ **Definition 1.** The *nearest color Voronoi diagram (NCVD)* of $\mathcal{P}$ is the subdivision of $\mathbb{R}^2$ into *nearest color Voronoi regions*. The *nearest color Voronoi region* of a cluster $P_i \in \mathcal{P}$ is $n_c reg(P_i) = \{x \in \mathbb{R}^2 \,|\, d_c(x, P_i) < d_c(x, P_j)\ \forall P_j \in \mathcal{P},\ j \neq i\}$.

▶ **Definition 2.** The *farthest color Voronoi diagram (FCVD)* of $\mathcal{P}$ is the subdivision of $\mathbb{R}^2$ into *farthest color Voronoi regions*. The *farthest color Voronoi region* of a cluster $P_i \in \mathcal{P}$ is $f_c reg(P_i) = \{x \in \mathbb{R}^2 \,|\, d_c(x, P_i) > d_c(x, P_j)\ \forall P_j \in \mathcal{P},\ j \neq i\}$.

A region $f_c reg(P_i)$ may consist of several maximally connected components, called *faces*. Faces of $f_c reg(P_i)$ are further subdivided by the ordinary Voronoi diagram of P_i, which is denoted Vor(P_i). This is called the *internal subdivision* of a face. For $p \in P_i : f_c reg(p) = \{x \in f_c reg(P_i) \,|\, d(x, p) < d(x, q)\ \forall q \in P_i \setminus \{p\}\}$. A region $f_c reg(p)$ may have several faces.

▶ **Definition 3.** Given two clusters P_i, P_j, their *color bisector* is the locus of points equidistant from the two clusters, that is, $b_c(P_i, P_j) = \{x \in \mathbb{R}^2 \,|\, d_c(x, P_i) = d_c(x, P_j)\}$.

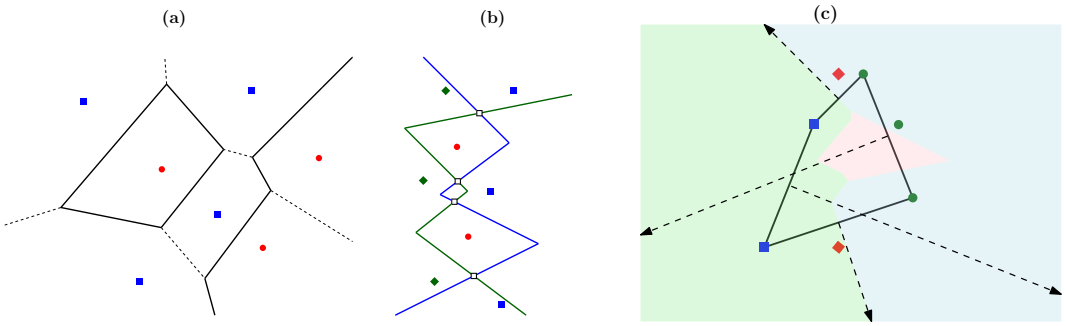


Figure 2 (a) A bisector consisting of a cycle and a chain. (b) Two bisectors sharing a site intersecting linearly many times. (c) Hull of the clusters in Fig. 1 and the associated normal vectors.

Bisector $b_c(P_i, P_j)$ is a subgraph of the Voronoi diagram $\mathrm{Vor}(P_i \cup P_j)$. It is a collection of edge-disjoint cycles and unbounded chains of total complexity $O(|P_i| + |P_j|)$, which is tight in the worst case (see Fig. 2a).

We refer to edges of the FCVD belonging to color bisectors as *pure edges*, and to edges or vertices of the internal subdivisions as *internal*. Voronoi vertices incident to three color bisectors are called *pure vertices*, and vertices incident to two color bisectors and one internal edge are called *mixed vertices*. See Fig. 3 for an illustration of these features.

The following lemma characterizes the structure of farthest color regions.

▶ **Lemma 2.1.** *A face f of $f_c reg(P_i)$ satisfies:*

1. *If f is bounded, its internal subdivision is a tree whose leaves are mixed vertices on ∂f.*
2. *If f is unbounded, its internal subdivision is a (possibly empty) forest, where each tree has exactly one unbounded edge and its remaining leaves are mixed vertices on ∂f.*

The boundary of a face $f_c reg(p), p \in P_i$, is a sequence of convex chains (as seen from p).

We use a refinement of the FCVD derived by the *visibility decomposition*, defined analogously to [16]: For each region $f_c reg(p)$ and for each pure or mixed vertex u on $\partial f_c reg(p)$, draw the portion of the line through p and u that lies inside $f_c reg(p)$ (see Fig. 3).

The *cluster hull*, for short *hull*, of a family of point clusters is a closed (not necessarily simple) polygonal chain that characterizes the unbounded faces of the FCVD and the HVD. We review the definition from [16], see Fig. 2c.

▶ **Definition 4.** Given a family of clusters $\mathcal{P}$, a point $p \in P_i$ is a *hull vertex* if p admits a supporting line l, such that P_i lies completely on one of the two halfplanes defined by l and the other one intersects every cluster $P_j \in \mathcal{P} \setminus \{P_i\}$. A *hull edge* is a segment connecting two hull vertices $p \in P_i, q \in P_j$ such that the line through p, q leaves P_i and P_j on one halfplane, while the other halfplane intersects all other clusters in $\mathcal{P}$. Such an edge is associated with a normal vector in the direction of the halfplane that does not include P_i, P_j. The hull edges sorted by the circular ordering of all such normal vectors define a closed polygonal chain called the *cluster hull* of $\mathcal{P}$, denoted $CLH(\mathcal{P})$.

We show that there is a one-to-one correspondence between the unbounded faces of the FCVD and the HVD. Therefore, results for hulls [16] directly follow.

▶ **Lemma 2.2.** *A region $f_c reg(p)$ is unbounded if and only if p is a vertex of $CLH(\mathcal{P})$. The circular order of hull edges along $CLH(\mathcal{P})$ is equal to that of unbounded edges of* $\mathrm{FCVD}(\mathcal{P})$.

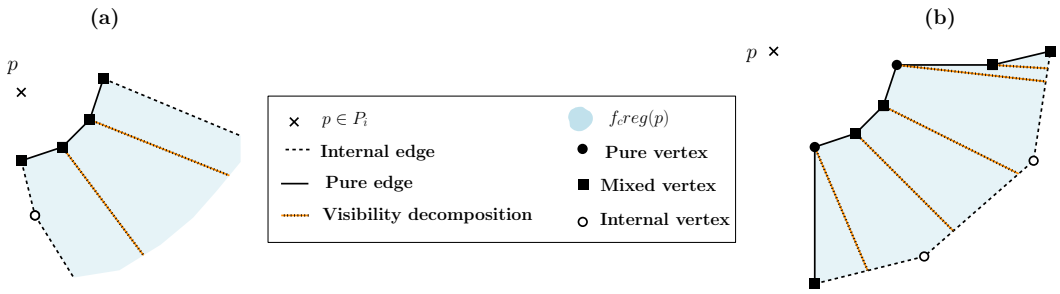


Figure 3 (a) An unbounded and (b) a bounded face of a point $p \in P_i$.

3 Conditions for linear-size diagrams

Abstract Voronoi diagrams were introduced by Klein [12]. Instead of sites and distance measures, these diagrams are defined in terms of bisecting curves satisfying a set of simple combinatorial properties, called axioms. In the context of color Voronoi diagram, these axioms can be stated as follows, for every subset $\mathcal{P}' \subseteq \mathcal{P}$:

(A1) Each region in NCVD($\mathcal{P}'$) is non-empty and path-wise connected.
(A2) Each point in the plane belongs to the closure of a region in NCVD($\mathcal{P}'$).
(A3) Each color bisector is an unbounded Jordan curve.
(A4) Any two color bisectors intersect transversally and in a finite number of points.

A family of clusters is called *admissible* if the system of bisectors satisfies (A1)-(A4). By the structural properties of farthest abstract Voronoi diagrams [5, 14] we derive the following.

▶ **Lemma 3.1.** *If $\mathcal{P}$ is admissible, then the skeleton of* FCVD($\mathcal{P}$) *is a tree of $O(n)$ total structural complexity. One region may consist of $\Theta(m)$ disjoint faces and the total number of faces is $O(m)$.*

Two clusters are called *linearly-separable* if they have disjoint convex hulls. A family of pairwise linearly-separable clusters is also called *linearly-separable*. The color bisector of two linearly-separable clusters is a single unbounded, monotone chain. The color bisectors of three pairwise linearly-separable clusters, however, $b_c(P_i, P_j)$ and $b_c(P_j, P_k)$ may intersect $\Theta(|P_i| + |P_j| + |P_k|)$ times (see Fig. 2b). Thus, a linearly separable family need not be admissible. By showing that if the regions of NCVD($\mathcal{P}$) are connected then the same should hold for NCVD($\mathcal{P}'$), for any $\mathcal{P}' \subseteq \mathcal{P}$, we derive the following.

▶ **Lemma 3.2.** *Let $\mathcal{P}$ be a linearly-separable family of clusters. If the regions in* NCVD($\mathcal{P}$) *are path-connected, then $\mathcal{P}$ is admissible.*

Lemma 3.2 indicates that we can determine if a family $\mathcal{P}$ is admissible in $O(n \log n)$ time. A family of clusters $\mathcal{P}$ is called *disk-separable* if for every cluster $P_i \in \mathcal{P}$ there exists a disk containing P_i and no point from other clusters (see Fig. 4). By proving that disk separability implies connected regions in NCVD($\mathcal{P}$), we derive:

▶ **Lemma 3.3.** *Any family of disk-separable clusters $\mathcal{P}$ is admissible.*

We now look into linearly-separable families of clusters. The following statement has been proven for clusters of cardinality two [6] but holds also for general clusters.

▶ **Lemma 3.4.** *If $\mathcal{P}$ is linearly-separable, then* FCVD($\mathcal{P}$) *has $O(m)$ unbounded faces.*

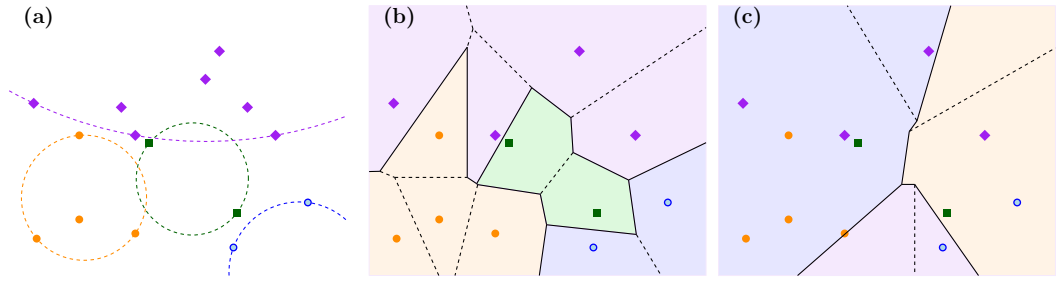


Figure 4 (a) A disk-separable family of clusters $\mathcal{P}$ along with (b) NCVD($\mathcal{P}$) and (c) FCVD($\mathcal{P}$).

A pair of points $(p_1, p_2) \in P_i$, which defines a Voronoi edge e in $\text{Vor}(P_i)$, is said to be *straddled* by a cluster $Q_j \in \mathcal{P}$ if the line through (p_1, p_2) intersects the segment $\overline{q_1 q_2}$ defined by $(q_1, q_2) \in Q_j$ and the circles through (q_1, p_1, p_2) and (q_2, p_1, p_2) are both centered on e (see Fig. 5a). We also say that (q_1, q_2) and Q_j straddle the Voronoi edge e.

We define the *straddling number* of e, denoted $s(e)$, as the number of clusters in $\mathcal{P}$ that straddle e. Clearly, for a cluster P_i, $s(P_i) = O(m|P_i|)$. The *straddling number of family* $\mathcal{P}$, is $s(\mathcal{P}) = \sum_{P_i \in \mathcal{P}} s(P_i)$. In the worst case, $s(\mathcal{P}) = \Theta(mn)$.

▶ **Lemma 3.5.** *If $\mathcal{P}$ is linearly-separable, then the number of bounded faces, and the overall structural complexity of* FCVD($\mathcal{P}$)*, is $O(n + s(\mathcal{P}))$.*

Proof. *(sketch)* For each Voronoi edge e of $\text{Vor}(P_i)$ we allow one bounded face of $f_c reg(P_i)$ and count the number of mixed vertices that may be incident to additional faces of $f_c reg(P_i)$ on e. Let v_1, v_2 be two consecutive mixed vertices on a Voronoi edge e of $\text{Vor}(P_i)$, induced by points (p_1, p_2), such that segment $\overline{v_1 v_2} \notin f_c reg(P_i)$ (see Fig.5). Suppose v_1 is induced by $q_1 \in Q_j$. By considering a disk moving from left to right on e and touching (p_1, p_2), we can show that v_2 must be induced by a point $q_2 \in Q_j$ such that (q_1, q_2) defines a straddle on e. In addition, cluster Q_j cannot induce any other mixed vertex on e. Thus, the pair of vertices (v_1, v_2) is charged to a unique cluster counted in the straddling number of e. ◀

By Lemma 3.5, if the straddling number $s(\mathcal{P})$ is $O(n)$, then FCVD($\mathcal{P}$) has complexity $O(n)$.

4 Construction algorithms

Consider a divide & conquer approach. Split $\mathcal{P}$ into $\mathcal{P}_L$ and $\mathcal{P}_R$ by a vertical line; Compute FCVD($\mathcal{P}_L$) and FCVD($\mathcal{P}_R$) recursively; Merge FCVD($\mathcal{P}_L$) and FCVD($\mathcal{P}_R$) to obtain FCVD($\mathcal{P}$). Merging requires constructing the *merge curve* $\mathcal{M}(\mathcal{P}_L \cup \mathcal{P}_R)$, which is the set of pure edges of FCVD($\mathcal{P}_L \cup \mathcal{P}_R$) belonging to bisectors $b_c(P_i, P_j)$ with $P_i \in \mathcal{P}_L$ and $P_j \in \mathcal{P}_R$. A merge curve may consist of linearly many chains, called *components*. To construct it, a *starting point* has to be found on each component and then the chain has to be *traced*.

Given a starting point on a component we can efficiently trace it, by adapting standard tracing methods and exploiting the visibility decomposition, similarly to [16].

▶ **Lemma 4.1.** *Given diagrams* FCVD($\mathcal{P}_L$), FCVD($\mathcal{P}_R$) *and a starting point on a component M of $\mathcal{M}(\mathcal{P}_A, \mathcal{P}_B)$, the component M can be computed in $O(|M|)$ time.*

Due to Lemma 2.2, we can identify starting points on the unbounded components of $\mathcal{M}(\mathcal{P}_A, \mathcal{P}_B)$ by merging $CLH(\mathcal{P}_L)$ and $CLH(\mathcal{P}_R)$, before merging the two diagrams. This can be done in time $O(|CLH(\mathcal{P}_L)| + |CLH(\mathcal{P}_R)|)$, see [16].

If $\mathcal{P}$ is admissible, (such as a family of disk separable clusters), then all regions of

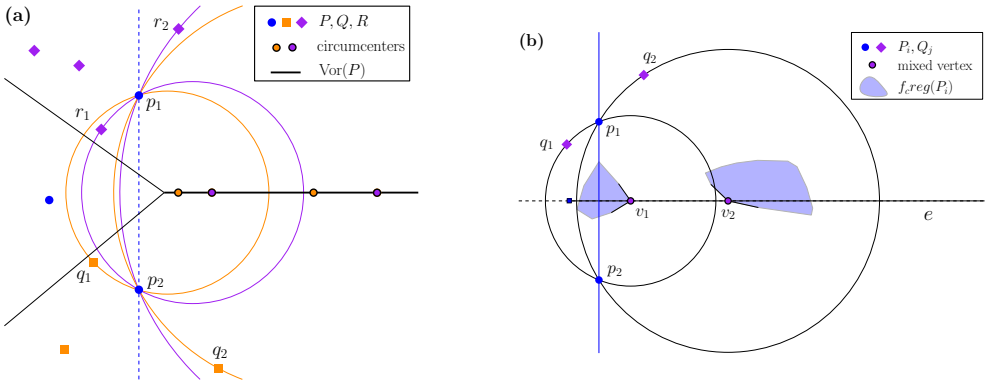


Figure 5 (a) A family $\mathcal{P}$, where pair (p_1, p_2) is straddled by two clusters Q, R.(b) Illustration of the proof of Lemma 3.5.

FCVD($\mathcal{P}$) are unbounded (Lemma 3.1) and this is true for all components of the merge curve. Thus, we derive the following.

▶ **Theorem 1.** If $\mathcal{P}$ is admissible, then FCVD($\mathcal{P}$) can be constructed in $O(n \log n)$ time.

Note that for an admissible family $\mathcal{P}$, FCVD($\mathcal{P}$) could also be computed using the randomized algorithm of [14] for abstract Voronoi diagrams. Color bisectors, however, may have $\Theta(n)$ complexity, so, a direct application would give time complexity $O(n^2 \log n)$.

When $\mathcal{P}$ is not admissible, the challenge is to identify starting points on the bounded components of the merge curve. For linearly-separable families where clusters have a constant straddling number, there are constant number of bounded components. To identify starting points on these components, the data structure and technique of [10] can be used to do this in $O(n \log n)$ time, yielding an $O(n \log^2 n)$-time algorithm.

▶ **Theorem 2.** If $\mathcal{P}$ is a linearly-separable family of clusters, where $s(P_i)$ is constant for any $P_i \in \mathcal{P}$, then FCVD($\mathcal{P}$) can be constructed in $O(n \log^2 n)$ time.

We conjecture that for linearly-separable families the FCVD can have complexity $\Theta(mn)$.

References

1 Manuel Abellanas, Ferran Hurtado, Christian Icking, Rolf Klein, Elmar Langetepe, Lihong Ma, Belén Palop, and Vera Sacristán. The farthest color Voronoi diagram and related problems. Technical report, Rheinische Friedrich–Wilhelms–Universität Bonn, 2006.

2 Franz Aurenhammer, Rolf Klein, and Der-Tsai Lee. *Voronoi Diagrams and Delaunay Triangulations*. World Scientific, 2013.

3 Sang Won Bae. On linear-sized farthest-color Voronoi diagrams. *IEICE Transactions on Information and Systems*, 95(3):731–736, 2012.

4 Sang-Won Bae, Chun-Seok Lee, and Sung-Hee Choi. On exact solutions to the euclidean bottleneck steiner tree problem. *Information Processing Letters*, 110(16):672–678, 2010.

5 Cecilia Bohler, Panagiotis Cheilaris, Rolf Klein, Chih-Hung Liu, Evanthia Papadopoulou, and Maksym Zavershynskyi. On the complexity of higher order abstract Voronoi diagrams. *Computational Geometry*, 48(8):539 – 551, 2015.

6 Mercè Claverol, Elena Khramtcova, Evanthia Papadopoulou, Maria Saumell, and Carlos Seara. Stabbing circles for sets of segments in the plane. *Algorithmica*, pages 1–36, 2017.

7 Frank Dehne, Anil Maheshwari, and Ryan Taylor. A coarse grained parallel algorithm for hausdorff voronoi diagrams. In *Parallel Processing, 2006. ICPP 2006. International Conference on*, pages 497–504. IEEE, 2006.

8 Herbert Edelsbrunner, Leonidas J Guibas, and Micha Sharir. The upper envelope of piecewise linear functions: algorithms and applications. *Discrete & Computational Geometry*, 4(1):311–336, 1989.

9 Daniel P Huttenlocher, Klara Kedem, and Micha Sharir. The upper envelope of Voronoi surfaces and its applications. *Discrete & Computational Geometry*, 9(3):267–291, 1993.

10 John Iacono, Elena Khramtcova, and Stefan Langerman. Searching edges in the overlap of two plane graphs. *arXiv preprint arXiv:1701.02229*, 2017.

11 Elena Khramtcova and Evanthia Papadopoulou. Randomized incremental construction for the Hausdorff Voronoi diagram revisited and extended. In *International Computing and Combinatorics Conference*, pages 321–332. Springer, 2017.

12 Rolf Klein. *Concrete and abstract Voronoi diagrams*, volume 400. Springer Science & Business Media, 1989.

13 Chunseok Lee, Donghoon Shin, Sang Won Bae, and Sunghee Choi. Best and worst-case coverage problems for arbitrary paths in wireless sensor networks. *Ad Hoc Networks*, 11(6):1699–1714, 2013.

14 Kurt Mehlhorn, Stefan Meiser, and Ronald Rasch. Furthest site abstract Voronoi diagrams. *International Journal of Computational Geometry & Applications*, 11(06):583–616, 2001.

15 Evanthia Papadopoulou. The Hausdorff Voronoi diagram of point clusters in the plane. *Algorithmica*, 40(2):63–82, 2004.

16 Evanthia Papadopoulou and Der-Tsai Lee. The Hausdorff Voronoi diagram of polygonal objects: A divide and conquer approach. *International Journal of Computational Geometry & Applications*, 14(06):421–452, 2004.

Unbounded Regions of Higher-Order Line and Segment Voronoi Diagrams in Higher Dimensions*

Gill Barequet[1], Evanthia Papadopoulou[2], and Martin Suderland[3]

1 **Dept. of Computer Science, The Technion—Israel Inst. of Technology, Haifa 3200003, Israel**
barequet@cs.technion.ac.il
2 **Faculty of Informatics, Università della Svizzera italiana, Lugano, Switzerland**
evanthia.papadopoulou@usi.ch
3 **Faculty of Informatics, Università della Svizzera italiana, Lugano, Switzerland**
martin.suderland@usi.ch

Abstract

We study the behavior at infinity of farthest and higher-order Voronoi diagrams of line segments or/and lines in a d-dimensional Euclidean space. The unbounded parts of these diagrams can be encoded by a *Gaussian map* on the sphere of directions $\mathbb{S}^{d-1}$. We show that the combinatorial complexity of the Gaussian map for the order-k Voronoi diagram of n line segments or/and lines is $O(\min(k, n-k)n^{d-1})$, which is tight for $n-k = O(1)$. The Gaussian map of the farthest Voronoi diagram of line segments or/and lines in $\mathbb{R}^3$ can be constructed in $O(n^2)$ time.

1 Introduction

The Voronoi diagram of a set of n geometric objects, called sites, is a well-known geometric space-partitioning structure. The *nearest* variant partitions the underlying space into maximal regions, such that all points within a region have the same nearest site. A very well-studied type of Voronoi diagram is the Euclidean Voronoi diagram of n points in $\mathbb{R}^d$, see [4, 7, 9].

Many algorithmic paradigms, such as plane sweep, incremental construction, and divide and conquer have been applied to construct the Voronoi diagram of line segments in the plane [2]. Already in a three-dimensional space, the algebraic description of the features, such as the edges, of the Voronoi diagram of line segments become complicated [8].

The order-k (resp., farthest) Voronoi diagram of a set of sites is a partition of the underlying space into regions, such that the points of one region have the same k nearest sites (resp., same farthest site). In two dimensions, the farthest Voronoi diagram of n segments has already been studied by Aurenhammer et al. [1], who give results on its structure and an algorithm to compute it in $O(n \log n)$ time. The order-k counterpart of this diagram has $O(k(n-k))$ complexity and it can be constructed iteratively [11]. Already in a three-dimensional space with the Euclidean metric, no tight asymptotic bound on the complexity of the farthest Voronoi diagram is known, and similarly for the nearest-neighbor diagram [10]. In both cases, the only known bounds are $\Omega(n^2)$ and $O(n^{3+\varepsilon})$, for any $\varepsilon > 0$ [3, 12].

The Euclidean farthest-neighbor Voronoi diagrams of lines and/or line segments in three dimensions has the property that all cells are unbounded [3]. This property motivated us to first study the unbounded parts of the farthest Voronoi diagram. Barequet and Papadopoulou [3] introduced a structure on the sphere of directions, called the Gaussian map, describing those unbounded parts. The Gaussian map associates with each cell of a diagram its unbounded directions. This results in a subdivision of the sphere of directions.

* Work on this paper by the first author was supported in part by BSF Grant 2017684. The last two authors were supported in part by the Swiss National Science Foundation, project SNF-200021E-154387.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

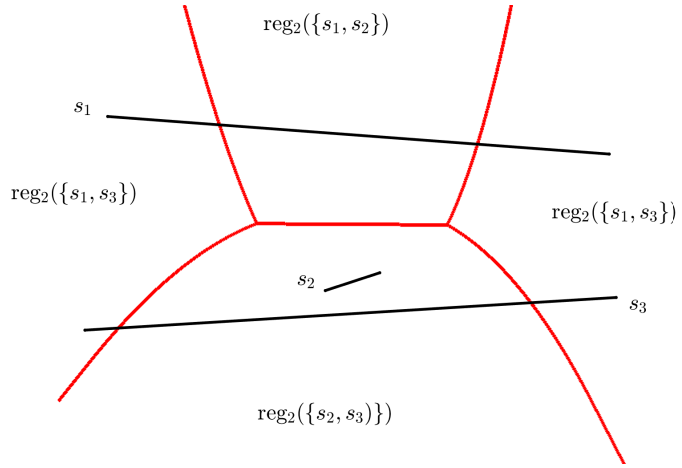


Figure 1 The order-2 Voronoi diagram (in red) of three segments s_1, s_2, s_3 in the plane.

In the current paper, we study the Gaussian map of order-k (and farthest) Voronoi diagrams of line segments and lines as sites in $\mathbb{R}^d$. We characterize the unbounded directions of the cells in these diagrams. We derive the bound $O(\min(k, n-k)n^{d-1})$ on the complexity of the Gaussian map of order-k Voronoi diagrams for these sites. We prove that when sites are segments, the complexity of the Gaussian map is $\Omega(k^{d-1})$, which is tight when $n - k = O(1)$. We also derive the bound $\Omega(k^{d-1})$ on the complexity of the entire order-k Voronoi diagram. We state an algorithm which computes the Gaussian map of the farthest Voronoi diagram in a three-dimensional space in worst-case optimal time $O(n^2)$.

2 Preliminaries

2.1 Order-k Voronoi diagrams

Let S be a set of non-intersecting sites in $\mathbb{R}^d$. In this paper, we consider n line segments or/and lines in $\mathbb{R}^d$ as sites. We assume that the directions of any d lines are linearly independent and no $d+1$ sites touch the same hyperplane, where sites contained in that hyperplane are counted twice. We denote by $d(x, y)$ the Euclidean distance between two points $x, y \in \mathbb{R}^d$. The distance $d(x, s)$ from a point $x \in \mathbb{R}^d$ to a site $s \in S$ is defined as $d(x, s) = min\{d(x, s)|y \in s\}$.

▶ **Definition 2.1.** For a subset of sites $H \subset S$ of cardinality $|H| = k$, the *order-k region* of H is the set of points in $\mathbb{R}^d$ whose distance to any site in H is smaller than to any site not in H, denoted as

$$\mathrm{reg}_k(H) = \{p \in \mathbb{R}^d \mid \forall h \in H\ \forall s \in S \setminus H :\ d(p, h) < d(p, s)\}.$$

The order-k regions form a subdivision of $\mathbb{R}^d$. The induced cell complex, denoted by $\mathrm{VD}_k(S)$, is called the *order-k Voronoi diagram* of S. If $k = 1$, this diagram is the well-known nearest-neighbor Voronoi diagram. For $k = n - 1$, it is the *farthest Voronoi diagram*, denoted by $\mathrm{FVD}(S)$. Its *farthest regions* can also be defined directly as

$$\mathrm{freg}(h) = \{p \in \mathbb{R}^d \mid s \in S \setminus \{h\} :\ d(p, h) > d(p, s)\}.$$

The i-skeleton of a Voronoi diagram is the union of all its j dimensional cells, where $j \leq i$.

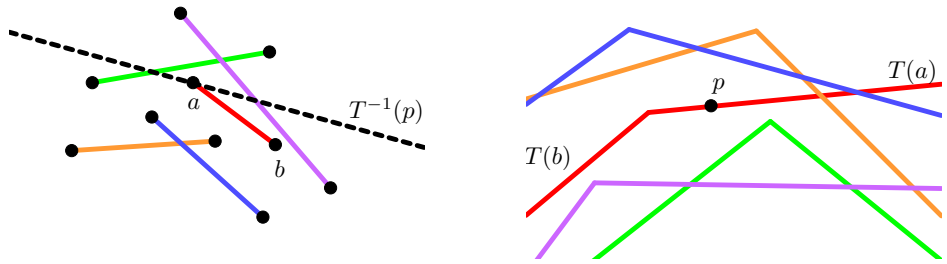


Figure 2 Point-hyperplane duality applied to segments: (left) Segments in primal space; and (right) their corresponding wedges in dual space

2.2 Point-Hyperplane Duality

Under the well-known standard point-hyperplane duality T in $\mathbb{R}^d$, a point $p \in \mathbb{R}^d$ is transformed to a nonvertical hyperplane $T(p)$, and vice versa. The transformation maps a point with coordinates $(p_1, p_2, ..., p_d)$ to the hyperplane $T(p)$ which satisfies the equation $x_d = -p_d + \sum_{i=1}^{d-1} p_i x_i$. The transformation is an involution, i.e., $T = T^{-1}$.

For a segment $s = uv$, the hyperplanes $T(u)$ and $T(v)$ partition the dual space into four *wedges*, among which the *lower wedge* (resp., the *upper wedge*) is the one that lies below (resp., above) both $T(u)$ and $T(v)$. The apex of the wedge is the intersection of $T(u)$ and $T(v)$.

Let S be a set of n segments, which corresponds in dual space to an arrangement of lower wedges. Let L_k be the kth level of that arrangement. Let p be a point on L_k, which touches the dual wedge of segment s, and let H be the set of segments whose wedge is below p. Then, the point p corresponds to a hyperplane $T^{-1}(p)$ which touches the segment s. The closed halfspace above $T^{-1}(p)$ has a non-empty intersection with the segments in H. The open halfspace above $T^{-1}(p)$ does not intersect any segment in $S \setminus H$, see Figure 2. We will use this property when we study the Gaussian map, which is defined in the next section.

2.3 Definition of the Gaussian Map

Let M be a cell complex in $\mathbb{R}^d$. We generalize the notion of the Gaussian map [3] which encodes information about the unbounded cells of M. This structure is of particular interest when all cells of M are unbounded. For example, all d-dimensional cells of the farthest Voronoi diagram of segments and/or lines are unbounded.

▶ **Definition 2.2.** A cell in M is called *unbounded in direction* $\overrightarrow{v}$ if it contains a ray with direction $\overrightarrow{v}$. The *Gaussian map* of M, denoted by $\mathrm{GM}(M)$, maps each cell in M to its unbounded directions, which are encoded on the unit sphere $\mathbb{S}^{d-1}$, see Figure 3. Let c be a cell of M; the set of directions, in which c is unbounded, is called the *region of* c on $\mathrm{GM}(M)$.

In this paper, we focus on cell complexes, such as the farthest Voronoi diagram and the order-k Voronoi diagram of lines and segments, where cells have unbounded direction and the Gaussian map implies a partition of $\mathbb{S}^{d-1}$. This induces a cell complex on $\mathbb{S}^{d-1}$. The collection of cells on the Gaussian map of a Voronoi diagram $\mathrm{VD}_k(S)$, which correspond to the same set of sites $H \subset S$, is called the *region of* H on $\mathrm{GM}(\mathrm{VD}_k(S))$. Note that a Gaussian map region of a cell can consist of several cells, *e.g.*, $\mathrm{reg}_2(\{s_3, s_4\})$ in Figure 3.

The order-k Voronoi diagram and its Gaussian map consist of vertices, edges, and cells in higher dimensions. The *complexity* of the order-k Voronoi diagram or the Gaussian map is the total number of all its cells of all dimensions.

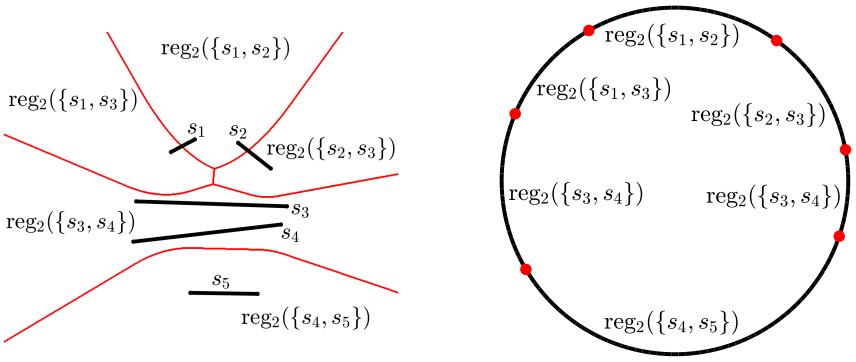


Figure 3 An order-2 Voronoi diagram $\mathrm{VD}_2(\{s_1, s_2, ..., s_5\})$ (left) and its Gaussian map (right).

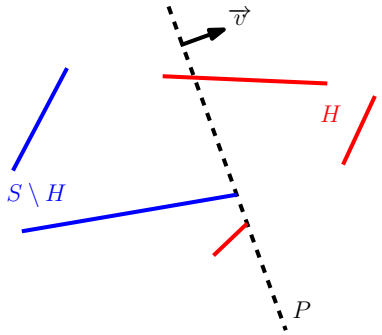


Figure 4 A supporting hyperplane P (in black, dashed) of sites H (in red) in direction $\overrightarrow{v}$.

3 Results

3.1 Supporting hyperplane

We first derive a characterization of the segments which induce an unbounded region of the order-k Voronoi diagram in a given direction.

▶ **Definition 3.1.** Let S be a set of segments, and let H be a subset of S. A hyperplane P is called a *supporting hyperplane* of H and S in direction $\overrightarrow{v}$ if

1. P is orthogonal to $\overrightarrow{v}$;
2. The closed halfspace P^+, bounded by P and unbounded in direction $\overrightarrow{v}$, has a non-empty intersection with each of the sites in H; and
3. The sites in $S \setminus H$ do not intersect the interior of P^+, and at least one site in $S \setminus H$ touches P.

Figure 4 illustrates a hyperplane supporting three segments.

▶ **Theorem 3.2.** *A set of segments H, with $|H| = k$, induces an unbounded region in direction $\overrightarrow{v}$ in the order-k Voronoi diagram of segments S, if and only if there exists a supporting hyperplane of H and S in direction $\overrightarrow{v}$.*

A supporting hyperplane, which touches i segments, corresponds to an unbounded cell of dimension $d - i + 1$ in the order-k Voronoi diagram. The proof of Theorem 3.2 and Theorem 3.7 is given in the full version.

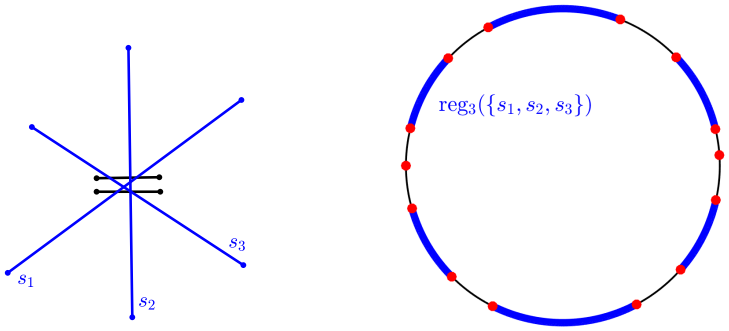


Figure 5 An instance of 5 segments (left), which has one region $\mathrm{reg}_3(\{s_1, s_2, s_3\})$, shown in blue, on the Gaussian map of the order-3 Voronoi diagram (right) with high complexity.

3.2 Combinatorial Properties of the Gaussian Map

The next theorem provides a lower bound on the complexity of the Gaussian map of order-k Voronoi diagrams. This bound is meaningful if k is a function of n.

▶ **Theorem 3.3.** *Let S be a set of n line segments in $\mathbb{R}^d$. The complexity of a single region of the Gaussian map of the order-k Voronoi diagram is $\Omega(k^{d-1})$ in the worst-case.*

Proof. The bound is shown by a generalization of the examples provided for $\mathbb{R}^2$ [1, 11]. Place k long segments connecting almost antipodal points on a $(d-1)$-dimensional hypersphere and $n-k$ additional short segments near the center of the sphere, see Figure 5. Any $d-1$ tuple of long segments, together with a specific short segment, define a supporting hyperplane corresponding to an unbounded edge of the order-k Voronoi diagram. An unbounded edge of the diagram manifests as a vertex on $\mathrm{GM}(\mathrm{FVD}(S))$. All these vertices are on the boundary of the Gaussian map region of the long segments. ◀

▶ **Theorem 3.4.** *The complexity of the Gaussian map of the order-k Voronoi diagram of n segments in $\mathbb{R}^d$ is $O(\min(k, n-k)n^{d-1})$.*

Proof. In order to derive an upper bound on the complexity of the Gaussian map, we use the point-hyperplane duality transformation T, which establishes a 1-1 correspondence between the upper Gaussian map of the order-k Voronoi diagram and the kth level of the arrangement of d-dimensional wedges. The lower Gaussian map is constructed in the same manner. Each segment is mapped to a lower wedge in dual space, which is bounded by two half-hyperplanes. Let p be a point in dual space. Each wedge below p corresponds to a segment in primal space, which has a non-empty intersection with the open halfspace above hyperplane $T^{-1}(p)$. Each wedge touching p corresponds to a segment in primal space, which is touching the closed halfspace above hyperplane $T^{-1}(p)$. Each wedge above p corresponds to a segment in primal space, whose intersection with the closed halfspace above hyperplane $T(p)$ is empty. Therefore, every point on the kth level of the arrangement of the lower wedges corresponds to a hyperplane in primal space which supports k segments. The upper or lower envelope of those wedges, composed of two half-hyperplanes each, has complexity $O(n^{d-1})$ [6].

Using the bound on the lower envelope, we can now also bound the complexity of the $\leq k$-level of the arrangement of lower wedges. We apply a result by Clarkson and Shor [5] to derive a complexity of $O\left((k+1)^d \left(\frac{n}{k+1}\right)^{d-1}\right) = O\left(kn^{d-1}\right)$. We can derive a similar upper

bound of $O((n-k)n^{d-1})$ by using the complexity of the upper envelope of lower wedges as a basis. The upper Gaussian map of the order-k Voronoi diagram corresponds to the $\leq k$-level of the lower wedges. Combining the two bounds completes the proof. ◀

Note that the bounds in Theorems 3.3 and 3.4 are tight for $n-k=O(1)$. In that case, the complexity of the Gaussian map of the order-k Voronoi diagram of n segments is $\Theta(n^{d-1})$ in the worst case.

3.3 Algorithm

▶ **Theorem 3.5.** *Let S be a set of n lines segments in $\mathbb{R}^3$. Then,* $\mathrm{GM}(\mathrm{FVD}(S))$ *can be constructed in worst-case optimal $O(n^2)$ time.*

Proof. We dualize each segment to derive a set of n lower wedges. The upper Gaussian map of the segments corresponds to the upper envelope of the lower wedges in dual space, as described in the proof of Theorem 3.4. The upper envelope of those wedges, each composed of two halfplanes, can be constructed in $O(n^2)$ time [6]. The lower Gaussian map can be constructed in the same manner. ◀

3.4 Properties of the order-k Voronoi Diagram

It was mentioned [3] that the complexity of the farthest Voronoi diagram in $\mathbb{R}^3$ is $O(n^{3+\varepsilon})$ (for any $\varepsilon > 0$), following the general bound on the upper envelope of "well-behaved" surfaces [12]. The same upper bound also holds for the order-k Voronoi diagram.

▶ **Corollary 3.6** (Theorem 3.3)**.** *The order-k Voronoi diagram of n segments in $\mathbb{R}^d$ has $\Omega(k^{d-1})$ complexity in the worst case. This bound becomes $\Omega(n^{d-1})$ for the farthest Voronoi diagram.*

Obviously, the stated lower bound for the order-k diagram is meaningful only for high values of k. Theorem 3.6 can be proven by showing that the number of vertices of the Gaussian map is $\Omega(k^{d-1})$ in the worst case. Each vertex of the Gaussian map corresponds to one edge in the Voronoi diagram. On the other hand, one edge of the diagram creates at most two vertices on the Gaussian map.

▶ **Theorem 3.7.** *The $(d-1)$-skeleton of the farthest Voronoi diagram of segments is connected.*

4 Lines and Combinations of Lines and Segments

For any set of n lines, there is a set of n segments, such that the Gaussian map of the order-k Voronoi diagram of the lines is the same as the one of the segments. Hence, Theorems 3.4, 3.5 extend to lines as sites. The same bounds hold for combined segments and lines as sites.

References

1 Franz Aurenhammer, Robert L. S. Drysdale, and Hannes Krasser. Farthest line segment Voronoi diagrams. *Information Processing Letters*, 100(6):220–225, 2006.

2 Franz Aurenhammer, Rolf Klein, and Der-Tsai Lee. *Voronoi diagrams and Delaunay triangulations*. World Scientific Publishing Company, 2013.

3 Gill Barequet and Evanthia Papadopoulou. On the farthest-neighbor Voronoi diagram of segments in three dimensions. In *10th International Symposium on Voronoi Diagrams in Science and Engineering (ISVD)*, pages 31–36. IEEE, 2013.

4 Bernard Chazelle. An optimal convex hull algorithm and new results on cuttings (extended abstract). In *32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico*, pages 29–38. IEEE Computer Society, 1991.

5 Kenneth L. Clarkson and Peter W. Shor. Applications of random sampling in computational geometry, II. *Discrete & Computational Geometry*, 4(5):387–421, 1989.

6 Herbert Edelsbrunner, Leonidas J. Guibas, and Micha Sharir. The upper envelope of piecewise linear functions: Algorithms and applications. *Discrete & Computational Geometry*, 4:311–336, 1989.

7 Herbert Edelsbrunner and Raimund Seidel. Voronoi diagrams and arrangements. *Discrete & Computational Geometry*, 1:25–44, 1986.

8 Hazel Everett, Daniel Lazard, Sylvain Lazard, and Mohab Safey El Din. The Voronoi diagram of three lines. *Discrete & Computational Geometry*, 42(1):94–130, 2009.

9 Victor Klee. On the complexity of d-dimensional Voronoi diagrams. *Archiv der Mathematik*, 34(1):75–80, 1980.

10 Joseph S. B. Mitchell and Joseph O'Rourke. Computational geometry column 42. *International Journal of Computational Geometry & Applications*, 11(5):573–582, 2001.

11 Evanthia Papadopoulou and Maksym Zavershynskyi. The higher-order Voronoi diagram of line segments. *Algorithmica*, 74(1):415–439, 2016.

12 Micha Sharir. Almost tight upper bounds for lower envelopes in higher dimensions. *Discrete & Computational Geometry*, 12:327–345, 1994.

Hamiltonicity for convex shape Delaunay and Gabriel graphs*

Prosenjit Bose[1], Pilar Cano[1,2], Maria Saumell[3,4], and Rodrigo I. Silveira[2]

1 **School of Computer Science, Carleton University, Ottawa**
jit@scs.carleton.ca
2 **Department de Matemàtiques, Universitat Politècnica de Catalunya**
{m.pilar.cano, rodrigo.silveira}@upc.edu
3 **Institute of Computer Science, The Czech Academy of Sciences**
4 **Department of Theoretical Computer Science, Faculty of Information Technology, Czech Technical University in Prague**
maria.saumell@fit.cvut.cz

Abstract

We study Hamiltonicity for some of the most general variants of Delaunay and Gabriel graphs. Let S be a point set in the plane. The k-order Delaunay graph of S, denoted k-$DG_{\mathcal{C}}(S)$, has vertex set S and edge pq provided that there exists *some* homothet of $\mathcal{C}$ with p and q on its boundary and containing at most k points of S different from p and q. The k-order Gabriel graph k-$GG_{\mathcal{C}}(S)$ is defined analogously, except for the fact that the homothets considered are restricted to be *smallest* homothets of $\mathcal{C}$ with p and q on its boundary. We provide upper bounds on the minimum value of k for which k-$GG_{\mathcal{C}}(S)$ is Hamiltonian. Since k-$GG_{\mathcal{C}}(S) \subseteq k$-$DG_{\mathcal{C}}(S)$, all results carry over to k-$DG_{\mathcal{C}}(S)$. In particular, we give upper bounds of 24 for every $\mathcal{C}$ and 15 for every point-symmetric $\mathcal{C}$. We also improve the bound to 7 for squares, 11 for regular hexagons, 12 for regular octagons, and 11 for even-sided regular t-gons (for $t \geq 10$).

1 Introduction

The study of the combinatorial properties of geometric graphs has played an important role in the area of Discrete and Computational Geometry. One of the fundamental structures that has been studied intensely is the *Delaunay triangulation* of a planar point set (see [9] for an encyclopedic treatment of this structure). It was conjectured by Shamos [10] that the Delaunay triangulation contains a Hamiltonian cycle. This was disproved by Dillencourt [5]. However, Dillencourt [6] showed that Delaunay triangulations are *almost* Hamiltonian, in the sense that they are 1-tough.[1]

Focus then shifted on determining how much the definition of the Delaunay triangulation can be loosened to achieve Hamiltonicity. To this end, Chang et al. [4] showed that the 19-Delaunay graph is Hamiltonian.[2] Given a planar point set S, the *k-Delaunay graph* has vertex set S and edge pq provided that there exists a disk with p and q on the boundary

* P.B. was partially supported by NSERC. P.C. was supported by CONACyT. M.S. was partially supported by the Czech Science Foundation, grant number GJ19-06792Y, and by institutional support RVO:67985807. 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] A graph is *1-tough* if removing any k vertices from it results in at most k connected components.

[2] According to the definition of k-DG in [4], they showed Hamiltonicity for 20-DG.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

Type of shape $\mathcal{C}$	k
Convex	24
Point-symmetric convex	15
Regular octagons	12
Regular hexagons & even-sided regular t-gons, with $t \geq 10$	11
Squares	7

Table 1 Obtained upper bounds on the minimum k for which k-$GG_{\mathcal{C}}(S)$ is Hamiltonian.

containing at most k points of S different from p and q.[3] If the disk with p and q on its boundary is restricted to disks with pq as diameter, then the graph is called the k-Gabriel graph and is a subgraph of the k-Delaunay graph. In fact, Chang et al. [4] showed that the 19-Gabriel graph is Hamiltonian. This was subsequently lowered to $k = 15$ [1] and the current best bound is $k = 10$ [8]. It is conjectured that 1-Delaunay is Hamiltonian [1].

In this article, we generalize the above results by replacing the disk with an arbitrary convex shape. We show that the k-Gabriel graph is Hamiltonian for any convex shape $\mathcal{C}$ when $k \geq 24$, and give improved bounds for various more specific convex shapes. Table 1 summarizes the bounds obtained. Our results rely on tools from metrics and packings.

2 Convex distances and the $\mathcal{C}$-Gabriel graph

Let p and q be two points in the plane. Let $\mathcal{C}$ be a compact convex set that contains the origin, denoted $\bar{o}$, in its interior. The *convex distance* $d_{\mathcal{C}}(p, q)$ is defined in the following way: If $p = q$, then $d_{\mathcal{C}}(p, q) = 0$. Otherwise, $d_{\mathcal{C}}(p, q) = \frac{d(p,q)}{d(p,q')}$, where q' is the intersection of the ray from p to q with the translate of $\mathcal{C}$ by the vector $\overrightarrow{\bar{o}p}$ (see Figure 1). The convex set $\mathcal{C}$ is the *unit $\mathcal{C}$-disk* of $d_{\mathcal{C}}$ with center $\bar{o}$, i.e., every point p in $\mathcal{C}$ satisfies that $d_{\mathcal{C}}(\bar{o}, p) \leq 1$. The *$\mathcal{C}$-disk with center c and radius r* is defined as the homothet of $\mathcal{C}$ centered at c and with scaling factor r.

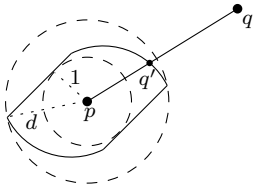


Figure 1 Convex distance from p to q.

The *triangle inequality* holds: $d_{\mathcal{C}}(p, q) \leq d_{\mathcal{C}}(p, r) + d_{\mathcal{C}}(r, q), \forall p, q, r \in \mathbb{R}^2$. However, this distance may not define a metric when $\mathcal{C}$ is not *point-symmetric*[4] about the origin, since there may be points p, q for which $d_{\mathcal{C}}(p, q) \neq d_{\mathcal{C}}(q, p)$. When $\mathcal{C}$ is point-symmetric with respect to the origin, $d_{\mathcal{C}}$ is called a *symmetric convex distance function*. Such a distance defines a metric; moreover, $d_{\mathcal{C}}(\bar{o}, p)$ defines a *norm*[5] of a *metric space*. In addition, if a point p is on the line segment ab, then $d_{\mathcal{C}}(a, b) = d_{\mathcal{C}}(a, p) + d_{\mathcal{C}}(p, b)$ (see [2, Chapter 7]).

[3] Note that this implies that the standard Delaunay triangulation is the 0-Delaunay graph.

[4] A shape $\mathcal{C}$ is point-symmetric with respect to a point $x \in \mathcal{C}$ provided that for every point $p \in \mathcal{C}$ there is a corresponding point $q \in \mathcal{C}$ such that $pq \in \mathcal{C}$ and x is the midpoint of pq.

[5] A function $\rho(x)$ is a norm if: (a) $\rho(x) = 0$ if and only if $x = \bar{o}$, (b) $\rho(\lambda x) = |\lambda|\rho(x)$ where $\lambda \in \mathbb{R}$, and (c) $\rho(x + y) \leq \rho(x) + \rho(y)$.

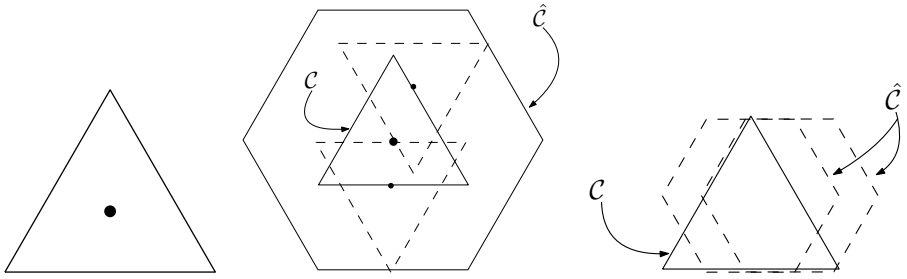


Figure 2 Left: A triangle is a $\mathcal{C}$ shape. Center: $\hat{\mathcal{C}}$ for this triangle is a hexagon. Right: the shape $\hat{\mathcal{C}}$ with radius $\frac{1}{2}$ does not contain $\mathcal{C}$.

Let S be a set of points in the plane satisfying the following general position assumption: For each pair $p, q \in S$, any minimum homothet of $\mathcal{C}$ having p and q on its boundary does not contain any other point of S on its boundary. The *k-order $\mathcal{C}$-Delaunay graph* of S, denoted k-$DG_{\mathcal{C}}(S)$, is the graph with vertex set S such that, for each pair of points $p, q \in S$, the edge pq is in k-$DG_{\mathcal{C}}(S)$ if there exists a $\mathcal{C}$-disk that has p and q on its boundary and contains at most k points of S different from p and q. When $k = 0$ and $\mathcal{C}$ is a circle, k-$DG_{\mathcal{C}}(S)$ is the standard *Delaunay triangulation*.

Aurenhammer and Paulini [3] showed how to define a point-symmetric distance function from any convex shape $\mathcal{C}$, as follows. Denote by $\mathcal{C}_v$ the shape $\mathcal{C}$ with $\bar{o}$ translated by vector v. The distance from p to q is given by the scaling factor of a smallest homothet containing p and q on its boundary, which is equivalent to $\min_{v \in \mathcal{C}} d_{\mathcal{C}_v}(p, q) = d_{\hat{\mathcal{C}}}(p, q)$ where $\hat{\mathcal{C}} = \bigcup_{v \in \mathcal{C}} \mathcal{C}_v$. The set $\hat{\mathcal{C}}$ is a point-symmetric convex set that is the Minkowski sum[6] of $\mathcal{C}$ and its shape reflected about its center. For an example, see Figure 2. The diameter and width of $\hat{\mathcal{C}}$ is twice the diameter and width of $\mathcal{C}$, respectively. Moreover, when $\mathcal{C}$ is point-symmetric, $d_{\hat{\mathcal{C}}}(p, q) = \frac{d_{\mathcal{C}}(p,q)}{2}$.

We define the *k-order $\mathcal{C}$-Gabriel graph* of S, denoted k-$GG_{\mathcal{C}}(S)$, as the graph with vertex set S such that, for every pair of points $p, q \in S$, the edge pq is in k-$GG_{\mathcal{C}}(S)$ if and only if there exists a $\mathcal{C}$-disk with radius $d_{\hat{\mathcal{C}}}(p, q)$ that has p and q on its boundary and contains at most k points of S different from p and q. From the definition of k-$GG_{\mathcal{C}}(S)$ and k-$DG_{\mathcal{C}}(S)$ we note that k-$GG_{\mathcal{C}}(S) \subseteq k$-$DG_{\mathcal{C}}(S)$, and it can be a proper subgraph. See Figure 3 for an example. Further, when $\mathcal{C}$ is not point-symmetric, then $\hat{\mathcal{C}}$ contains $\mathcal{C}$ in its interior; however, for some shapes it is not true that the $\hat{\mathcal{C}}$-disk with radius $\frac{1}{2}$ contains $\mathcal{C}$ (refer to Figure 2, right). Thus, for asymmetric shapes $\mathcal{C}$, in general $GG_{\hat{\mathcal{C}}} \not\subseteq GG_{\mathcal{C}}$.

3 Hamiltonicity for convex shapes

3.1 General convex shapes

Define $\mathcal{H}$ to be the set of all Hamiltoninan cycles of the point set S. Define the $d_{\hat{\mathcal{C}}}$-length sequence of $h \in \mathcal{H}$, denoted $ds_{\mathcal{C}}(h)$, as the edge sequence sorted in decreasing order with respect to the length of the edges in $d_{\hat{\mathcal{C}}}$-metric. Sort the elements of $\mathcal{H}$ in lexicographic order with respect to their $d_{\hat{\mathcal{C}}}$-length sequence, breaking ties arbitrarily. This order is strict. For $h_1, h_2 \in \mathcal{H}$, if h_1 is smaller than h_2 in this order, we write $h_1 \prec h_2$.

[6] The Minkowski sum of two sets A and B is defined as $A \oplus B = \{a + b : a \in A, b \in B\}$.

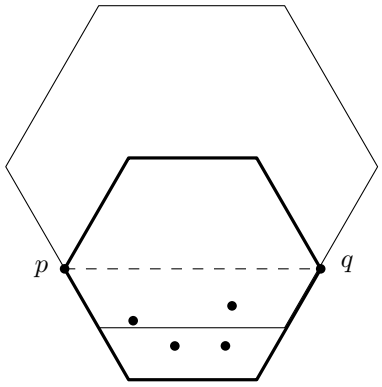


Figure 3 $\mathcal{C}$ is a regular hexagon. Edge pq is in 2-$DG_{\mathcal{C}}(S)$ but it is not in 2-$GG_{\mathcal{C}}(S)$.

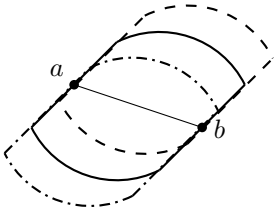


Figure 4 Many $\mathcal{C}$-disks $\mathcal{C}(a,b)$ may exist for a and b.

For simplicity, denote by $\mathcal{C}_r(a,b)$, a $\mathcal{C}$-disk with radius r containing the points a and b on its boundary. For the special case of a *diametral disk*, i.e., when the radius of $\mathcal{C}_r(a,b)$ is $d_{\hat{\mathcal{C}}}(a,b)$, we denote it as $\mathcal{C}(a,b)$. Note that $\mathcal{C}(a,b)$ may not be unique, see Figure 4. In addition, we denote by $D_{\mathcal{C}}(c,r)$ the $\mathcal{C}$-disk centered at point c with radius r.

▶ Claim 3.1. Let $\mathcal{C}$ be a point-symmetric convex shape. Let u be a point in the plane different from the origin $\bar{o}$. Let $r < d_{\mathcal{C}}(u,\bar{o})$. Let p be the intersection point of $D_{\mathcal{C}}(u,r)$ and line segment $\bar{o}u$. Let $u' = \lambda u$, with $\lambda > 1 \in \mathbb{R}$, be a point defined by vector u scaled by a factor of λ. Then $D_{\mathcal{C}}(u,r) \subset D_{\mathcal{C}}(u', d_{\mathcal{C}}(u',p))$. (See Figure 5.)

Proof. Let $q \in D_{\mathcal{C}}(u,r)$; then $d_{\mathcal{C}}(u,q) \leq d_{\mathcal{C}}(u,p)$. Since u is on the line segment $u'p$, we have that $d_{\mathcal{C}}(u',p) = d_{\mathcal{C}}(u',u) + d_{\mathcal{C}}(u,p)$. Hence $d_{\mathcal{C}}(u',q) \leq d_{\mathcal{C}}(u',u) + d_{\mathcal{C}}(u,q) \leq d_{\mathcal{C}}(u',u) + d_{\mathcal{C}}(u,p) = d_{\mathcal{C}}(u',p)$. Therefore, $D_{\mathcal{C}}(u,r)$ is contained in $D_{\mathcal{C}}(u', d_{\mathcal{C}}(u',p))$. ◀

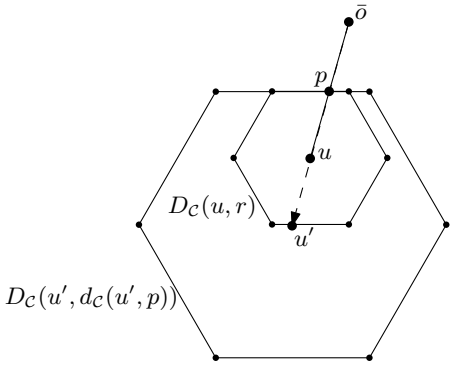


Figure 5 $D_{\mathcal{C}}(u,r)$ is contained in $D_{\mathcal{C}}(u', d_{\mathcal{C}}(u',p))$, where $u' = \lambda u$ with $\lambda > 1$.

The approach we follow to prove our bounds, which is similar to the approach in [1, 4, 8], is to show that the minimum element in $\mathcal{H}$ is contained in k-$GG_{\mathcal{C}}(S)$ for a small value of k. Let h be the minimum element in $\mathcal{H}$. Let $ab \in h$; we can assume without loss of generality that $d_{\hat{\mathcal{C}}}(a,b) = 1$. Let $U = \{u_1, u_2, \ldots, u_k\}$ be the set of points in S different from a and b that are in the interior of $\mathcal{C}(a,b)$.[7] We assume that, when traversing h from b to a, we visit the points of U in order $u_1, \ldots, u_k$. For each point u_i, we define s_i to be the point preceding u_i in h. See Figure 6.

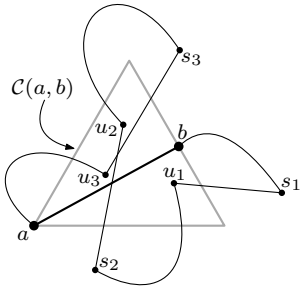


Figure 6 Example of U in $\mathcal{C}(a,b)$.

Note that if a point p is in the interior of $\mathcal{C}(a,b)$, then there exists a $\mathcal{C}(p,q)$ contained in $\mathcal{C}(a,b)$ for any point q on the boundary of $\mathcal{C}(a,b)$. Therefore, $d_{\hat{\mathcal{C}}}(a,u_i) < 1$ and $d_{\hat{\mathcal{C}}}(b,u_i) < 1$ for any $i \in \{1,\ldots,k\}$. Furthermore, we have the following:

▶ Claim 3.2. Let $1 \le i \le k$. Then $d_{\hat{\mathcal{C}}}(a,s_i) \ge \max\{d_{\hat{\mathcal{C}}}(s_i,u_i), 1\}$.

Proof. If $s_1 = b$, then $d_{\hat{\mathcal{C}}}(a,s_1) = 1$ and $d_{\hat{\mathcal{C}}}(s_1,u_1) < 1$. Otherwise, we define $h'=(h \setminus \{ab, s_iu_i\}) \cup \{as_i, u_ib\}$. For the sake of a contradiction, assume that $d_{\hat{\mathcal{C}}}(a,s_i) < \max\{d_{\hat{\mathcal{C}}}(s_i,u_i), 1\}$. Since $d_{\hat{\mathcal{C}}}(a,b) = 1$, this implies that $d_{\hat{\mathcal{C}}}(a,s_i) < \max\{d_{\hat{\mathcal{C}}}(s_i,u_i), d_{\hat{\mathcal{C}}}(a,b)\}$. Moreover, since $u_i \in \mathcal{C}(a,b)$, we have $d_{\hat{\mathcal{C}}}(u_i,b) < 1$. Thus, $\max\{d_{\hat{\mathcal{C}}}(a,s_i), d_{\hat{\mathcal{C}}}(u_i,b)\} < \max\{d_{\hat{\mathcal{C}}}(s_i,u_i), d_{\hat{\mathcal{C}}}(a,b)\}$. Therefore $h' \prec h$, which contradicts the definition of h. ◀

Claim 3.2 implies that, for each $i \in \{1,\ldots,k\}$, the point s_i is not in the interior of $\mathcal{C}(a,b)$.

▶ Claim 3.3. Let $1 \le i < j \le k$. Then $d_{\hat{\mathcal{C}}}(s_i,s_j) \ge \max\{d_{\hat{\mathcal{C}}}(s_i,u_i), d_{\hat{\mathcal{C}}}(s_j,u_j), 1\}$.

The proof of this claim is similar to the proof of Claim 3.2.

Without loss of generality we assume that a is the origin $\bar{o}$. Then, by the definition of $\hat{\mathcal{C}}$, we have that $D_{\hat{\mathcal{C}}}(\bar{o},1)$ contains $\mathcal{C}(a,b)$. Also, from Claim 3.2, we have that s_i is not in the interior of $D_{\hat{\mathcal{C}}}(\bar{o},1)$ for all $i \in \{1,\ldots,k\}$. Let $D_{\hat{\mathcal{C}}}(\bar{o},2)$ be the $\hat{\mathcal{C}}$-disk centered at a with radius 2. For each $s_i \notin D_{\hat{\mathcal{C}}}(\bar{o},2)$, define s_i' as the intersection of $D_{\hat{\mathcal{C}}}(\bar{o},2)$ with the ray $\overrightarrow{as_i}$. We let $s_i' = s_i$ when s_i is inside $D_{\hat{\mathcal{C}}}(\bar{o},2)$. See Figure 7.

▶ Observation 3.4. If $s_j \notin D_{\hat{\mathcal{C}}}(\bar{o},2)$ (with $1 \le j \le k$), the $d_{\hat{\mathcal{C}}}$-distance from s_j' to $D_{\hat{\mathcal{C}}}(\bar{o},1)$ is 1.

▶ **Lemma 3.5.** *For any pair s_i and s_j with $i \neq j$, we have that $d_{\hat{\mathcal{C}}}(s_i',s_j') \ge 1$.*

Proof. If both s_i and s_j are in $D_{\hat{\mathcal{C}}}(\bar{o},2)$, then from Claim 3.3 we have that $d_{\hat{\mathcal{C}}}(s_i',s_j') = d_{\hat{\mathcal{C}}}(s_i,s_j) \ge 1$. In the following, we assume, without loss of generality, that $d_{\hat{\mathcal{C}}}(\bar{o},s_j) \ge d_{\hat{\mathcal{C}}}(\bar{o},s_i)$. Since s_j' is on the line segment $\bar{o}s_j$, we have $s_j = \lambda s_j'$ for some $\lambda > 1 \in \mathbb{R}$. Let p be the intersection point of $D_{\hat{\mathcal{C}}}(\bar{o},1)$ and $\bar{o}s_j$. Since $d_{\hat{\mathcal{C}}}$ defines a norm, we have

[7] By our general position assumption, the only points of S on the boundary of $\mathcal{C}(a,b)$ are a and b.

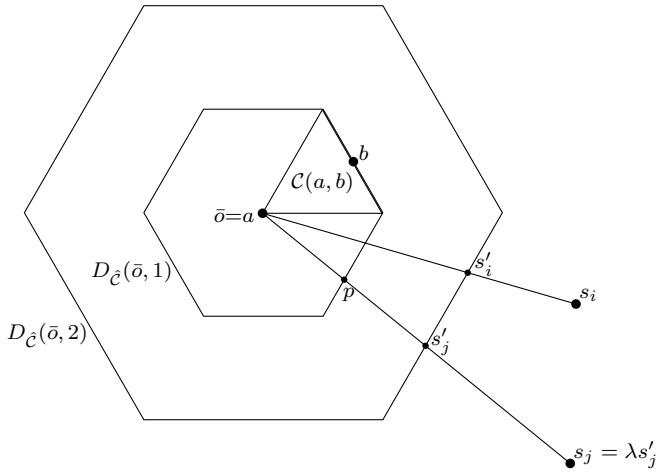


Figure 7 The points s'_i and s'_j are projections of s_i and s_j on $D_{\hat{\mathcal{C}}}(\bar{o},2)$, respectively.

$d_{\hat{\mathcal{C}}}(\lambda s'_j,\bar{o}) = \lambda d_{\hat{\mathcal{C}}}(s'_j,\bar{o})$. By Observation 3.4 we have that $d_{\hat{\mathcal{C}}}(s_j,p) = d_{\hat{\mathcal{C}}}(s_j,\bar{o}) - d_{\hat{\mathcal{C}}}(p,\bar{o}) = \lambda d_{\hat{\mathcal{C}}}(s'_j,\bar{o}) - 1 = 2\lambda - 1$, which is the distance from s_j to $D_{\hat{\mathcal{C}}}(\bar{o},1)$. Further, $d_{\hat{\mathcal{C}}}(s_j,s'_j) = d_{\hat{\mathcal{C}}}(s_j,\bar{o}) - d_{\hat{\mathcal{C}}}(s'_j,\bar{o}) = 2\lambda - 2$. For the sake of a contradiction, assume that $d_{\hat{\mathcal{C}}}(s'_i,s'_j) \leq 1$. If $s_j \notin D_{\hat{\mathcal{C}}}(\bar{o},2)$, we consider two cases:
Case 1) $s_i \in D_{\hat{\mathcal{C}}}(\bar{o},2)$. Then $d_{\hat{\mathcal{C}}}(\bar{o},s_i) \leq 2$. Let $D_{s'_j} = D_{\hat{\mathcal{C}}}(s'_j,1)$. Since $d_{\hat{\mathcal{C}}}(s'_i,s'_j) \leq 1$, we have $s_i \in D_{s'_j}$. From Claim 3.1 it follows that $d_{\hat{\mathcal{C}}}(s_j,s'_i) = d_{\hat{\mathcal{C}}}(s_j,s_i) \leq d_{\hat{\mathcal{C}}}(s_j,p) < d_{\hat{\mathcal{C}}}(s_j,u_j)$, which contradicts Claim 3.3.
Case 2) $s_i \notin D_{\hat{\mathcal{C}}}(\bar{o},2)$. Then $d_{\hat{\mathcal{C}}}(\bar{o},s_i) > 2$. Thus, $s_i = \delta s'_i$ for some $\delta > 1 \in \mathbb{R}$. Moreover, since $d_{\hat{\mathcal{C}}}(\bar{o},s_j) \geq d_{\hat{\mathcal{C}}}(\bar{o},s_i)$ and s'_i, s'_j are on the boundary of $D_{\hat{\mathcal{C}}}(\bar{o},2)$, $\delta \leq \lambda$. Hence, s_i is on the line segment $s'_i(\lambda s'_i)$. Let $D_{s_j} = D_{\hat{\mathcal{C}}}(s_j, 2\lambda - 1)$. Note that $\lambda < 2\lambda - 1$ because $\lambda > 1$. Since $d_{\hat{\mathcal{C}}}$ defines a norm, $d_{\hat{\mathcal{C}}}(s_j,\lambda s'_i) = d_{\hat{\mathcal{C}}}(\lambda s'_j,\lambda s'_i) = \lambda d_{\hat{\mathcal{C}}}(s'_j,s'_i) \leq \lambda < 2\lambda - 1$. Hence, $\lambda s'_i \in D_{s_j}$. In addition, from Claim 3.1 it follows that $D_{s'_j} \subseteq D_{s_j}$. Therefore, $s'_i \in D_{s_j}$. Thus, the line segment $s'_i(\lambda s'_i)$ is contained in D_{s_j}. Hence, $s_i \in D_{s_j}$. Then, $d_{\hat{\mathcal{C}}}(s_j,s_i) \leq 2\lambda - 1 = d_{\hat{\mathcal{C}}}(s_j,p) < d_{\hat{\mathcal{C}}}(s_j,u_j)$ which contradicts Claim 3.3. ◀

▶ **Theorem 3.6.** *For any set of points S in general position and convex shape $\mathcal{C}$, the graph 24-$GG_{\mathcal{C}}(S)$ is Hamiltonian.*

Proof. For each s_i we define the $\hat{\mathcal{C}}$-disk $D_i = D_{\hat{\mathcal{C}}}(s'_i, \frac{1}{2})$. We also set $D_0 := D_{\hat{\mathcal{C}}}(a, \frac{1}{2})$. By Lemma 3.5, each pair of $\hat{\mathcal{C}}$-disks D_i and D_j $(i \neq j)$ are internally disjoint. See Figure 8. Since $s'_i \in D_{\hat{\mathcal{C}}}(\bar{o},2)$ for all i, each disk D_i is inside $D_{\hat{\mathcal{C}}}(a, \frac{5}{2})$. There can be at most $\frac{Area(D_{\hat{\mathcal{C}}}(\bar{o},\frac{5}{2}))}{Area(D_0)} = \frac{(\frac{5}{2})^2 Area(\hat{\mathcal{C}})}{(\frac{1}{2})^2 Area(\hat{\mathcal{C}})} = 25$ disjoint disks in $D_{\hat{\mathcal{C}}}(\bar{o},2)$. Thus, there are at most 24 points s'_i in $D_{\hat{\mathcal{C}}}(\bar{o},1)$, since D_0 is centered at a. Hence, there are at most 24 points in the interior of $\mathcal{C}(a,b)$. ◀

3.2 Point-symmetric convex shapes

Using the fact that $d_{\mathcal{C}}$ defines a metric when $\mathcal{C}$ is point-symmetric, we can improve the upper bound for point-symmetric convex shapes. Indeed, given that $d_{\mathcal{C}} = 2d_{\hat{\mathcal{C}}}$ we can prove that: (i) $d_{\mathcal{C}}(s_i,a) \geq \max\{d_{\mathcal{C}}(s_i,u_i),2\}$; and (ii) $d_{\mathcal{C}}(s_i,s_j) \geq \max\{d_{\mathcal{C}}(s_i,u_i),d_{\mathcal{C}}(s_j,u_j),2\}$, for any $1 \leq i < j \leq k$. By using $\mathcal{C}(a,b)$ instead of $D_{\hat{\mathcal{C}}}(\bar{o},1)$, $D_{\mathcal{C}}(\bar{o},3)$ instead of $D_{\hat{\mathcal{C}}}(\bar{o},2)$, and $D_{\mathcal{C}}(\bar{o},4)$ instead of $D_{\hat{\mathcal{C}}}(\bar{o},\frac{5}{2})$, in combination with arguments similar to those in the previous section, we obtain that 15-$GG_{\mathcal{C}}$ is Hamiltonian.

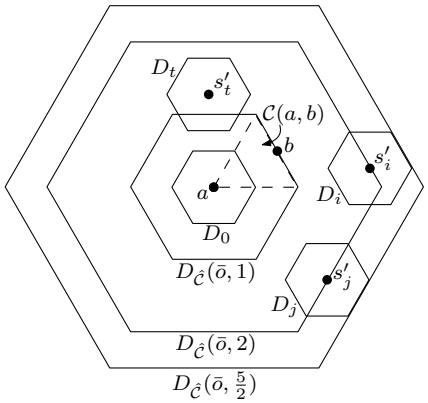


Figure 8 The $\hat{\mathcal{C}}$-disks D_i, D_j and D_t are contained in $D_{\hat{\mathcal{C}}}(a, \frac{5}{2})$.

When $\mathcal{C}$ is a regular polygon $\mathcal{P}_t$ with t even sides we can improve this bound by analyzing specific values of t. When $\mathcal{C}$ is a square, we divide $D_{\mathcal{P}_4}(\bar{o}, 3)$ into nine disjoint squares of radius 1 and show that only seven of them can contain points from $\{s'_1, \ldots, s'_k\}$, with at most one point in each square. Using similar arguments as those for squares, we show that 11-$GG_{\mathcal{P}_6}$ is Hamiltonian. Finally, for the remaining regular polygons with even sides we use that the ex-circle of $D_{\mathcal{P}_{10}}(\bar{o}, 3)$ contains $D_{\mathcal{P}_t}(\bar{o}, 3)$ for all even $t \geq 10$. Such a circle has radius $r \approx 3.154$. Hence, we can prove Hamiltonicity for 11-$GG_{\mathcal{P}_t}$ using a result by Fodor [7] that states that the minimum radius of a circle having 13 points at pairwise Euclidean distance at least 2 is $R \approx 3.236$, which is greater than r. Analogously, we show that there are at most 13 points inside $D_{\mathcal{P}_8}(\bar{o}, 3)$ such that each pair is at Euclidean distance at least 2, which proves Hamiltonicity for 12-$GG_{\mathcal{P}_8}$.

References

1 Manuel Abellanas, Prosenjit Bose, Jesús García-López, 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(6):595–615, 2009.

2 Franz Aurenhammer, Rolf Klein, and Der-Tsai Lee. *Voronoi diagrams and Delaunay triangulations.* World Scientific Publishing Company, 2013.

3 Franz Aurenhammer and Günter Paulini. On shape Delaunay tessellations. *Inf. Process. Lett.*, 114(10):535–541, 2014.

4 Maw-Shang Chang, Chuan Yi Tang, and Richard C. T. Lee. 20-relative neighborhood graphs are Hamiltonian. *J. Graph Theory*, 15(5):543–557, 1991.

5 Michael B. Dillencourt. A non-Hamiltonian, nondegenerate Delaunay triangulation. *Inf. Process. Lett.*, 25(3):149–151, 1987.

6 Michael B. Dillencourt. Toughness and Delaunay triangulations. *Discrete Comput. Geom.*, 5:575–601, 1990.

7 Ferenc Fodor. The densest packing of 13 congruent circles in a circle. *Beitr. Algebra Geom.*, 44(2):431–440, 2003.

8 Tomáš Kaiser, Maria Saumell, and Nico Van Cleemput. 10-Gabriel graphs are Hamiltonian. *Inf. Process. Lett.*, 115(11):877–881, 2015.

9 Atsuyuki Okabe, Barry Boots, Kokichi Sugihara, and Sung Nok Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams.* Wiley, 2000.

10 Michael Shamos. Computational geometry. *PhD Thesis, Yale University*, 1978.

Delaunay triangulations of symmetric hyperbolic surfaces

Matthijs Ebbens[1], Iordan Iordanov[2], Monique Teillaud[2], and Gert Vegter[1]

1 **Bernoulli Institute for Mathematics, Computer Science and Artificial Intelligence, University of Groningen, Netherlands**
2 **Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France**

Abstract

Of the several existing algorithms for computing Delaunay triangulations of point sets in Euclidean space, the incremental algorithm has recently been extended to the Bolza surface, a hyperbolic surface of genus 2. We will generalize this algorithm to so called symmetric hyperbolic surfaces of arbitrary genus. Delaunay triangulations of point sets on hyperbolic surfaces can be constructed by using the fact that such point sets can be regarded as periodic point sets in the hyperbolic plane. However, one of the main issues is then that the result might contain 1- or 2-cycles, which means that the triangulation is not simplicial. As the incremental algorithm that we use can only work with simplicial complexes, this situation must be avoided.

In this work, we will first compute the systole of the symmetric hyperbolic surfaces, i.e., the length of the shortest non-contractible loop. The value of the systole is used in a condition to ensure that the triangulations will be simplicial. Secondly, we will show that it is sufficient to consider only a finite subset of the infinite periodic point set in the hyperbolic plane. Finally, we will algorithmically construct a point set with which we can initialize the algorithm.

1 Introduction

The incremental algorithm, one of the known algorithms for computing Delaunay triangulations of point sets in Euclidean space, inserts the points one by one and updates the triangulation after each insertion [3]. It is used in practice for example in triangulation packages of CGAL [9]. This algorithm has been extended to periodic point sets, which can be seen as the image of a finite point set under the action of a group of translations [6, 5]. For example, given a finite point set in the unit square in the Euclidean plane and the group generated by the Euclidean translations of unit length in the x- and y-direction, one obtains a periodic point set in the Euclidean plane. Equivalently, this can be seen as a finite point set on the flat torus, where the flat torus is identified with the quotient space of the Euclidean plane under the action of the group mentioned above. If we consider the Delaunay triangulation of the periodic point set in the Euclidean plane, and project this triangulation to the flat torus, the result may be non-simplicial. Namely, if for example both endpoints of an edge project to the same point, then we obtain a loop. Since the incremental algorithm that we use assumes that the triangulation is a simplicial complex, this situation must be avoided. It is known that $1-$ and $2-$cycles can be avoided when the inequality $\mathrm{sys}(\mathbb{M}) > 2\delta_{\mathcal{P}}$ is satisfied, where $\mathrm{sys}(\mathbb{M})$ denotes the systole of the surface $\mathbb{M}$, i.e. the length of the shortest non-contractible curve, and $\delta_{\mathcal{P}}$ the diameter of a largest disk not containing any points from the input set $\mathcal{P}$ in its interior. Intuitively, this condition means that the length of every edge in the triangulation is less than $\frac{1}{2}\,\mathrm{sys}(\mathbb{M})$, which implies that there can be no $1-$ or $2-$cycles. To make sure that this condition is satisfied, the triangulation can be initialized with a dummy point set, for which the inequality is satisfied by construction. After inserting the input points, the dummy points are removed (if possible).

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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 will consider Delaunay triangulations on hyperbolic surfaces. Unlike Euclidean surfaces, in general the systole of a hyperbolic surface is unknown. An upper bound of order $O(\log g)$ is known [4, Lemma 5.2.1], where g denotes the genus, but lower bounds exist only for specific families of surfaces, often constructed using algebraic methods; in general the systole can be made arbitrarily small. The exact value of the systole is known for only a few specific hyperbolic surfaces. The Bolza surface, the most symmetric hyperbolic surface of genus 2, is one of the hyperbolic surfaces for which the systole is known. Here the regular hyperbolic octagon is a fundamental region for the group of translations. A generalization and implementation of the incremental algorithm for the Bolza surface is known [2, 8].

We will generalize the incremental algorithm for the Bolza surface to what we call 'symmetric hyperbolic surfaces' of arbitrary genus. Just like the Bolza surface (genus $g = 2$) corresponds to the regular hyperbolic octagon, the symmetric hyperbolic surface $\mathbb{M}_g$ of genus $g \geq 2$ corresponds to the regular hyperbolic $4g$-gon. Firstly, we derive the value of the systole of these surfaces to be able to verify the condition $\text{sys}(\mathbb{M}_g) > 2\delta_{\mathcal{P}}$. Secondly, we show that it is sufficient to consider a finite subset of the (infinite) periodic point set. Finally, we will construct for each symmetric hyperbolic surface a dummy point set with which to initialize the algorithm.

2 Preliminaries

2.1 Hyperbolic geometry

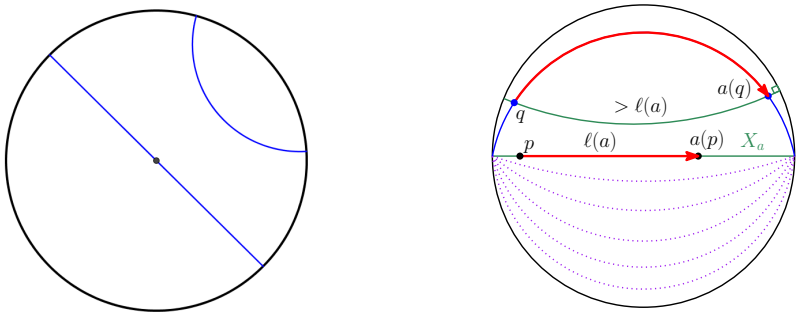
We will study periodic triangulations in the hyperbolic plane, or equivalently, triangulations on hyperbolic surfaces. There are several models for the hyperbolic plane; we will use the Poincaré disk model [1]. Here, the hyperbolic plane $\mathbb{H}^2$ is represented by the open unit disk $\mathbb{D}$ in the complex plane. This space is endowed with a specific metric, for which the hyperbolic lines (i.e. geodesics) are represented by diameters of $\mathbb{D}$ or circle arcs intersecting the boundary of $\mathbb{D}$ orthogonally (see Figure 1a). Isometries of $\mathbb{H}^2$ can be written in the form

$$f(z) = \frac{\alpha z + \beta}{\overline{\beta} z + \overline{\alpha}},$$

where $\alpha, \beta \in \mathbb{C}$ such that $|\alpha|^2 - |\beta|^2 = 1$. There are several types of isometries of $\mathbb{H}^2$, of which we will only consider *hyperbolic* isometries, also called translations, for which the real part of α is larger than 1. Every hyperbolic translation f has an invariant hyperbolic line, called the axis of f (see Figure 1b). A striking difference between translations of the Euclidean plane and translations of the hyperbolic plane is that the latter are in general not commutative.

▶ **Definition 2.1.** *A hyperbolic surface is a connected 2-dimensional manifold that is locally isometric to an open subset of* $\mathbb{H}^2$.

It is known that every hyperbolic surface $\mathbb{M}$ can be written as a quotient space $\mathbb{M} = \mathbb{H}^2/\Gamma$ of the hyperbolic plane under the action of a Fuchsian group Γ, i.e., a discrete subgroup of the group of isometries of $\mathbb{H}^2$ [11]. This quotient space can be represented by a polygon that is a fundamental region for the action of Γ, combined with a side pairing. In the case of the flat torus, the unit square is a fundamental polygon and the side pairings are given by the translations in the $x-$ and $y-$direction. For the Bolza surface, a fundamental domain is given by the regular hyperbolic octagon with total interior angle 2π, where opposite sides are paired (see Figure 2). In the same way, the symmetric hyperbolic surface $\mathbb{M}_g = \mathbb{H}^2/\Gamma_g$



(a) The Poincaré disk model of the hyperbolic plane with some hyperbolic lines

(b) A hyperbolic translation a, its axis X_a and the image of a point q not on the axis

Figure 1 Hyperbolic geometry

of genus $g \geq 2$ corresponds to the regular hyperbolic polygon F_g with total interior angle 2π, where opposite sides are paired.

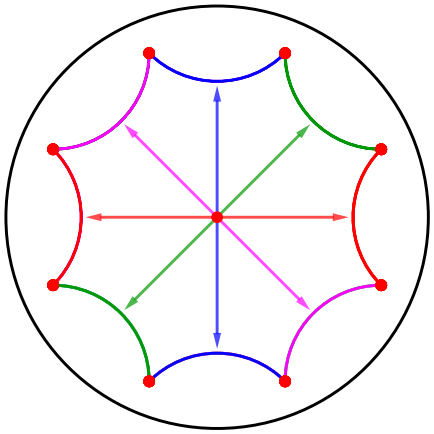


Figure 2 The Bolza surface

The action of the group Γ_g on the fundamental region F_g induces a tesselation of the hyperbolic plane into $4g$-gons. Because the interior angles of the $4g$-gon F_g add up to 2π, it can be seen that $4g$ polygons meet in every vertex in the tesselation. As a comparison, the group of translations of the flat torus induce a tesselation of the Euclidean plane into unit squares.

Finally, the systole of a hyperbolic surface $\mathbb{M}$ is the length of smallest non-contractible closed curve and is denoted by $\text{sys}(\mathbb{M})$. We will sketch a derivation of the value of the systole of the symmetric hyperbolic surfaces in Section 3.

2.2 Delaunay triangulations

In this work we will consider simplicial Delaunay triangulations, which satisfy the following two conditions:

- they are a simplicial complex,

- they satisfy the empty circle property.

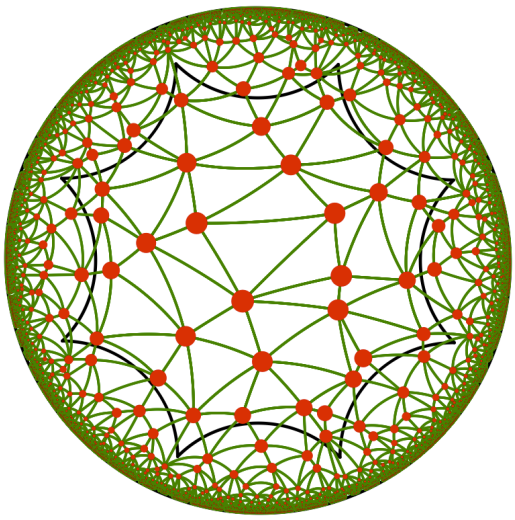


Figure 3 Example of a Delaunay triangulation of a periodic point set in the hyperbolic plane

Consider a finite point set $\mathcal{P}$ on a hyperbolic surface $\mathbb{M} = \mathbb{H}^2/\Gamma$. To define the Delaunay triangulation $\mathrm{DT}_{\mathbb{M}}(\mathcal{P})$ of $\mathcal{P}$ on $\mathbb{M}$, we can consider the images $\Gamma_g\mathcal{P}$ of $\mathcal{P}$ under the group action of Γ_g. Then we project the Delaunay triangulation $\mathrm{DT}_{\mathbb{H}}(\Gamma\mathcal{P})$ of the infinite point set $\Gamma\mathcal{P}$ in $\mathbb{H}^2$ to $\mathbb{M}$ using the universal covering map $\pi : \mathbb{H}^2 \to \mathbb{H}^2/\Gamma$. In that case, we could define "$\mathrm{DT}_{\mathbb{M}}(\mathcal{P}) = \pi(\mathrm{DT}_{\mathbb{H}}(\Gamma\mathcal{P}))$". However, the result is not necessarily a simplicial complex. It is known that the following criterion is sufficient to guarantee that $\pi(\mathrm{DT}_{\mathbb{H}}(\Gamma\mathcal{P}))$ is a simplicial complex [2]. Here $\delta_{\mathcal{P}}$ denotes the diameter of the largest disk in $\mathbb{H}^2$ that does not contain any point of $\Gamma\mathcal{P}$ in its interior.

▶ **Proposition 2.2.** *Let $\mathbb{M} = \mathbb{H}^2/\Gamma$ be a hyperbolic surface and $\mathcal{P} \subset \mathbb{M}$ a finite point set. If $\mathrm{sys}(\mathbb{M}) > 2\delta_{\mathcal{P}}$, then $\pi(\mathrm{DT}_{\mathbb{H}}(\Gamma\mathcal{P}))$ is a simplicial complex.*

Because $\delta_{\mathcal{Q}} \leq \delta_{\mathcal{P}}$ for $\mathcal{Q} \supseteq \mathcal{P}$, it follows that if $\mathrm{sys}(\mathbb{M}) > 2\delta_{\mathcal{P}}$ for some set $\mathcal{P}$, then the Delaunay triangulation of any superset of $\mathcal{P}$ is a simplicial complex as well. This makes the incremental algorithm work in this case.

3 Systole of symmetric hyperbolic surfaces

Recall that $\mathbb{M}_g$ denotes the symmetric hyperbolic surface of genus g. As mentioned before, to be able to verify the inequality $\mathrm{sys}(\mathbb{M}_g) > 2\delta_{\mathcal{P}}$, we have to know the value of $\mathrm{sys}(\mathbb{M}_g)$. This value is given in the following theorem.

▶ **Theorem 3.1.** *The systole of the surface $\mathbb{M}_g$ corresponding to the regular $4g$-gon satisfies*

$$\cosh(\tfrac{1}{2}\,\mathrm{sys}(\mathbb{M}_g)) = 1 + 2\cos(\tfrac{\pi}{2g}).$$

In the proof, we first show that there exists a closed, non-contractible curve with the stated length. To prove that there are no shorter such curves, we represent each closed geodesic on the surface $\mathbb{M}_g$ as a sequence of hyperbolic line segments between sides of the regular $4g$-gon F_g (see Figure 4) and analyze the different configurations of sequences of segments.

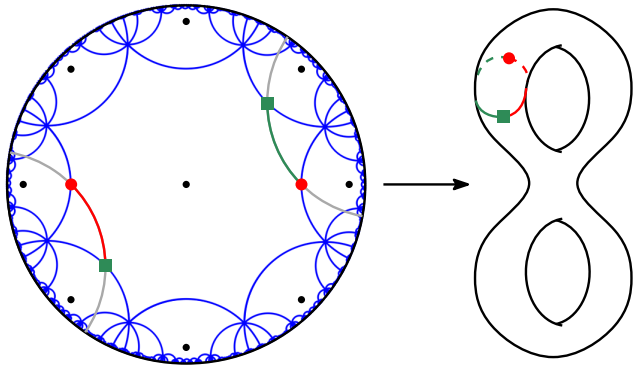


Figure 4 Representation of a systole of $\mathbb{M}_g$ as a sequence of segments between sides

4 Representation of Delaunay triangulations

In Section 2.2 we considered the Delaunay triangulation of $\Gamma_g\mathcal{P}$. However, practically speaking it is not possible to work with triangulations of point sets with infinitely many points. For this reason, let $D_\mathcal{N}$ be the union of translates of F_g, that share are least one point with F_g (see Figure 5).

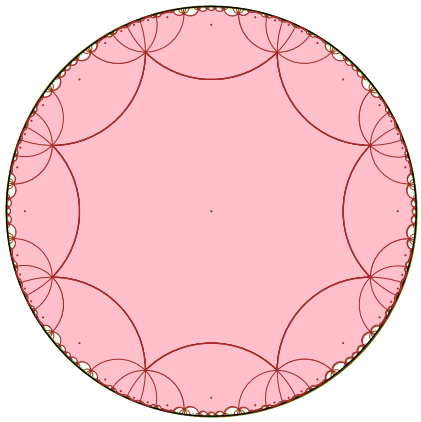


Figure 5 The union $D_\mathcal{N}$ of neighboring regions for the Bolza surface

Now, the next proposition states that it is sufficient to consider the combinatorics of the Delaunay triangulation of the points inside $D_\mathcal{N}$ if the inequality $\mathrm{sys}(\mathbb{M}_g) > 2\delta_\mathcal{P}$ is satisfied.

▶ **Proposition 4.1.** *Assume that $\mathcal{P}$ satisfies $\mathrm{sys}(\mathbb{M}_g) > 2\delta_\mathcal{P}$. Let Δ be a triangle in $\mathrm{DT}_\mathbb{H}(\Gamma_g\mathcal{P})$. If $\Delta \cap F_g \neq \emptyset$, then $\Delta \subset D_\mathcal{N}$.*

In other words, as soon as the condition $\mathrm{sys}(\mathbb{M}_g) > 2\delta_\mathcal{P}$ is satisfied, it suffices to compute the Delaunay triangulation of the finite point set $\Gamma_g\mathcal{P} \cap D_\mathcal{N}$ instead of the Delaunay triangulation of $\Gamma_g\mathcal{P}$. As we mentioned before, a dummy point set $\mathcal{Q}$ is used to guarantee that the condition holds. However, we cannot use Proposition 4.1 to find the Delaunay triangulation of the dummy point set. Instead we will use the following proposition. Here, $\mathcal{Q}_0$ denotes the set consisting of the origin, the vertex and the midpoints of the sides of F_g.

▶ **Proposition 4.2.** *Assume that $\mathcal{Q} \supseteq \mathcal{Q}_0$. Let Δ be a triangle in $\mathrm{DT}_\mathbb{H}(\Gamma_g\mathcal{Q})$. If $\Delta \cap F_g \neq \emptyset$, then $\Delta \subset D_\mathcal{N}$.*

Now, if we construct the dummy point set $\mathcal{Q}$ in such a way that it contains $\mathcal{Q}_0$, then we can find the Delaunay triangulation of $\mathcal{Q}$ by considering the Delaunay triangulation of $\Gamma_g\mathcal{Q} \cap D_{\mathcal{N}}$.

5 Initialization

In this section we will present an algorithm to compute a dummy point set, i.e., a finite point set $\mathcal{Q}$ that satisfies $\text{sys}(\mathbb{M}_g) > 2\delta_{\mathcal{Q}}$. The idea of this algorithm is similar to Delaunay refinement, also known as Ruppert's algorithm [10]. This works as follows. Initially, the dummy point set $\mathcal{Q}$ will only contain $\mathcal{Q}_0$. Then we consider the Delaunay triangulation $\text{DT}_{\mathbb{H}}(\Gamma_g\mathcal{Q} \cap D_{\mathcal{N}})$ of $\Gamma_g\mathcal{Q} \cap D_{\mathcal{N}}$ in $\mathbb{H}^2$. If there is a triangle in this triangulation with circumradius at least $\frac{1}{2}\,\text{sys}(\mathbb{M}_g)$, which has a non-empty intersection with the fundamental polygon, then the circumcenter of this triangle is added to $\mathcal{Q}$. This process continues until there are no more such triangles. See Figure 6 for an application of the algorithm to the symmetric hyperbolic surface of genus 3. A more formal description can be found below.

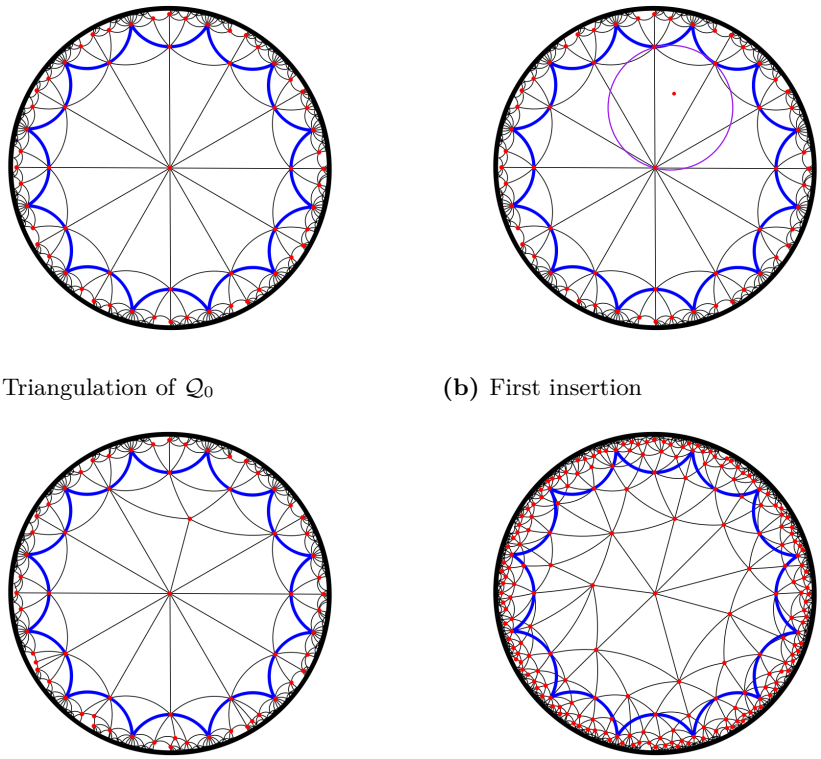
Algorithm 1: Dummy point algorithm

Input : hyperbolic surface $\mathbb{M}_g$
Output: finite point set $\mathcal{Q} \subset \mathbb{M}_g$ such that $\text{sys}(\mathbb{M}_g) > 2\delta_{\mathcal{Q}}$

```
Initialize: let Q = Q_0.
Compute DT_H(Γ_g Q ∩ D_N).
while there exists a triangle Δ in DT_H(Γ_g Q ∩ D_N) with circumdiameter at least
  (1/2) sys(M_g) and Δ ∩ F_g ≠ ∅ do
    Add the circumcenter of Δ to Q
    Update DT_H(Γ_g Q ∩ D_N)
end
```

▶ **Theorem 5.1.** *The dummy point algorithm terminates. The resulting dummy point set $\mathcal{Q}$ satisfies $\text{sys}(\mathbb{M}_g) > 2\delta_{\mathcal{Q}}$ and has cardinality of order $\Theta(g)$.*

To give an idea of the proof that the number of iterations of the **while** loop is of order $\Theta(g)$ (and hence, that the algorithm terminates), we observe that the distance between every pair of points in $\mathcal{Q}$ is at least $\frac{1}{4}\,\text{sys}(\mathbb{M}_g)$. It follows that we can construct a circle packing on $\mathbb{M}_g$ with circles of radius $\frac{1}{8}\,\text{sys}(\mathbb{M}_g)$ centered at the points in $\mathcal{Q}$. Since the area of $\mathbb{M}_g$ is of order $\Theta(g)$, the above claims follows. Finally, we note that the size complexity of the resulting dummy point set is asymptotically optimal [7, Prop. 9.1].



(a) Triangulation of $\mathcal{Q}_0$

(b) First insertion

(c) After first insertion

(d) After last insertion

Figure 6 Several steps in the dummy point algorithm

References

1 Alan F. Beardon. *The geometry of discrete groups*, volume 91 of *Graduate Texts in Mathematics*. Springer-Verlag, 2012.

2 Mikhail Bogdanov, Monique Teillaud, and Gert Vegter. Delaunay triangulations on orientable surfaces of low genus. In Leibniz International Proceedings in Informatics, editor, *Proceedings of the Thirty-second International Symposium on Computational Geometry (SoCG 2016)*, pages 20:1–20:17, 2016. doi:10.4230/LIPIcs.SoCG.2016.20.

3 A. Bowyer. Computing Dirichlet tessellations. *The Computer Journal*, 24(2):162–166, 1981. doi:10.1093/comjnl/24.2.162.

4 Peter Buser. *Geometry and spectra of compact Riemann surfaces*. Springer-Verlag, 2010.

5 Manuel Caroli, Aymeric Pellé, Mael Rouxel-Labbé, and Monique Teillaud. 3D periodic triangulations. In CGAL Editorial Board, editor, *CGAL User and Reference Manual*. 4.13 edition, 2018. URL: http://doc.cgal.org/latest/Manual/packages.html#PkgPeriodic3Triangulation3Summary.

6 Manuel Caroli and Monique Teillaud. Delaunay triangulations of closed Euclidean d-orbifolds. *Discrete & Computational Geometry*, 55(4):827–853, 2016. URL: https://hal.inria.fr/hal-01294409, doi:10.1007/s00454-016-9782-6.

7 Matthijs Ebbens. Delaunay triangulations on hyperbolic surfaces. Master's thesis, University of Groningen, 2017. URL: http://fse.studenttheses.ub.rug.nl/id/eprint/15727.

8 Iordan Iordanov and Monique Teillaud. Implementing Delaunay triangulations of the Bolza surface. In Leibniz International Proceedings in Informatics, editor, *Proceedings of the Thirty-third International Symposium on Computational Geometry (SoCG 2017)*, pages 44:1–44:15, 2017. doi:10.4230/LIPIcs.SoCG.2017.44.

9 Clément Jamin, Sylvain Pion, and Monique Teillaud. 3D triangulations. In CGAL Editorial Board, editor, *CGAL User and Reference Manual*. 4.13 edition, 2018. URL: http://doc.cgal.org/latest/Manual/packages.html#PkgTriangulation3Summary.

10 Jim Ruppert. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *Journal of algorithms*, 18(3):548–585, 1995.

11 John Stillwell. *Geometry of surfaces*. Springer-Verlag, 1992.

Computing the Straight Skeleton of an Orthogonal Monotone Polygon in Linear Time*

Günther Eder[1], Martin Held[1], and Peter Palfrader[1]

1 **Universität Salzburg, FB Computerwissenschaften, 5020 Salzburg, Austria**
{geder,held,palfrader}@cs.sbg.ac.at

Abstract

We introduce a simple algorithm to construct the straight skeleton of an n-vertex orthogonal monotone polygon in optimal $\mathcal{O}(n)$ time and space.

1 Introduction

The straight skeleton $\mathcal{S}(\mathcal{P})$ of a simple polygon $\mathcal{P}$ was introduced by Aichholzer et al. [2]. It is the result of a wavefront-propagation process where the edges of $\mathcal{P}$ move inwards at unit speed in a self-parallel manner, forming one or more wavefront polygons whose combinatorics change when wavefront edges collapse or wavefront vertices move into other parts of the wavefront. The straight skeleton is the trace of the vertices of these wavefront polygons over their propagation, cf. Figure 1.

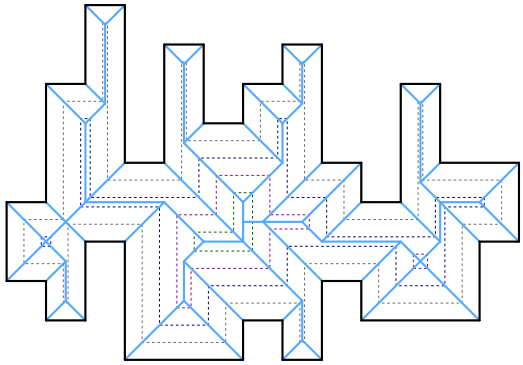


Figure 1 A polygon (black) with its straight skeleton (blue) and some wavefronts (dotted).

The currently best known algorithm for constructing the straight skeleton of unrestricted input is by Eppstein and Erickson [5] and runs in $\mathcal{O}(n^{17/11+\varepsilon})$ time and space for an n-vertex polygon and any $\varepsilon > 0$. In the case of a convex input polygon, the straight skeleton coincides with the medial axis and can be computed in linear time [1]. For monotone polygons, Biedl et al. [3] present an algorithm to compute the straight skeleton in $\mathcal{O}(n \log n)$ time.

In this work we show that an approach which is similar to that of Biedl et al. [3] makes it possible to construct $\mathcal{S}(\mathcal{P})$ in optimal linear time if $\mathcal{P}$ is monotone and orthogonal: We also construct the straight skeleton for each of the two monotone chains of $\mathcal{P}$ separately and then merge them to obtain $\mathcal{S}(\mathcal{P})$. Since $\mathcal{P}$ is orthogonal, $\mathcal{S}(\mathcal{P})$ coincides with the Voronoi diagram of $\mathcal{P}$ in the L_∞-norm [2]. Papadopoulou shows how to compute the L_∞-norm Voronoi diagram of orthogonal planar straight-line graphs in $\mathcal{O}(n \log n)$ time [6].

* Work supported by Austrian Science Fund (FWF): Grant ORD 53-VO.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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

Let $\mathcal{P}$ be an x-monotone, axis-aligned orthogonal polygon. For the sake of descriptive simplicity we assume that $\mathcal{P}$ has no vertex with interior angle equal to π. (Our algorithm can be extended to handle such input at no additional computational cost.) Let $\mathcal{C}_l$ and $\mathcal{C}_u$ denote the lower and upper monotone chain of $\mathcal{P}$, respectively. As the leftmost and rightmost edges of $\mathcal{P}$ are vertical, we arbitrarily assign the leftmost edge to $\mathcal{C}_l$ and the rightmost edge to $\mathcal{C}_u$.

For an edge e of $\mathcal{P}$, denote by $I(e)$ the half-plane induced by the supporting line $\ell(e)$ of e which locally (close to e) overlaps with the interior of $\mathcal{P}$. For two non-parallel edges e_i and e_j of the polygon, the bisector $b_{i,j}$ is the ray lying on the angular bisector of the supporting lines of e_i and e_j within the common interior region $I(e_i) \cap I(e_j)$. If the edges e_i, e_j are parallel then we use their wavefront edges $e_i(t)$ and $e_j(t)$ to build $b_{i,j}$. (But we will still refer to $b_{i,j}$ as the bisector of e_i and e_j.) If the wavefront edges overlap at a specific time t' then $b_{i,j}$ is the segment formed by the non-empty intersection $e_i(t') \cap e_j(t')$. Otherwise, if $e_i(t')$ and $e_j(t')$ share a common end-point p then $b_{i,j}$ is the ray perpendicular to them that starts at p and lies in the common interior $I(e_i) \cap I(e_j)$. A wavefront vertex that moves along such a bisector is called a *ghost vertex* [4], and we call the resulting straight-skeleton arc a *ghost arc*. It is *unfinished* if its extent is not yet known.

Let $\mathcal{C}$ be an x-monotone polygonal chain. For $e_i \in \mathcal{C}$, let Π_i denote the portion of $I(e_i)$ that is incident at e_i and limited by the two bisectors through the endpoints of e_i. We call Π_i the *half-plane slab* of e_i.

▶ **Lemma 1.1.** *Every face of $\mathcal{S}(\mathcal{C})$ is monotone with respect to its defining edge and is also monotone with respect to a line perpendicular to its defining edge.*

▶ **Corollary 1.2.** *Every face of $\mathcal{S}(\mathcal{C})$ is x-monotone.*

▶ **Lemma 1.3.** *For every edge e_i of $\mathcal{C}$ the straight-skeleton face $f(e_i)$ incident at e_i lies inside of the half-plane slab Π_i.*

2 Computing the Straight Skeleton of a Single Chain

In order to compute the straight skeleton of a polygon, we first construct the straight skeletons of its lower and upper chains individually, and then we merge them; cf. Section 3. We start with describing the construction of the skeleton $\mathcal{S}(\mathcal{C}_l)$ of the lower chain $\mathcal{C}_l := e_1, \ldots, e_{n'}$. (The upper chain $\mathcal{C}_u$ is processed in an identical fashion.) If we extend the leftmost edge e_1 and rightmost edge $e_{n'}$ of $\mathcal{C}_l$ to infinity then the plane is split into two areas. The area which contains $\mathcal{P}$ is tessellated by $\mathcal{S}(\mathcal{C}_l)$ into straight-skeleton faces, with one face $f(e_i)$ being incident at each input edge e_i. As $f(e_i)$ is monotone with respect to the normal of e_i (Lemma 1.1), we can meaningfully split the arcs bounding $f(e_i)$ into a left and a right chain. Each chain is a list of arcs (edges and potentially one ray) that starts in a vertex of e_i and either ends in a ray for unbounded faces or ends when it meets the other chain. If the final arc of a chain is parallel to e_i then it can be assigned arbitrarily to the left or the right chain.

We construct the straight skeleton of $\mathcal{C}_l$ incrementally. As we insert edges of $\mathcal{C}_l$ from left to right, we maintain a partial straight skeleton $\mathcal{S}^*$. We store in $\mathcal{S}^*$ for each input edge e_i the left chain of $f(e_i)$ as a list of arcs. Additionally, $\mathcal{S}^*$ maintains a stack `R` of edges whose faces have a left chain that terminates in a ray and another stack `G` of edges which have faces whose left chain terminates in an unfinished ghost arc, a vertical edge where the second endpoint is not yet known. Figure 2 (left) shows blue and purple rays of `R` and `G`, respectively. Initially, `R` contains e_1, which does not have a left chain as it extends to infinity itself.

Once an edge e_i has been inserted into $\mathcal{S}^*$, the left chain of $f(e_i)$ in $\mathcal{S}^*$ can be modified only in two specific ways: If the left chain of $f(e_i)$ ends in a ray, this ray may be replaced by a bounded segment. If the left chain of $f(e_i)$ ends in an unfinished ghost arc, this arc may be replaced by a bounded, finished ghost arc, or it may be replaced by a bounded, finished ghost arc followed by another bounded segment. In Figure 2, the ray $b_{2,3}$ is replaced by a bounded segment and the ghost arc a' is terminated at the intersection with arc a. As we

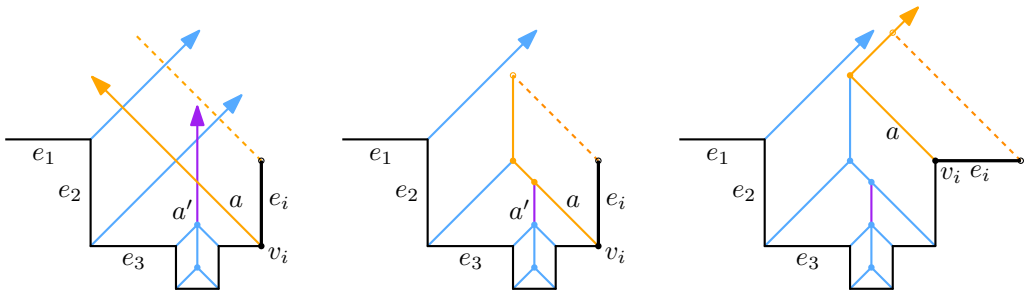


Figure 2 (Left, Center) Inserting edge e_i; (Right) Incrementing i and adding the next edge e_i; $\mathcal{S}^*$, including rays, in blue; (Unfinished) Ghost arcs in purple, and arcs added due to e_i in orange.

insert edge e_i, we have to build the left chain of its face. We do this iteratively, starting at the left vertex of e_i. This chain starts with an arc that lies on a bisector with direction vector either $\binom{-1}{1}$ or $\binom{1}{1}$. We continue to append arc segments, starting each arc where the previous segment terminated. We stop when we append a ghost arc (which we will complete later), when we append a ray, or in some cases when we appended a bounded (vertical) arc segment. Note that all arcs lie on bisectors with direction vectors $\binom{-1}{1}$, $\binom{0}{1}$, or $\binom{1}{1}$. The direction of the initial arc segment is given by the bisector of e_i and its predecessor e_{i-1}. For all subsequent arcs, the direction depends on e_i and the edge on top of the stack `R`.

▶ **Lemma 2.1.** *No arc inserted into $\mathcal{S}^*$ with direction $\binom{1}{1}$ can intersect any arc of $\mathcal{S}^*$ and, thus, is a ray which escapes to infinity.*

Arcs with Direction $\binom{1}{1}$ Therefore, if the left chain of $f(e_i)$ ever includes an arc on a $\binom{1}{1}$-bisector, this arc will be a $\binom{1}{1}$-ray because nothing in the current $\mathcal{S}^*$ can intersect it. Thus, this arc terminates the left chain. We add e_i with this chain to $\mathcal{S}^*$ and also put e_i on the corresponding stack `R`.

Arcs with Direction $\binom{-1}{1}$ If the arc a to be added to the left chain of $f(e_i)$ lies on a $\binom{-1}{1}$-bisector, then this arc may be a ray but it may also be bounded and interact with faces previously inserted. If this is the first arc of the left chain of $f(e_i)$ then we check whether the previous face, $f(e_{i-1})$, has its left chain already completed, terminating in a bounded segment. If this is true then a is an arc from the left vertex of e_i along the bisector to the end of the previous face's chain. Otherwise, or if this is not the first arc of the left chain, we look at the top of our stack `R`. Let e_t be the edge on top of `R`. If the left chain of $f(e_t)$ does not terminate in a $\binom{1}{1}$-ray, then there is no $\binom{1}{1}$-ray in $\mathcal{S}^*$, and a therefore is a $\binom{-1}{1}$-ray which finishes the left chain of this face. An argument similar to the one used in Lemma 2.1, now applied to $\binom{-1}{1}$-direction, shows that a cannot be intersected by any new arc. If `G` is not empty then all unfinished ghost arcs on that stack get turned into finished ghost-arcs which terminate at their intersection points with a. If, however, the left chain of $f(e_t)$ terminates in a $\binom{1}{1}$-ray r then a will intersect r in a point p. We modify $f(e_t)$ by replacing r with a line

segment r' terminating at p, and we remove e_t from R. Furthermore a is a line segment that terminates at p. Lastly, we have to process any elements on G that had been inserted after e_t was processed, as these ghost arcs lie below r' and a. These get popped from G and their unfinished ghost arcs are replaced with arcs terminating where they intersect a or r'.

Arcs with Direction $\binom{0}{1}$ The last possible direction that an arc a of the left chain of e_i may have is $\binom{0}{1}$. Let p be the point where the previous arc ended. Then a is either a ghost arc or a standard vertical arc. In the first case we push e_i onto G. We also store a reference to the edge on top of R at this time, and thereby complete the processing of $f(e_i)$. The latter case is the result of two vertical input segments whose wavefront segments meet. Let e_t be the edge at the top of R. Then a is the line segment from p that is contained in both Π_t and Π_i. If the extent of a is limited by Π_i only, then this finishes the construction of the left chain of $f(e_i)$. Otherwise, the ray of the left chain of $f(e_t)$ touches the end-point of a. We replace that ray with a bounded segment terminating at this intersection and pop e_t from R. If the extent of a was limited by Π_t only, we continue with constructing the next arc of the left chain of $f(e_i)$ whose direction is determined by e_i and the edge now on top of R. Otherwise, if a was limited by both Π_t and Π_i, the construction of the left chain of $f(e_i)$ is also finished.

▶ **Lemma 2.2.** *All arcs created by inserting e_i intersect only rays or ghost arcs of $\mathcal{S}^*$.*

Finalizing $\mathcal{S}(\mathcal{C}_l)$ Once all input edges have been inserted into $\mathcal{S}^*$, we may still have elements on the stack G. For every element on G, we replace the unfinished ghost arc with a segment of finite length that terminates at the $\binom{1}{1}$-ray which it intersects first. We know where to look since we stored a reference to the correct face when we pushed an element onto G. If there is no $\binom{1}{1}$-ray (because the reference was to e_1), then these ghost arcs are finalized as vertical rays that go to infinity. The resulting structure $\mathcal{S}^*$ is now the straight skeleton $\mathcal{S}(\mathcal{C}_l)$.

▶ **Theorem 2.3.** *Our incremental construction computes the straight skeleton of a monotone orthogonal chain of n vertices in $\mathcal{O}(n)$ time and space.*

3 Merging $\mathcal{S}(\mathcal{C}_l)$ and $\mathcal{S}(\mathcal{C}_u)$ into $\mathcal{S}(\mathcal{P})$

While merging the two skeletons we create a polygonal merge chain $\mathcal{M} := a_1, \ldots, a_m$. This chain connects the first (western) vertex v_W of $\mathcal{C}_l$ and $\mathcal{C}_u$ to their last (eastern) vertex v_E. This merge is similar to the algorithm used for merging Voronoi diagrams of point sites [7].

▶ **Lemma 3.1** (Lemmas 4 and 5 in Biedl et al. [4])**.** *The polygonal chain $\mathcal{M}$ created by merging $\mathcal{S}(\mathcal{C}_u)$ and $\mathcal{S}(\mathcal{C}_l)$ is x-monotone.*

We use the notion of a bisector between two faces and thereby relate to the bisector between the two edges that define these faces. Let $f_l(i)$ and $f_u(j)$ denote the i-th and j-th face of $\mathcal{S}(\mathcal{C}_l)$ and $\mathcal{S}(\mathcal{C}_u)$, ordered from left to right along each chain, where $0 < i < n'$ and $0 < j < n''$. Clearly, every arc of $\mathcal{M}$ is a portion of a bisector between two such faces, one from $\mathcal{S}(\mathcal{C}_l)$ and one from $\mathcal{S}(\mathcal{C}_u)$; cf. Figure 3 (Left).

We start at the first vertex $p := v_W$ and look at the bisector b between $f_l(1)$ and $f_u(1)$. It starts at p. Let p' denote the intersection closest to p between b and $\mathcal{S}(\mathcal{C}_l)$ as well as b and $\mathcal{S}(\mathcal{C}_u)$. W.l.o.g., we assume that p' is formed between b and an arc a of $\mathcal{S}(\mathcal{C}_l)$. Let $f_l(i)$ denote the second face incident at a, with $1 < i \leq n'$. Then p' forms a node in $\mathcal{S}(\mathcal{P})$ that has the same distance to the edges of $f_u(1)$, $f_l(1)$, and $f_l(i)$. Thus, we let a end at p' and add an arc $a_1 := \overline{pp'}$ to $\mathcal{M}$. This arc also forms an arc in $\mathcal{S}(\mathcal{P})$. For the next incremental

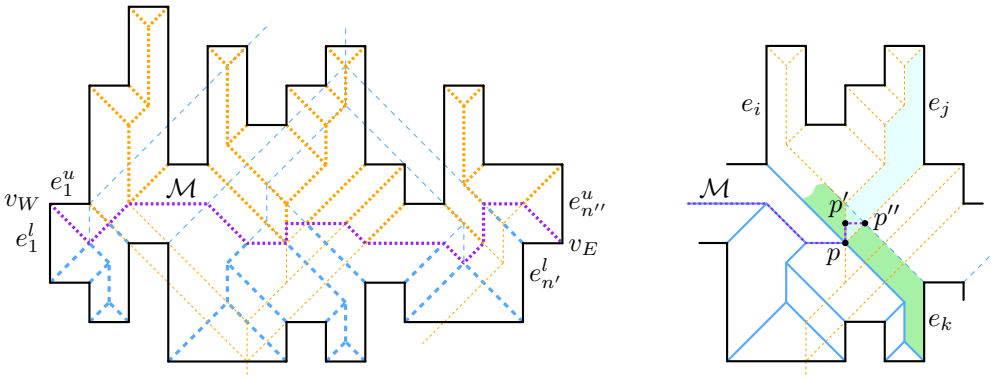


Figure 3 (Left) Monotone orthogonal polygon (black) with the upper (orange) and lower (blue) partial skeleton and the merge line $\mathcal{M}$ (purple); (Right) The merge step creates a vertical arc $\overline{pp'}$.

step, let $p := p'$ and let b denote the bisector between $f_u(1)$ and $f_l(i)$. It starts at p. Again we find the next intersection p' closest to p of b with both skeletons that does not lie left of p. We repeat this process until we arrive at the last vertex v_E.

Note that the intersection between b and one of the skeletons can form a vertical line segment instead of a single point if b coincides with a vertical skeleton arc; cf. Figure 3 (Right). To find the next node p' in this case we have to look at the relevant faces of $\mathcal{S}(\mathcal{C}_l)$ and $\mathcal{S}(\mathcal{C}_u)$. Let s denote the vertical segment formed by such an intersection and let a denote the vertical arc that is intersected. W.l.o.g., we assume that a belongs to $\mathcal{S}(\mathcal{C}_u)$. Let $f_u(i)$ and $f_u(j)$ denote the two faces incident at a. Let $f_l(k)$ together with $f_u(i)$ define b. Thus, the next bisector b' that starts on some point on s is defined by $f_u(j)$ and $f_l(k)$. We observe that both e_j and e_k must be vertical and have the same distance to s, since the bisector between e_i, e_j and between e_i, e_k lies on a common line. Hence, e_j and e_k lie on a common supporting line and their faces in the upper and lower skeleton contain s. We can infer that the wavefront edges $e_j(t)$ and $e_k(t)$ must become adjacent at some point p''. At that point a ghost vertex traces out a horizontal arc. Since we are in the process of merging the two skeletons, this arc is not yet present in either of the two skeletons. Both faces, $f_u(j)$ and $f_l(k)$, are x-monotone. To find p'' we start at their defining input edge and walk along their boundary until we find that intersection. A horizontal line through p'' intersects s and defines the node p' sought.

Complexity of the Merge At every step a pair of faces $f_l(i)$ and $f_u(j)$ contributes a single arc to $\mathcal{M}$. Upon insertion of this arc we increase at least one of the two indices i, j. Thus, $\mathcal{M}$ has size at most $n' + n''$, which is equal to n. It remains to discuss how to find the next intersection p' efficiently. A new bisector b, defined by the faces $f_l(i)$ and $f_u(j)$, starts at the known point p. Both faces are x-monotone; cf. Corollary 1.2. The monotonicity of a face also holds after the merge since $\mathcal{M}$ is x-monotone as well. We use the x-coordinate of the vertex that traces out b to walk along the boundary of both faces. When an intersection is found we add a node, cross to the neighboring face, and start a new arc. Since $\mathcal{M}$ is x-monotone we can simply continue traversing the face whose boundary was not intersected. Therefore, we traverse every arc of both $\mathcal{S}(\mathcal{C}_l)$ and $\mathcal{S}(\mathcal{C}_u)$ at most once.

▶ **Theorem 3.2.** *The merge process merges $\mathcal{S}(\mathcal{C}_l)$ and $\mathcal{S}(\mathcal{C}_u)$ and obtains $\mathcal{S}(\mathcal{P})$ for an n-vertex monotone orthogonal polygon $\mathcal{P}$ in $\mathcal{O}(n)$ time.*

References

1 Alok Aggarwal, Leonidas J. Guibas, James Saxe, and Peter W. Shor. A Linear-Time Algorithm for Computing the Voronoi Diagram of a Convex Polygon. *Discrete & Computational Geometry*, 4(6):591–604, 1989. doi:10.1007/BF02187749.

2 Oswin Aichholzer, Franz Aurenhammer, David Alberts, and Bernd Gärtner. A Novel Type of Skeleton for Polygons. *Journal of Universal Computer Science*, 1(12):752–761, 1995. doi:10.1007/978-3-642-80350-5_65.

3 Therese Biedl, Martin Held, Stefan Huber, Dominik Kaaser, and Peter Palfrader. A Simple Algorithm for Computing Positively Weighted Straight Skeletons of Monotone Polygons. *Information Processing Letters*, 115(2):243–247, February 2015. doi:10.1016/j.ipl.2014.09.021.

4 Therese Biedl, Martin Held, Stefan Huber, Dominik Kaaser, and Peter Palfrader. Weighted Straight Skeletons in the Plane. *Computational Geometry: Theory and Applications*, 48(2):120–133, 2015. doi:10.1016/j.comgeo.2014.08.006.

5 David Eppstein and Jeff 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. doi:10.1145/276884.276891.

6 Evanthia Papadopoulou. Critical Area Computation for Missing Material Defects in VLSI Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(5):583–597, May 2001. doi:10.1109/43.920683.

7 Michael I. Shamos and Dan Hoey. Closest-Point Problems. In *Foundations of Computer Science, 1975, 16th Annual Symposium on*, pages 151–162. IEEE, October 1975. doi:10.1109/sfcs.1975.8.

Maximum Rectilinear Convex Subsets *

Hernán González-Aguilar[1], David Orden[2], Pablo Pérez-Lantero[3], David Rappaport[4], Carlos Seara[5], Javier Tejel[6], and Jorge Urrutia[7]

1 **Facultad de Ciencias, Universidad Autónoma de San Luis Potosí, Mexico**
`hernan@fc.uaslp.mx`
2 **Departamento de Física y Matemáticas, Universidad de Alcalá, Spain**
`david.orden@uah.es`
3 **Departamento de Matemática y Ciencia de la Computación, Universidad de Santiago de Chile, Chile**
`pablo.perez.l@usach.cl`
4 **School of Computing, Queen's University, Canada**
`daver@cs.queensu.ca`
5 **Departament de Matemàtiques, Universitat Politècnica de Catalunya, Spain**
`carlos.seara@upc.edu`
6 **Departamento de Métodos Estadísticos, IUMA, Universidad de Zaragoza, Spain**
`jtejel@unizar.es`
7 **Instituto de Matemáticas, Universidad Nacional Autónoma de México, Mexico**
`urrutia@matem.unam.mx`

Abstract

Let P be a set of n points in the plane. We consider a variation of the classical Erdős-Szekeres problem, presenting efficient algorithms with $O(n^3)$ running time and $O(n^2)$ space which compute: (1) A subset S of P such that the boundary of the rectilinear convex hull of S has the maximum number of points from P, (2) a subset S of P such that the boundary of the rectilinear convex hull of S has the maximum number of points from P and its interior contains no element of P, and (3) a subset S of P such that the rectilinear convex hull of S has maximum area and its interior contains no element of P.

1 Introduction

Let P be a point set in general position in the plane. A subset S of P with k points is called a *convex k-gon* if the elements of S are the vertices of a convex polygon, and it is called a *convex k-hole* if the interior of the convex hull of S contains no point of P. The study of convex k-gons and convex k-holes of point sets started in a seminal paper by Erdős and Szekeres [11]. Since then, a plethora of papers studying both the combinatorial and the algorithmic aspects of convex k-gons and convex k-holes has been published. The reader can consult the two survey papers [8, 12] about so-called Erdős-Szekeres type problems.

Some recent papers studying the existence and the number of convex k-gons and convex k-holes for sets of points in the plane are [1, 2, 3]. Papers dealing with the problem of finding

* 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. D. O. is supported by project MTM2017-83750-P of the Spanish Ministry of Science (AEI/FEDER, UE). P. P-L. is supported by project CONICYT FONDECYT/Regular 1160543 (Chile). D. R. is supported by NSERC of Canada Discovery Grant RGPIN/06662-2015. C. S. is supported by projects Gen. Cat. DGR 2017SGR1640 and MINECO MTM2015-63791-R. J. T. is supported by project MTM2015-63791-R MINECO/FEDER. J. U. is supported by PAPIIT grant IN102117 from UNAM.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 19–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

largest convex k-gons and convex k-holes are, respectively, Chvátal and Kincsek [10] and Avis and Rappaport [7], which solve these problems in $O(n^3)$ time and $O(n^2)$ space.

In this paper we study Erdős-Szekeres type problems under a variation of convexity known as *rectilinear convexity* (or *orthoconvexity*): Let $P = \{p_1, \ldots, p_n\}$ be a n point set in the plane in general position. A *quadrant* is the intersection of two open half-planes whose supporting lines are parallel to the x- and y-axes. We say that a quadrant Q is *P-free* if it contains no point of P. The *rectilinear convex hull* (or *orthogonal convex hull*) of P, denoted as $RCH(P)$, was introduced by Ottmann et al. [13] (see also [15]) and is defined as:

$$RCH(P) = \mathbb{R}^2 - \bigcup_{Q \text{ is } P\text{-free}} Q.$$

The rectilinear convex hull of a point set might be a simply connected set, yielding an intuitive and appealing structure (see Figure 1, left). However, in other cases the rectilinear convex hull can have several connected components (see Figure 1, right), some of which might be single points which we call *pinched* points. The *size* of $RCH(S)$ is the number of points of S on the boundary of $RCH(S)$. The sizes of the rectilinear convex hulls in Figure 1 are thirteen and twelve. In this paper, we present algorithms for the following problems:

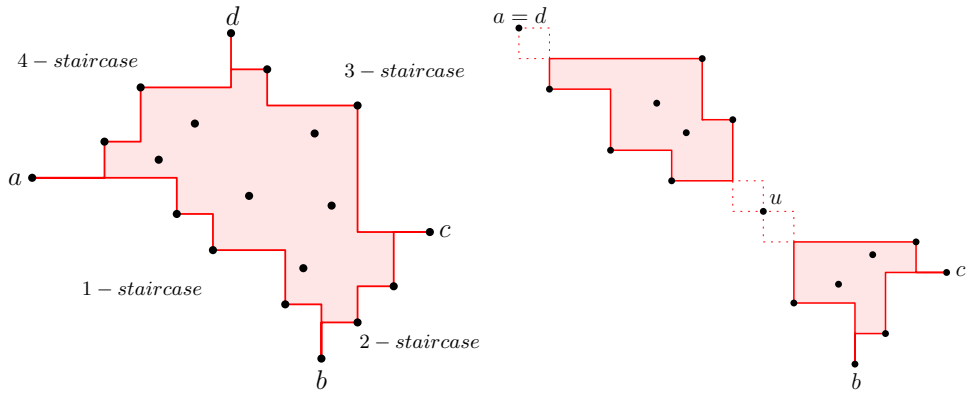


Figure 1 Left: A point set with a connected rectilinear convex hull. Right: A point set whose rectilinear convex hull is disconnected. Two of its components are pinched points.

(1) MaxRCH: Given a set P of n points in the plane, find a subset $S \subseteq P$ such that the size of $RCH(S)$ is maximized. We solve the MaxRCH problem given an algorithm which runs in $O(n^3)$-time and $O(n^2)$-space. Then, we adapt our algorithm to solve the following problems, each in $O(n^3)$ time and $O(n^2)$ space.
(2) MaxEmptyRCH: Given a set P of n points in the plane, find a subset $S \subseteq P$ such that the interior of $RCH(S)$ contains no point of P and the size of $RCH(S)$ is maximized.
(3) MaxAreaRCH: Given a set P of n points in the plane, find a subset $S \subseteq P$ such that the interior of $RCH(S)$ contains no point of P and the area of $RCH(S)$ is maximized.

Related work: Erdős-Szekeres type problems have also been studied for colored point sets. Let P be a set of points such that each of its elements is assigned a color, say red or blue. Bautista-Santiago et al. [9] studied the problem of finding a monochromatic subset S of P of maximum size such that all of the elements of P contained in the convex hull of S have the same color. They also solve the same problem for each element of P having a weight. All of these algorithms run in $O(n^3)$ time and $O(n^2)$ space.

Notation and definitions: For a point p of the plane, let p_x and p_y denote the x- and y-coordinates of p, respectively. For $p \neq q \in \mathbb{R}^2$, we write $p \prec q$ to denote that $p_x < q_x$ and

$p_y < q_y$, and $p \prec' q$ to denote that $p_x < q_x$ and $p_y > q_y$. Any point p in the plane defines four axis-aligned quadrants $Q_i(p)$, $i = 1, 2, 3, 4$ as follows (see Figure 2, left):

- $Q_2(p) = \{q \in \mathbb{R}^2 \mid q \prec' p\}$, $\quad Q_1(p) = \{q \in \mathbb{R}^2 \mid p \prec q\}$,
- $Q_3(p) = \{q \in \mathbb{R}^2 \mid q \prec p\}$, $\quad Q_4(p) = \{q \in \mathbb{R}^2 \mid p \prec' q\}$.

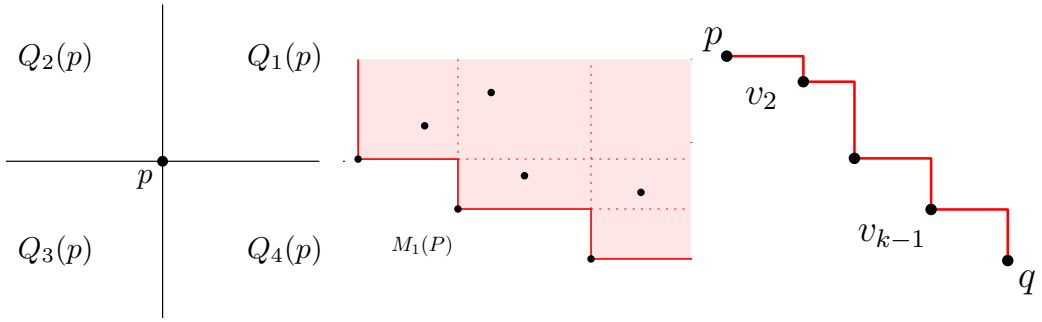


Figure 2 Left: The definition of the sets $Q_i(p)$. Middle: A 7-point set P and the set $M_1(P)$. The vertices of $M_1(P)$ in P are the 1-extremal points of P. Right: A 1-staircase.

Given P, for $i = 1, 2, 3, 4$, let $M_i(P) = \bigcup_{p \in P} \overline{Q_i(p)}$ where $\overline{Q_i(p)}$ denotes the closure of $Q_i(p)$. The elements of P that belong to the boundary of $M_i(P)$, are called the (rectilinear) *i-extremal* points of P (see Figure 2, middle). For every $J \subseteq \{1, 2, 3, 4\}$, we say that $p \in P$ is *J-extremal* if p is j-extremal for every $j \in J$. The *rectilinear convex hull* of P is the set[1]

$$RCH(P) = \bigcap_{i=1}^{4} M_i(P),$$

see Figure 1, left. For the sake of simplicity, we assume that all point sets P considered in this paper are in *general position*, which here means that no two points of P share the same x- or y-coordinate. Let a, b, c, d denote the leftmost, bottommost, rightmost, and topmost points of P, respectively. Note that a is $\{1, 4\}$-extremal, b is $\{1, 2\}$-extremal, c is $\{2, 3\}$-extremal, and d is $\{3, 4\}$-extremal.

i-staircases: Let $S = \{v_1, \ldots, v_k\}$ be a set of vertices such that $v_1 = p$, $v_k = q$, and $v_i \prec' v_j$ for every $i < j$. A 1-*staircase* joining p to q (Figure 2, right) is the boundary of $M_1(S)$ minus the infinite rays (Figure 2, middle), i.e., it is an orthogonal polygonal chain such that two consecutive points of S are joined by an *elbow*, i.e., a horizontal followed by a vertical segment. A 3-*staircase* joining p to q is defined similarly using elbows whose first segment is vertical. Similarly for 2- and 4-staircases, except that we require $v_i \prec v_j$ (Figure 1, left).

The boundary of the rectilinear convex hull of a point set P is a subset of the union of four staircases, a 1-, a 2-, a 3-, and a 4-staircase whose vertices are the 1-, 2-, 3-, and 4-extremal points of P. See again Figure 1. Observe that the rectilinear convex hull $RCH(P)$ of a point set P is disconnected when either the complements $\mathbb{R}^2 - M_1(P)$ and $\mathbb{R}^2 - M_3(P)$ intersect, as shown in Figure 1, right, or the complements $\mathbb{R}^2 - M_2(P)$ and $\mathbb{R}^2 - M_4(P)$ intersect. A pinched point u of $RCH(P)$ occurs when u is either both 1-extremal and 3-extremal, as shown in Figure 1, right, or both 2-extremal and 4-extremal. Note that the *size* of $RCH(P)$ is the number of points of P which are i-extremal for at least one $i \in \{1, 2, 3, 4\}$.

[1] Some recent works [4, 5, 6] use the notation $\mathcal{RH}(P)$ for the rectilinear convex hull of P.

Throughout this paper, we will use a, b, c, and d to denote the leftmost, bottommost, rightmost, and topmost points of P, where a, b, c, d are not necessarily different (see Figure 1, right). Given two points u and v in the plane, let $B(u, v)$ be the smallest open axis-aligned rectangle containing u and v, and let $P(u, v) = P \cap B(u, v)$. We say that $RCH(P)$ is *vertically separable* if $B(a, d)$ and $B(b, c)$ are separated by a vertical line. Given P and a horizontal line ℓ, let P' be the image of P under a reflection around ℓ. The following lemma is key for our algorithms, and we assume, when needed, that P is vertically separable.

▶ **Lemma 1.1.** *$RCH(P)$ or $RCH(P')$ is vertically separable.*

2 Rectilinear convex hull of maximum size

In this section, we solve the MaxRCH problem. For every pair of points p, q such that $p \prec q$, let $\mathcal{C}^i_{p,q}$, $i = 2, 4$, be an i-staircase with endpoints p and q of maximum size. Similarly, if $p \prec' q$, let $\mathcal{C}^i_{p,q}$, $i = 1, 3$, be an i-staircase with endpoints p and q of maximum size, see Figure 3. Our goal is to combine four staircases in order to obtain a subset S of P whose rectilinear convex hull is of maximum size. All of this has to be done carefully, since the occurrence of pinched points may lead to over counting.

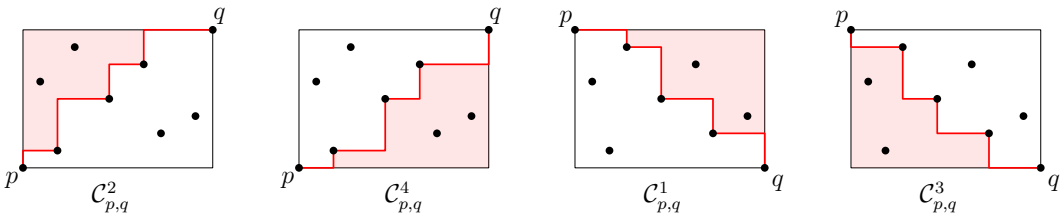


Figure 3 Examples of $\mathcal{C}^i_{p,q}$.

Our algorithm to solve the MaxRCH problem proceeds in three steps: In the first step we calculate all of the $\mathcal{C}^i_{p,q}$, $i = 1, \ldots, 4$. In the second step we calculate what we call *triple staircases* (yet to be defined). In the third step we show how to combine triple staircases and the $\mathcal{C}^4_{p,q}$ staircases to solve the MaxRCH problem. In this step we will make sure that the solution thus obtained is vertically separable. Our algorithm will run in $O(n^3)$ time and $O(n^2)$ space. The main tool is the use of dynamic programming applying to some recurrences. (In the appendix we describe in detail the steps of our algorithm.) Let $C^i_{p,q}$ be the number of elements of P in $\mathcal{C}^i_{p,q}$. Note that $C^i_{p,q}$ equals the maximum number of i-extremal points over all $X \subseteq \{p, q\} \cup P(p, q)$ with $p, q \in X$.

The first step: Compute the numbers $C^i_{p,q}$, for $i \in \{1, 2, 3, 4\}$, $p, q \in P$. These can be done in $O(n^3)$ time and $O(n^2)$ space, using dynamic programming applying the recurrence:

$$C^i_{p,q} = \begin{cases} 1 & \textbf{if } p = q \\ \max\{1 + C^i_{r,q}\} \text{ over all } r \in P(p, q) & \textbf{if } p \neq q. \end{cases} \tag{1}$$

Using the $C^i_{p,q}$, it is a routine matter to determine a staircase $\mathcal{C}^i_{p,q}$ of maximum size.

The second step: Given a point set S, we define the *triple staircase* associated to S, as the concatenation of the 1-, 2-, and the 3-staircases of the rectilinear convex hull of S. In this step, our goal is to obtain triple staircases of maximum cardinality starting and ending at some pairs of points of P. Triple staircases allow us to manage pinched points.

Consider $p, q \in P$ such that $p \prec q$ or $p = q$. Let $Z(p, q) = B(p, q) \cup Q_4(p) \cup Q_4(q)$, and let $z(p, q) = Z(p, q) \cap P$ (see Figure 4). Let S' be a subset of $z(p, q)$ such that the triple

staircase, denoted as $\mathcal{T}_{p,q}$, associated to $S' \cup \{p, q\}$ is of maximum cardinality. Observe that $M_1(S') \cap M_2(S') \cap M_3(S')$ may contain points in $P(p, q)$, it may be disconnected, and it may have pinched points. Note that p and q are always the endpoints of $\mathcal{T}_{p,q}$ (see Figure 5). Let $X_{p,q}$ denote the set of extreme vertices of $\mathcal{T}_{p,q}$, and let $T_{p,q}$ be the cardinality of $X_{p,q}$.

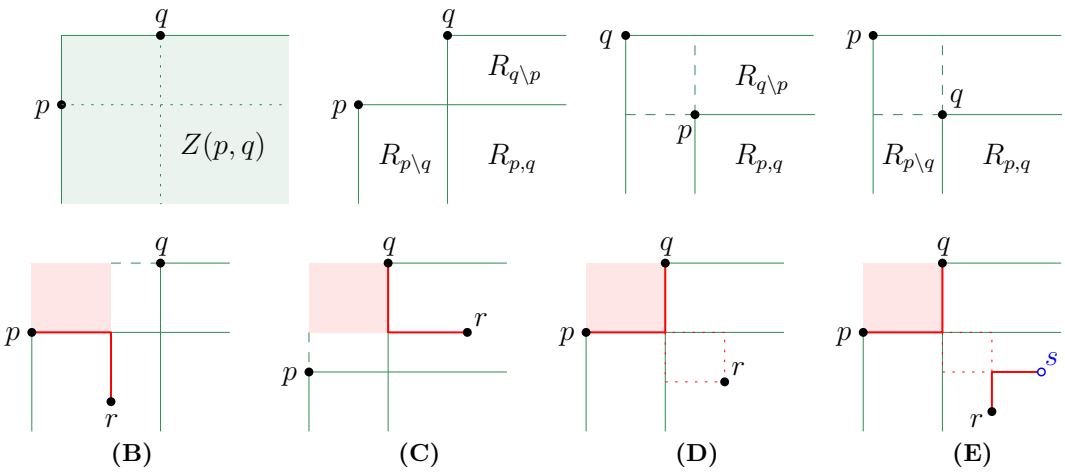


Figure 4 Top: Region $Z(p, q)$ and subsets $R_{p\setminus q}$, $R_{q\setminus p}$, and $R_{p,q}$. Bottom: cases in the recursive computation of $T_{p,q}$.

We calculate all of the $T_{p,q}$'s using Equation (2). Let $\alpha_{p,q} = 1$ if $p = q$, and $\alpha_{p,q} = 2$ if $p \neq q$. We use dynamic programming to compute the values of the table T using the following recurrence:

$$T_{p,q} = \max \begin{cases} C^2_{p,q} & \textbf{(A)} \\ 1 + T_{r,q} \text{ over all } r \in R_{p\setminus q} : P(p, r) = \emptyset & \textbf{(B)} \\ 1 + T_{p,r} \text{ over all } r \in R_{q\setminus p} : P(q, r) = \emptyset & \textbf{(C)} \\ \alpha_{p,q} + T_{r,r} \text{ over all } r \in R_{p,q} : P(p, r) = P(q, r) = \emptyset & \textbf{(D)} \\ \alpha_{p,q} + U_{p,r} \text{ over all } r \in R_{p,q} : P(p, r) = P(q, r) = \emptyset & \textbf{(E)} \end{cases} \tag{2}$$

where

$$U_{p,r} = \max\{T_{r,s}\} \text{ over all } s \in R_{p\setminus r} : P(p, s) = \emptyset. \tag{3}$$

▶ **Lemma 2.1.** *The previous recurrence correctly calculates $T_{p,q}$, the size of $X_{p,q}$.*

We use now triple staircases and the $\mathcal{C}^4_{p,q}$ staircases to solve the MaxRCH problem.

The third step: We proceed as follows. For $a, d \in P$ with $a \prec d$, we compute an optimal solution $S_{a,d} \subseteq P$ having a as its leftmost point and d as its topmost point. Finding $S_{a,d}$ is not as simple as joining the maximum 4-staircase of size $\mathcal{C}^4_{a,d}$ with $\mathcal{T}_{a,d}$, since $\mathcal{T}_{a,d}$ might have extreme vertices in the rectangle $B(a, d)$ which can also be vertices of the 4-staircase. To see how we arrive to this solution, consider a vertically separable optimal solution $S_{a,d}$. We traverse the 1-staircase of $S_{a,d}$ from a to d, and let $e \in S_{a,d}$ be the first point of P that belongs to the set $R_{a,d}$. Let $f \in S_{a,d}$ be the point of P that precedes e in the staircase and belongs to $\{a\} \cup R_{a\setminus d}$ (see Figure 6). Let ℓ be the horizontal line through e. Then, f must satisfy the conditions of the next lemma.

▶ **Lemma 2.2.** *By the optimality of $S_{a,d}$, the point f satisfies: (i) $P(f, e) = \emptyset$, (ii) $C^1_{a,f}$ is maximum among all points of P in $\{a\} \cup R_{a\setminus d}$ that are above ℓ (see Figure 7).*

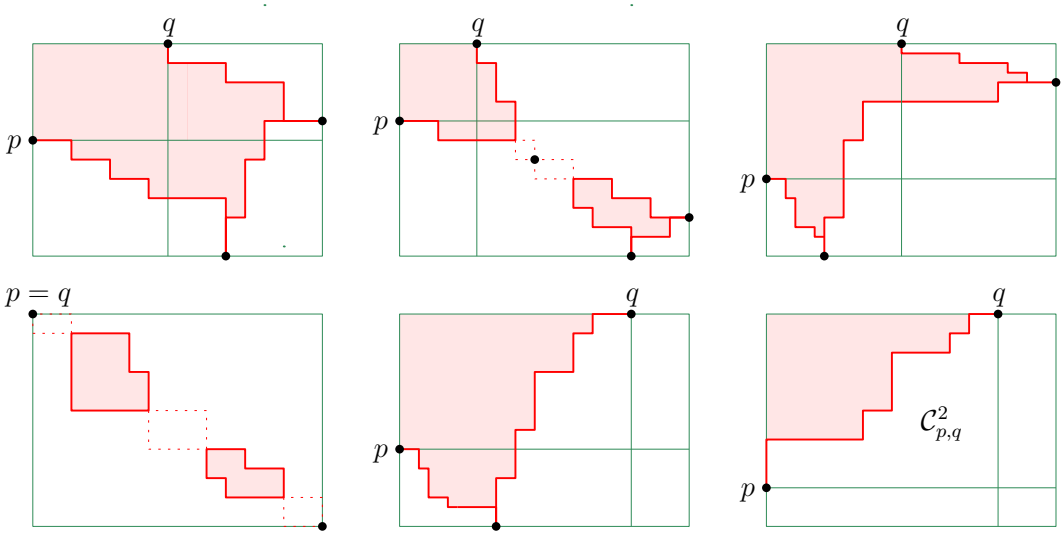


Figure 5 Examples of triple staircases $\mathcal{T}_{p,q}$.

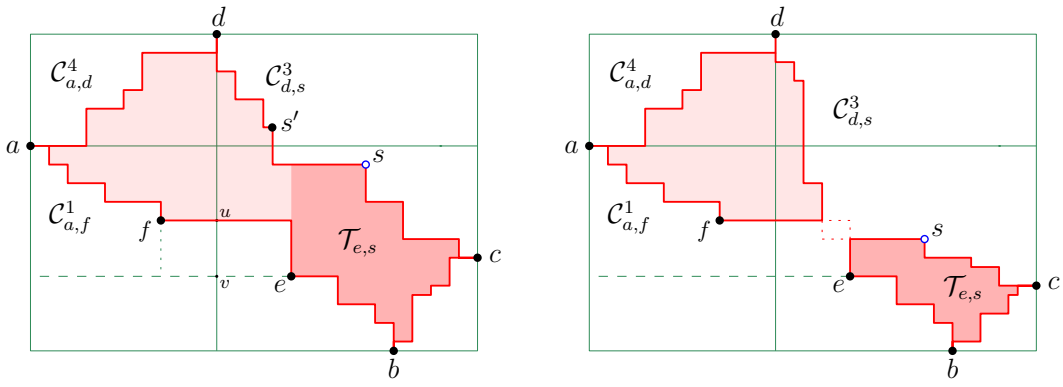


Figure 6 The third step of the algorithm.

Using Lemma 2.2, $S_{a,d}$ can be found as follows. Sweep a line ℓ from top to bottom, stopping at all the points of $\{a\} \cup R_{a\setminus d} \cup R_{a,d}$. Every time ℓ passes over a point of $\{a\} \cup R_{a\setminus d}$, we update the point f satisfying condition (ii) in $O(1)$ time. Furthermore, every time ℓ passes over a point $e \in R_{a,d}$, we verify whether condition (i) is satisfied. If this is the case, we consider e as a candidate to be the first point of the 1-staircase of $S_{a,d}$ in $R_{a,d}$, and set $f(e) = f$. Let E be the set of all points that are candidates to be point e. If $E = \emptyset$, we return $T_{a,d} = -1$. Otherwise, we return:

$$\max_{e \in E} \left\{ C^4_{a,d} + C^1_{a,f(e)} + V_{d,e} - 2 \right\} \tag{4}$$

as the size of the optimal $S_{a,d}$, where table V (similar to table U) is a table such that it contains an entry $V_{d,e}$ for every pair $d, e \in P$ such that $d \prec' e$. Each $V_{d,e}$ satisfies:

$$V_{d,e} \;=\; \max \left\{ C^3_{d,s} + T_{e,s} - 1 \right\} \text{ over all } s \in R_{d\setminus e} \cup \{e\} \text{ such that } pred(s,d) \prec' e \tag{5}$$

where $pred(s,d)$ (s' in Figure 6) is the point of P on $\mathcal{C}^3_{d,s}$ which precedes s while traversing $\mathcal{C}^3_{d,s}$ from d to s. Hence, the size of $RCH(S_{a,d})$ equals:

$$C^4_{a,d} + C^1_{a,f} + C^3_{d,s} + T_{e,s} - 3 \;=\; C^4_{a,d} + C^1_{a,f} + V_{d,e} - 2. \tag{6}$$

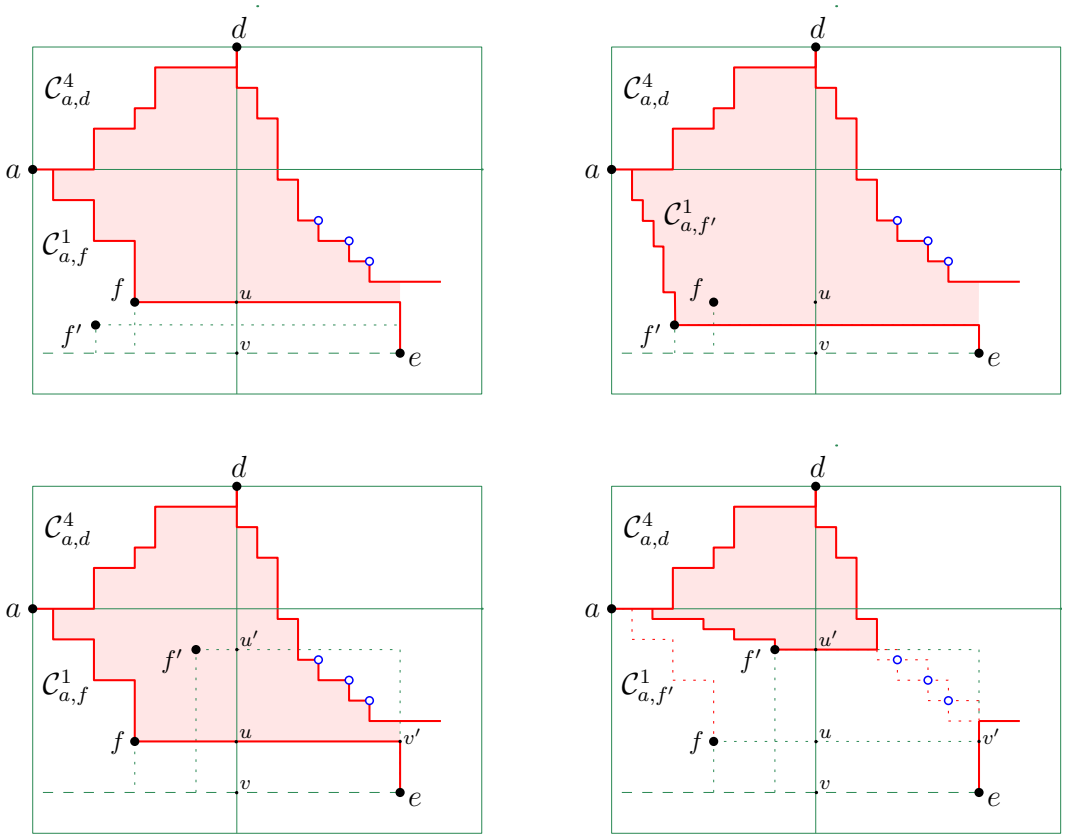


Figure 7 Illustration of Lemma 2.2.

We subtract 3 since a, d, and s are counted twice each. By using table V and Equations (4) and (5), we compute the optimal solution in $O(n^3)$ time and $O(n^2)$ space. Note that all the $pred(s, d)$ can be computed in $O(n^3)$ time using Equation (1). Hence, we have:

▶ **Theorem 2.3.** *The* MaxRCH *problem can be solved in* $O(n^3)$ *time and* $O(n^2)$ *space.*

3 Maximum size/area empty rectilinear convex hulls

In this section, we adapt the algorithm of Section 2 to solve the MaxEmptyRCH and the MaxAreaRCH problems. Note that, by Lemma 1.1, we can assume that $B(a, d)$ and $B(b, c)$ are separated by a vertical line in both the MaxEmptyRCH and MaxAreaRCH problems.

To solve the MaxEmptyRCH problem in $O(n^3)$ time and $O(n^2)$ space, we make modifications to the steps of our previous algorithm. The rest of the algorithm is the same.

▶ **Theorem 3.1.** *The* MaxEmptyRCH *problem can be solved in* $O(n^3)$ *time and* $O(n^2)$ *space.*

Given a bounded set $Z \subset \mathbb{R}^2$, let $\mathsf{Area}(Z)$ denote the area of Z. To solve the MaxAreaRCH problem, we will proceed as in the previous result, but we need to sum areas, not to count points in all of our recurrences. Given an empty i-staircase $\mathcal{C}^i_{p,q}$, let $C^i_{p,q}$ be now $\mathsf{Area}(B(p, q) \cap M_i(P(p, q)))$.

▶ **Theorem 3.2.** *The* MaxAreaRCH *problem can be solved in* $O(n^3)$ *time and* $O(n^2)$ *space.*

References

1 O. Aichholzer, R. Fabila-Monroy, H. González-Aguilar, T. Hackl, M. A. Heredia, C. Huemer, J. Urrutia, P. Valtr, and B. Vogtenhuber. On k-gons and k-holes in point sets. *Computational Geometry*, 48(7):528–537, 2015.

2 O. Aichholzer, R. Fabila-Monroy, H. González-Aguilar, T. Hackl, M. A. Heredia, C. Huemer, J. Urrutia, and B. Vogtenhuber. 4-holes in point sets. *Computational Geometry*, 47(6):644–650, 2014.

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. *Computational Geometry*, 47(5):605–613, 2014.

4 C. Alegría-Galicia, T. Garduño, A. Rosas-Navarrete, C. Seara, and J. Urrutia. Rectilinear convex hull with minimum area. In *Computational Geometry, LNCS, vol 7579 - XIV Spanish Meeting on Computational Geometry, EGC 2011*, 226–235, 2012.

5 C. Alegría-Galicia, D. Orden, C. Seara, and J. Urrutia. Rectilinear and $\mathcal{O}$-convex hull with minimum area. *CoRR*, abs/1710.10888, 2017.

6 C. Alegría-Galicia, C. Seara, and J. Urrutia. Computing containment relations between rectilinear convex hulls. In *Mexican Conference on Discrete Mathematics and Computational Geometry, 60th birthday of Jorge Urrutia*, November 11–15, 2013.

7 D. Avis and D. Rappaport. Computing the largest empty convex subset of a set of points. In *Proceedings of the 1st Annual Symposium on Computational Geometry*, 161–167, 1985.

8 P. Brass, W. Moser, and J. Pach. Convex polygons and the Erdős Szekeres problem. Chapter 8.2 in Research Problems in Discrete Geometry, Springer, 2005.

9 C. Bautista-Santiago, J. M. Díaz-Báñez, D. Lara, P. Pérez-Lantero, J. Urrutia, and I. Ventura. Computing optimal islands. *Operations Research Letters*, 39(4):246–251, 2011.

10 V. Chvátal and G. Klincsek. Finding largest convex subsets. *Congressus Numerantium*, 29:453–460, 1980.

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

12 W. Morris and V. Soltan. The Erdős-Szekeres problem on points in convex position–a survey. *Bulletin of the American Mathematical Society*, 37(4):437–458, 2000.

13 T. Ottmann, E. Soisalon-Soininen, and D. Wood. On the definition and computation of rectilinear convex hulls. *Information Sciences*, 33(3):157–171, 1984.

14 Preparata, Franco P. and Shamos, Michael I. Computational geometry: an introduction, 2012, Springer Science & Business Media.

15 G. J. Rawlins and D. Wood. Ortho-convexity and its generalizations. In *Machine Intelligence and Pattern Recognition*, 6:137–152, 1988.

Routing in Histograms*

Man-Kwun Chiu[1], Jonas Cleve[1], Katharina Klost[1], Matias Korman[3], Wolfgang Mulzer[1], André van Renssen[4], Marcel Roeloffzen[5], and Max Willert[1]

1 **Institut für Informatik, Freie Universität Berlin, 14195 Berlin, Germany**
{chiumk,jonascleve,kathklost,mulzer,willerma}@inf.fu-berlin.de
3 **Department of Computer Science, Tufts University, Medford, MA, USA**
matias.korman@tufts.edu
4 **School of Computer Science, University of Sydney, Sydney, Australia**
andre.vanrenssen@sydney.edu.au
5 **TU Eindhoven, Eindhoven, The Netherlands**
m.j.m.roeloffzen@tue.nl

Abstract

Let P be a *histogram* with n vertices, i.e., an x-monotone orthogonal polygon whose upper boundary is a single edge. Two points $p, q \in P$ are *co-visible* if and only if the (axis-parallel) bounding rectangle of p and q is in P. In the r-visibility graph of P, we connect two vertices of P with an *unweighted* edge if and only if they are co-visible. We consider *routing with preprocessing* in P. We may preprocess P to obtain a *label* and a *routing table* for each vertex of P. Then, we must be able to route a packet between any two vertices s and t of P, where each step may use only the label of the target node t, the routing table, and the neighborhood of the current node.

We present a routing scheme for histograms that sends any data packet along a shortest path. Each label needs $O(\log n)$ bits, while the routing table of each node consists of a single bit.

1 Introduction

The *routing* problem is a classic question in distributed graph algorithms [15, 22]. We have a graph G and would like to preprocess it for the following task: route a data packet located at some *source* vertex s to a *target* vertex t, given by its *label* – a bit string that identifies the node in the network. The routing should have the following properties: (A) *locality*: to determine the next step of the packet, it should use only information available locally at the current vertex. The most important local information consists of a *routing table* for each vertex; (B) *efficiency*: the packet should travel along a path whose length is not much larger than the length of a shortest path between s and t. The ratio between the length of this routing path and a shortest path is called the *stretch factor*; and (C) *compactness*: the space requirements for labels and routing tables should be small. Storing the complete shortest path tree of v in every node v of G leads to a perfect efficiency but lacks compactness.

There are many compact routing schemes for general graphs [1, 2, 11–13, 23, 24]. For example, the scheme by Roditty and Tov [24] needs to store a poly-logarithmic number of bits in the packet header and it 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. The local routing tables use $mn^{O(1/\sqrt{\log n})}$ space. In the late 1980's, Peleg and Upfal [22] proved that in general graphs, any routing

* Partially supported by ERC STG 757609, DFG grant MU 3501/1-2, MEXT KAKENHI No. 17K12635, NSF award CCF-1422311, and JST ERATO Grant Number JPMJER1201, Japan. A full version is available on the arXiv under arXiv:1902.06599.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

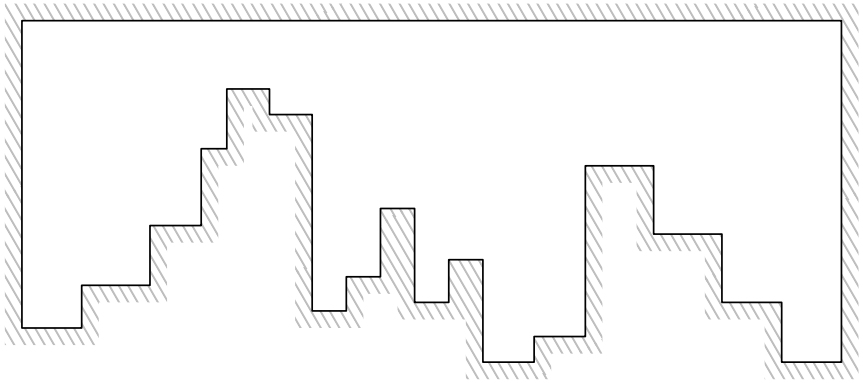


Figure 1 The boundary of a histogram has an x-monotone chain and a single horizontal edge.

scheme with constant stretch factor must store $\Omega(n^c)$ bits per vertex, for some constant $c > 0$. Thus, it is natural to focus on special graph classes to obtain better routing schemes. For instance, there are compact and efficient routing schemes for trees, planar graphs, unit disk graphs, and metric spaces with bounded doubling dimension [14, 18, 19, 25–27, 29].

Another approach is *geometric routing*: the graph resides in a geometric space, and the routing algorithm has to determine the next vertex for the packet purely based on the local geometric information (and possibly the packet header), see for instance [9, 10] and the references therein. There are no routing tables. In a recent result, Bose *et al.* [10] show that when vertices do not store any routing tables, no geometric routing scheme can achieve stretch factor $o(\sqrt{n})$. This lower bound applies irrespective of the header size.

We consider routing in visibility graphs of polygons. Banyassady *et al.* [3] presented an efficient and compact routing scheme for polygonal domains assuming Euclidean weights. They ask whether there is an efficient routing scheme for visibility graphs with unit weights, arguably a more applied setting. We address this open problem by combining the two approaches of geometric and compact routing: we use routing tables at the vertices to represent information about the structure of the graph, but we also assume that the labels of all adjacent vertices are stored in a link table and are therefore available for each node. This is reasonable from a practical point of view. The link table is not part of the routing table and the size of this list is not relevant for the compactness, since it depends purely on the graph and cannot be influenced during preprocessing. We focus on r-visibility (*rectilinear*-visibility) graphs of histograms: a histogram P is an orthogonal polygon bounded by an x-monotone polygonal chain and a single horizontal line segment, see Figure 1. Two vertices v, w in P are connected in the r-visibility graph $G(P)$ by an unweighted edge if and only if the axis-parallel rectangle spanned by v and w is contained in the (closed) region P. Even this seemingly simple case turns out to be quite challenging and reveals the whole richness of the compact routing problem in unweighted, geometrically defined graphs. Furthermore, histograms constitute a natural starting point, since they are crucial building blocks in many visibility problems; see, for instance, [4–8, 17]. In addition, r-visibility is a popular concept that enjoys many useful structural properties, see, e.g., [16, 17, 20, 21, 28].

We present a routing scheme for $G(P)$ with label size $2 \cdot \lceil \log n \rceil$, routing table size 1 and stretch 1, i.e., we route on a shortest path.

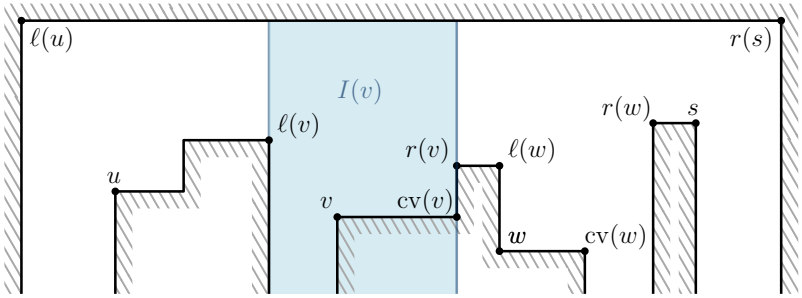


Figure 2 The interval $I(v)$ as well as left, right, and corresponding vertices.

2 Preliminaries

Routing schemes. Let $G = (V, E)$ be an *undirected, unweighted, simple, connected graph*. The (closed) *neighborhood* of a vertex $v \in V$, $N(v)$, is the set containing v and its adjacent nodes. The length of a path π in G is denoted by $|\pi|$. Moreover, for $v, w \in V$, we let $d(v, w)$ denote the length of a shortest path in G with endpoints v and w.

We define a *routing scheme*. Every node v is assigned a *label* $\mathrm{lab}(v) \in \{0,1\}^*$ to identify v in the network and a *routing table* $\rho(v) \in \{0,1\}^*$ storing relevant properties of G. Labels and routing tables are chosen during *preprocessing*. Moreover, every node has a *link table*—a list of the labels of $N(v)$. The algorithm to find the next step of a packet is modeled by a *routing function* $f\colon \left(\{0,1\}^*\right)^3 \to V$. The function uses the link and routing table at a current node s as well as the label $\mathrm{lab}(t)$ of the target node t to determine a next node v adjacent to s where the packet is forwarded to. The routing scheme is *correct* if the following holds: for any $s, t \in V$, let $p_0 = s$ and $p_{i+1} = f\Big(\mathrm{lab}\left(N(p_i)\right), \rho(p_i), \mathrm{lab}(t)\Big)$, for $i \geq 0$. Then, there is a $k = k(s,t) \geq 0$ with $p_k = t$ and $p_i \neq t$, for $i < k$. The routing scheme *reaches* t in k steps. We call $\pi : \langle p_0, \ldots, p_k \rangle$ the *routing path* from s to t. The *routing distance* is $k(s,t)$.

The various pieces of information used for the routing should be small. This is measured by the *label* and *routing table size*. The routing path should be as small as possible. This is measured by the *stretch*—the ratio of the lengths of the routing and the shortest path.

Polygons. Let P be an *x-monotone orthogonal polygon* in general position with n vertices $V(P)$. No three vertices in $V(P)$ lie on a horizontal line. We call P a *histogram* if the upper boundary is a single horizontal *base edge*. Its endpoints are the *base vertices*. The vertices of P are indexed counterclockwise from 0 to $n-1$ starting at the left base vertex. For $v \in V(P)$, we write v_x for the x-coordinate, v_y for the y-coordinate, and v_{id} for the index.

We consider the *r-visibility graph* $G(P) = \big(V(P), E(P)\big)$ of P: there is an edge between two vertices $v, w \in V(P)$ if they are *co-visible*, i.e., the axis-aligned rectangle spanned by v and w is in (the closed set) P. We call $d(v, w)$ the *hop distance* between two vertices in v, w.

Next, we classify the vertices of P. A vertex v in P is incident to exactly one horizontal edge h. We call v a *left* vertex if it is the left endpoint of h; otherwise, v is a *right* vertex. Furthermore, v is *convex* if the interior angle at v is $\pi/2$; otherwise, v is *reflex*. Accordingly, every vertex of P is either *ℓ-convex*, *r-convex*, *ℓ-reflex*, or *r-reflex*.

Visibility Landmarks. Let P be a histogram. We associate with each $v \in V(P)$ three landmark vertices in P; see Figure 2. The *corresponding vertex* of v, $\mathrm{cv}(v)$, shares the same horizontal edge with v. The *left bounding vertex* of v, $\ell(v)$, is the leftmost visible vertex

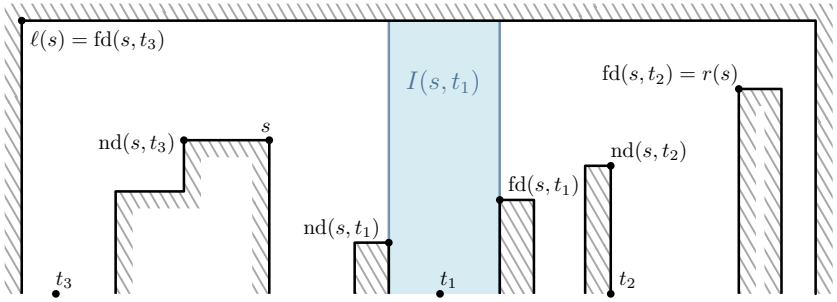


Figure 3 The near and the far dominators. Observe that $\mathrm{fd}(s, t_3)$ is not a vertex.

from v closest to the base edge, i.e., $\ell(v) = \mathrm{argmin}\{w_{\mathrm{id}} \mid w \in N(v)\}$. The *right bounding vertex* of v, $r(v)$, is defined analogously, i.e., $r(v) = \mathrm{argmax}\{w_{\mathrm{id}} \mid w \in N(v)\}$.

Let $v, w \in V(P)$. The *interval* $[v, w]$ is the set of vertices in P whose x-coordinates lie between those of v and w, i.e., $[v, w] = \{u \in V(P) \mid v_x \leq u_x \leq w_x\}$. By general position, this corresponds to index intervals. More precisely, if v is either an r-reflex vertex or the left base vertex and w is either ℓ-reflex or the right base vertex, then $[v, w] = \{u \in V(P) \mid v_{\mathrm{id}} \leq u_{\mathrm{id}} \leq w_{\mathrm{id}}\}$. The set $I(v) = [\ell(v), r(v)]$ is called the *interval of* v. We have $N(v) \subseteq I(v)$.

Let s and t be two vertices with $t \in I(s) \setminus N(s)$. We define two more landmarks for s and t. Assume $s_x < t_x$, the other case is symmetric. The *near dominator* $\mathrm{nd}(s, t)$ of t with respect to s is the rightmost vertex in $N(s)$ that is not to the right of t. If there is more than one such vertex, $\mathrm{nd}(s, t)$ is the vertex closest to the base line. The *far dominator* $\mathrm{fd}(s, t)$ of t with respect to s is the leftmost vertex in $N(s)$ that is no to the left of t. If there is more than one such vertex, $\mathrm{fd}(s, t)$ is the vertex closest to the base line. The interval $I(s, t) = [\mathrm{nd}(s, t), \mathrm{fd}(s, t)]$ has all vertices between the near and far dominator; see Figure 3.

3 Visibility and Paths

Let P be a histogram. We present some observations on the visibility in P. Then, we analyze the structure of (shortest) paths in a histograms. We omit the proofs for space reasons.

▶ **Observation 3.1.** *Let $v \in V(P)$ be r-reflex or the left base vertex, and let $u \in [v, r(v)]$ be a vertex distinct from v and $r(v)$. Then, $I(u) \subseteq [v, r(v)]$.*

▶ **Observation 3.2.** *Let $v \in V(P)$ be a left (right) vertex distinct from the base vertex. Then, v can see exactly two vertices to its right (left): $\mathrm{cv}(v)$ and $r(v)$ ($\ell(v)$).*

The following lemma identifies some vertices that must appear on any path; see Figure 4.

▶ **Lemma 3.3.** *Let $v, w \in V(P)$ be co-visible vertices such that v is either r-reflex or the left base vertex and w is either ℓ-reflex or the right base vertex. Let s and t be two vertices with $s \in [v, w]$ and $t \notin [v, w]$. Then, any path between s and t includes v or w.*

The next lemma shows that if $t \notin I(s)$, there is a shortest path from s to t that uses the higher vertex of $\ell(s)$ and $r(s)$; see Figure 4.

▶ **Lemma 3.4.** *Let s and t be two vertices with $t \notin I(s)$. If $\ell(s)_y > r(s)_y$ ($\ell(s)_y < r(s)_y$), then there is a shortest path from s to t using $\ell(s)$ ($r(s)$).*

The next lemma considers the case where t is in $I(s)$. Then, the near and far dominator are the potential vertices that lie on a shortest path from s to t.

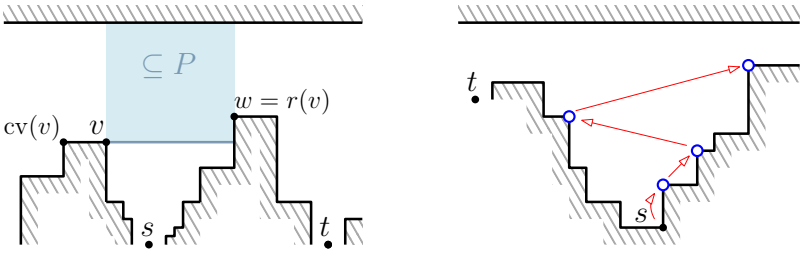


Figure 4 Left: Any s-t-path includes v or w. Right: A shortest s-t path using the higher vertex.

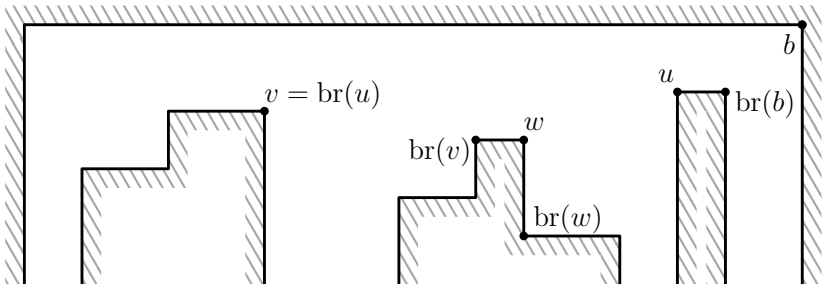


Figure 5 The breakpoints of some vertices.

▶ **Lemma 3.5.** *Let s and t be two vertices with $t \in I(s) \setminus N(s)$. Then, $\mathrm{nd}(s,t)$ is reflex and either $\mathrm{fd}(s,t) = \ell(\mathrm{nd}(s,t))$ or $\mathrm{fd}(s,t) = r(\mathrm{nd}(s,t))$.*

4 The Routing Scheme

Let P be a histogram, $|V(P)| = n$. Our approach is as follows: as long as a target vertex t is not contained in the interval $I(s)$ of a current vertex s, i.e., as long as there is a higher vertex blocking visibility between s and t, we have to leave the current pocket as quickly as possible. Once we have reached a high enough spike, we find the pocket containing t.

Labels and routing tables. Let v be a vertex. If v is convex and not a base vertex we let $\mathrm{lab}(v) = v_{\mathrm{id}}$. Otherwise, suppose that v is an r-reflex vertex or the left base vertex. The *breakpoint of* v, $\mathrm{br}(v)$, is the left endpoint of the horizontal edge with the highest y-coordinate to the right of and below v visible from v; analogous definitions apply to ℓ-reflex vertices and the right base vertex; see Figure 5. We set $\mathrm{lab}(v) = (v_{\mathrm{id}}, \mathrm{br}(v)_{\mathrm{id}})$. The routing table $\rho(v)$ stores one bit, indicating whether $\ell(v)_y > r(v)_y$, or not.

The routing function. We are given the current vertex s and the label $\mathrm{lab}(t)$ of the target vertex t. If t is visible from s, i.e., if $\mathrm{lab}(t) \in \mathrm{lab}(N(s))$, we directly go from s to t. Thus, assume $t \notin N(s)$. First, we check $t \in I(s)$ as follows: we determine the smallest and largest id in the link table of s, i.e., we determine $\ell(s)_{\mathrm{id}}$ and $r(s)_{\mathrm{id}}$ and check $t_{\mathrm{id}} \in [\ell(s)_{\mathrm{id}}, r(s)_{\mathrm{id}}]$, which is the case if and only if $t \in I(s)$. There are two cases, illustrated in Figure 6.

First, assume $t \notin I(s)$. If $\rho(s)$ indicates $\ell(s)_y > r(s)_y$, we take the hop to $\ell(s)$; otherwise, we take the hop to $r(s)$. By Lemma 3.4, this hop is on a shortest path from s to t.

Second, suppose that $t \in I(s) \setminus N(s)$. This case is a bit more involved. We use the link table of s and the label of t to determine $\mathrm{fd}(s,t)$ and $\mathrm{nd}(s,t)$. Again, we can do this by

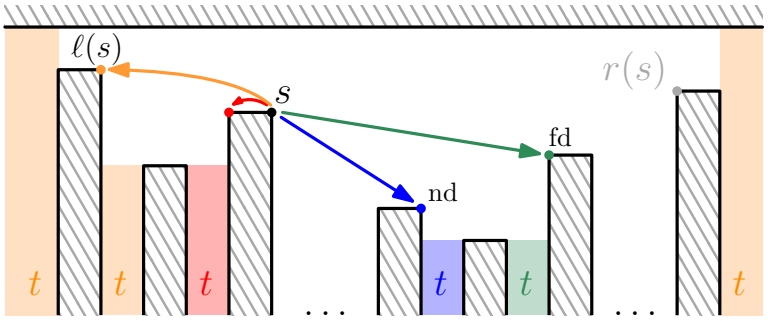


Figure 6 The cases where the vertex t lies and the vertices where the data packet is sent to.

comparing the ids. Lemma 3.5 states that either $\mathrm{fd}(s,t) = \ell(\mathrm{nd}(s,t))$ or $\mathrm{fd}(s,t) = r(\mathrm{nd}(s,t))$. We discuss the case that $\mathrm{fd}(s,t) = r(\mathrm{nd}(s,t))$, the other case is symmetric. By Lemma 3.3, any shortest path between s and t includes $\mathrm{fd}(s,t)$ or $\mathrm{nd}(s,t)$. Moreover, due to Lemma 3.5, $\mathrm{nd}(s,t)$ is reflex, and we can use its label to access $b_{\mathrm{id}} = \mathrm{br}(\mathrm{nd}(s,t))_{\mathrm{id}}$. The vertex b splits $I(s,t) = [\mathrm{nd}(s,t), \mathrm{fd}(s,t)]$ into two disjoint subintervals $[\mathrm{nd}(s,t), b]$ and $[\mathrm{cv}(b), \mathrm{fd}(s,t)]$. Also, b and $\mathrm{cv}(b)$ are not visible from s, as they are located strictly between the far and the near dominator. Based on b_{id}, we can now decide on the next hop.

If $t \in [\mathrm{nd}(s,t), b]$, we take the hop to $\mathrm{nd}(s,t)$. If $t = b$, our packet uses a shortest path. Assume that t lies between $\mathrm{nd}(s,t)$ and b. Then, b is ℓ-reflex and we can apply Lemma 3.3 to see that any shortest path from s to t includes $\mathrm{nd}(s,t)$ or b. But since $d(s,b) = 2$, our data packet routes along a shortest path. If $t \in [\mathrm{cv}(b), \mathrm{fd}(s,t)]$, we take the hop to $\mathrm{fd}(s,t)$. The argument is similar. The following theorem summarizes our discussion.

▶ **Theorem 4.1.** *Let P be a histogram with n vertices. There is a routing scheme for $G(P)$ with label size $2 \cdot \lceil \log n \rceil$, routing table size 1, and stretch 1.*

5 Conclusion

We gave the first routing scheme for the hop-distance in simple polygons. In particular, we have a routing scheme for histograms with label size $2 \cdot \lceil \log n \rceil$, routing table size 1, and stretch 1. As we show in the full version, our method also extends to more general *double histograms*. The following open problems arise naturally. First, it would be interesting to see how the routing scheme extends to monotone polygons as well as arbitrary orthogonal polygons, assuming r-visibility. After that, it will be interesting to take a closer look at (orthogonal) polygons assuming the more common notion of l-visibility (*line*-visibility). Here, the structure of visibility—even in simple histograms—is much more complicated and we can no longer assume integer coordinates.

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 *Proc. 28th Annu. Internat. Sympos. Algorithms Comput. (ISAAC)*, pages 10:1–10:13, 2017.

4 Andreas Bärtschi. Coloring variations of the art gallery problem. *Master's thesis, Department of Mathematics, ETH Zürich*, 2011.

5 Andreas Bärtschi, Subir Kumar Ghosh, Matúš Mihalák, Thomas Tschager, and Peter Widmayer. Improved bounds for the conflict-free chromatic art gallery problem. In *Proc. 30th Annu. Sympos. Comput. Geom. (SoCG)*, page 144, 2014.

6 Andreas Bärtschi and Subhash Suri. Conflict-free chromatic art gallery coverage. *Algorithmica*, 68(1):265–283, 2014.

7 Pritam Bhattacharya, Subir Kumar Ghosh, and Sudebkumar Pal. Constant approximation algorithms for guarding simple polygons using vertex guards. `arXiv:1712.05492`, 2017.

8 Pritam Bhattacharya, Subir Kumar Ghosh, and Bodhayan Roy. Approximability of guarding weak visibility polygons. *Discrete Applied Mathematics*, 228:109–129, 2017.

9 Prosenjit Bose, Rolf Fagerberg, André van Renssen, and Sander Verdonschot. Optimal local routing on Delaunay triangulations defined by empty equilateral triangles. *SIAM J. Comput.*, 44(6):1626 – 1649, 2015.

10 Prosenjit Bose, Rolf Fagerberg, André van Renssen, and Sander Verdonschot. Competitive local routing with constraints. *J. of Computational Geometry*, 8(1):125–152, 2017.

11 Shiri Chechik. Compact routing schemes with improved stretch. In *Proc. ACM Symp. Princ. Dist. Comp. (PODC)*, pages 33–41, 2013.

12 Lenore J Cowen. Compact routing with minimum stretch. *J. Algorithms*, 38(1):170–183, 2001.

13 Tamar Eilam, Cyril Gavoille, and David Peleg. Compact routing schemes with low stretch factor. *J. Algorithms*, 46(2):97–114, 2003.

14 Pierre Fraigniaud and Cyril Gavoille. Routing in trees. In *Proc. 28th Internat. Colloq. Automata Lang. Program. (ICALP)*, pages 757–772, 2001.

15 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.

16 Frank Hoffmann. On the rectilinear art gallery problem. In *Proc. 17th Internat. Colloq. Automata Lang. Program. (ICALP)*, pages 717–728, 1990.

17 Frank Hoffmann, Klaus Kriegel, Subhash Suri, Kevin Verbeek, and Max Willert. Tight bounds for conflict-free chromatic guarding of orthogonal art galleries. *Comput. Geom. Theory Appl.*, 2018.

18 Haim Kaplan, Wolfgang Mulzer, Liam Roditty, and Paul Seiferth. Routing in unit disk graphs. *Algorithmica*, 80(3):830–848, 2018.

19 Goran Konjevod, Andréa W. Richa, and Donglin Xia. Scale-free compact routing schemes in networks of low doubling dimension. *ACM Trans. Algorithms*, 12(3):27:1–27:29, 2016.

20 Rajeev Motwani, Arvind Raghunathan, and Huzur Saran. Covering orthogonal polygons with star polygons: The perfect graph approach. *J. Comput. System Sci.*, 40(1):19–48, 1990.

21 Joseph O'Rourke. *Art gallery theorems and algorithms*. Oxford University Press, 1987.

22 David Peleg and Eli Upfal. A trade-off between space and efficiency for routing tables. *J. ACM*, 36(3):510–530, 1989.

23 Liam Roditty and Roei Tov. New routing techniques and their applications. In *Proc. ACM Symp. Princ. Dist. Comp. (PODC)*, pages 23–32, 2015.

24 Liam Roditty and Roei Tov. Close to linear space routing schemes. *Distributed Computing*, 29(1):65–74, 2016.

25 Nicola Santoro and Ramez Khatib. Labelling and implicit routing in networks. *The Computer Journal*, 28(1):5–8, 1985.

26 Mikkel Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *J. ACM*, 51(6):993–1024, 2004.

27 Mikkel Thorup and Uri Zwick. Compact routing schemes. In *Proc. 13th ACM Symp. Par. Algo. Arch. (SPAA)*, pages 1–10, 2001.

28 Chris Worman and J Mark Keil. Polygon decomposition and the orthogonal art gallery problem. *Internat. J. Comput. Geom. Appl.*, 17(02):105–138, 2007.

29 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.

Peeling Digital Potatoes

Loïc Crombez[1], Guilherme D. da Fonseca[1], and Yan Gérard[1]

1 **Université Clermont Auvergne and LIMOS, Clermont-Ferrand, France**

Abstract

The potato-peeling problem (also known as convex skull) is a fundamental computational geometry problem and the fastest algorithm to date runs in $O(n^8)$ time for a polygon with n vertices that may have holes. In this paper, we consider a digital version of the problem. A set $K \subset \mathbb{Z}^2$ is *digital convex* if $\text{conv}(K) \cap \mathbb{Z}^2 = K$, where $\text{conv}(K)$ denotes the convex hull of K. Given a set S of n lattice points, we present polynomial time algorithms for the problems of finding the largest digital convex subset K of S (*digital potato-peeling problem*) and the largest union of two digital convex subsets of S. The two algorithms take roughly $O(n^3)$ and $O(n^9)$ time, respectively. We also show that those algorithms provide an approximation to the continuous versions.

1 Introduction

The *potato-peeling problem* [16] (also known as *convex skull* [23]) consists of finding the convex polygon of maximum area that is contained inside a given polygon (possibly with holes) with n vertices. The fastest exact algorithm known takes $O(n^7)$ time without holes and $O(n^8)$ if there are holes [9]. The problem is arguably the simplest geometric problem for which the fastest exact algorithm known is a polynomial of high degree and this high complexity motivated the study of approximation algorithms [8, 17]. Multiple variations of the problem have been considered, including triangle-mesh [1] and orthogonal [14, 24] versions. In this paper, we consider a digital geometry version of the problem.

The *digital potato-peeling problem* is defined as follows and is illustrated in Figure 1(a,b).

▶ Problem 1 (Digital potato-peeling). Given a set $S \subset \mathbb{Z}^2$ of n lattice points described by their coordinates, determine the *largest* set $K \subseteq S$ that is digital convex (i.e., $\text{conv}(K) \cap \mathbb{Z}^2 = K$), where largest refers either to the area of $\text{conv}(K)$, or $|K|$.

Heuristics for the digital potato-peeling problem have been presented in [7, 10], but no exact algorithm. We also consider the question of covering the largest area with two digital convex subsets. The problem is defined as follows and is illustrated in Figure 1(a,c).

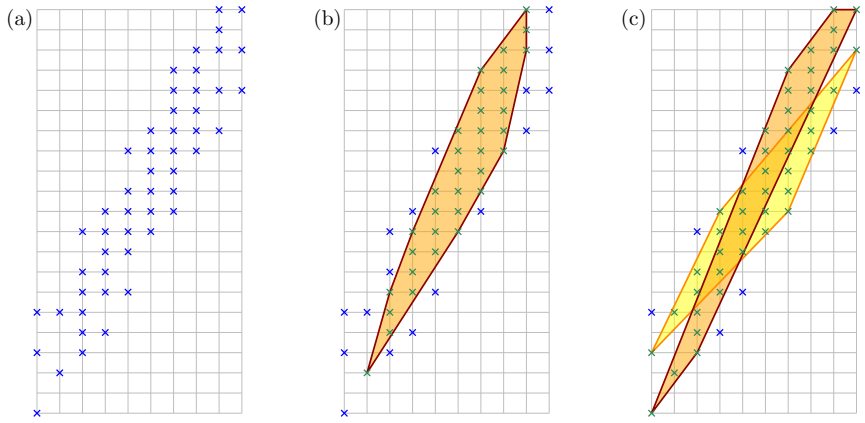


Figure 1 (a) Input lattice set S. (b) Largest digital convex subset of S (Problem 1). (c) Largest union of two digital convex subsets of S (Problem 2).

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

▶ Problem 2 (Digital 2-potato peeling). Given a set $S \subset \mathbb{Z}^2$ of n lattice points described by their coordinates, determine the largest set $K = K_1 \cup K_2 \subseteq S$ such that K_1 and K_2 are both digital convex, where largest refers to the area of $\text{conv}(K_1) \cup \text{conv}(K_2)$.

A related continuous problem consists of completely covering a polygon by a small number of convex polygons inside of it. O'Rourke showed that covering a polygon with the minimum number of convex polygons is decidable [18, 19], but the problem has been shown to be NP-Hard with or without holes [13, 20]. Shermer [22] presents a linear time algorithm for the case of two convex polygons and Belleville [5] provides a linear time algorithm for three. We are not aware of any previous results on finding a fixed (non-unit) number of convex polygons inside a given polygon and maximizing the area covered.

Our results

We present polynomial time algorithms to solve each of these two problems. In Section 2, we show how to solve the digital potato-peeling problem in $O(n^3 \log r)$ time, where r is the diameter of the input S. Our algorithm builds the convex polygon $\text{conv}(K)$ through its triangulation, using a triangle range counting data structure [11] together with Pick's theorem [21] to test the validity of each triangle. The $O(\log r)$ factor comes from the gcd computation to apply Pick's theorem. Our algorithm makes use of the following two properties: (i) it is possible to triangulate K using only triangles that share a common bottom-most vertex v and (ii) if the polygons lying on both sides of one such triangle (including the triangle itself) are convex, then the whole polygon is convex.

These two properties are no longer valid for Problem 2, in which the solution $\text{conv}(K_1) \cup \text{conv}(K_2)$ is the union of two convex polygons. Also, since convex shapes are not pseudo-disks (the boundaries may cross an arbitrarily large number of times), separating the input with a constant number of lines is not an option. Instead of property (i), our approach uses the fact that the union of two (intersecting) convex polygons can be triangulated with triangles that share a common vertex ρ (that may not be a vertex of either convex polygon). Since ρ may not have integer coordinates, we can no longer use Pick's theorem, and resort to the formulas from Beck and Robins [4] or the algorithm from Barvinok [3] to count the lattice points inside each triangle in $O(\text{polylog}\, r)$ time.

Furthermore, to circumvent the fact that the solution no longer obeys property (ii), we use a directed acyclic graph (DAG) that encapsulates the orientation of the edges of both convex polygons. For those reasons, the running time of our algorithm for Problem 2 increases to $O(n^9 + n^6 \,\text{polylog}\, r)$. The corresponding algorithm is described in Section 3.

2 Digital Potato Peeling

In this section, we present an algorithm to solve the digital potato-peeling problem in $O(n^3 \log r)$ time, where n is the number of input points and r is the diameter of the set.

A digital convex set K can be described by its convex hull $\text{conv}(K)$ whose vertices are lattice points. Instead of explicitly building K, our algorithm constructs $conv(K)$. Note that it is always possible to triangulate a convex polygon with k vertices using $k-2$ triangles that share a bottom-most vertex ρ (*fan triangulation*). We first consider the rooted variation of the digital potato-peeling problem, where the point ρ is given as part of the input.

▶ Problem 3 (Rooted digital potato peeling). Given a set $S \subset \mathbb{Z}^2$ of n lattice points given by their coordinates and a point $\rho \in S$, determine the *largest* set $K \subseteq S$ that is digital convex and has ρ as the right-most point at the bottom-most row of K.

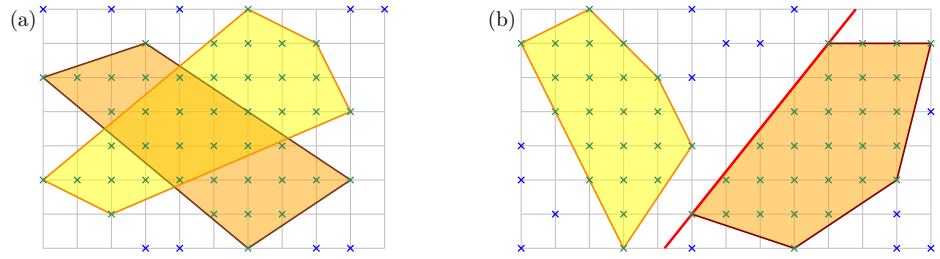


Figure 2 (a) The two optimal sets intersect. (b) The two optimal sets are disjoint and there is a supporting separating line.

Without loss of generality, we assume that all points in S lie either on the same row or on a row above ρ and all points on the same row of ρ are to the left of ρ. We refer to ρ as the *root*. Let $p_1, \ldots, p_n$ denote the points of S sorted clockwise around ρ, starting from left.

Let $\triangle_{i,j}$ denote the (closed) triangle whose vertices are ρ, p_i, p_j with $i < j$. We say that a triangle $\triangle_{i,j}$ is *valid* if $\triangle_{i,j} \cap \mathbb{Z}^2 = \triangle_{i,j} \cap S$. To algorithmically verify that $\triangle_{i,j}$ is valid, we compare $|\triangle_{i,j} \cap S|$ and $|\triangle_{i,j} \cap \mathbb{Z}^2|$ using Pick's theorem and a triangle range counting query [11]. The total time to test the validity of a triangle (after preprocessing) is $O(\log r)$.

The algorithm incrementally builds the fan triangulation of $\mathrm{conv}(K)$ by appending valid triangles from left to right using dynamic programming.

For all $p_i, p_j \in S$ with $i < j$ and such that $\triangle_{i,j}$ is valid, the algorithm determines the largest convex polygon that has $\triangle_{i,j}$ as the right-most triangle. We refer to this convex polygon as $C_{i,j}$. The key property to efficiently compute $C_{i,j}$ is

$$C_{i,j} = \triangle_{i,j} \cup \max_h C_{h,i}, \text{ where } h < i \text{ is such that } \triangle_{i,j} \cup \triangle_{h,i} \text{ is convex.}$$

For a given i, by sorting all $C_{h,i}$ with $h < i$ according to their size and sorting all $\triangle_{i,j}$ according to the position of p_j around p_i, all the $C_{i,j}$ can be computed in $O(n \log n)$ time using the aforementioned property.Considering all n values of i and the initial sorting, the total time to solve Problem 3 is $O(n^2 \log r)$. In order to solve Problem 1, we test all n possible values of $\rho \in S$, proving the following theorem.

▶ **Theorem 1.** *There exists an algorithm to solve Problem 1 (digital potato peeling) in $O(n^3 \log r)$ time, where n is the number of input points and r is the diameter of the input.*

3 Digital 2-Potato Peeling

In this section, we show how to find two digital convex sets K_1, K_2, maximizing the area of $\mathrm{conv}(K_1) \cup \mathrm{conv}(K_2)$. Either the two convex hulls intersect or they do not (Figure 2). We treat those two cases separately and the solution to Problem 2 is the largest among both. Hence, we consider the two following variations of the 2-potato-peeling problem.

▶ Problem 4 (Disjoint 2-potato peeling). Given a set $S \subset \mathbb{Z}^2$ of n lattice points given by their coordinates, determine the *largest* two digital convex sets $K_1 \cup K_2 \subseteq S$ such that $\mathrm{conv}(K_1) \cap \mathrm{conv}(K_2) = \emptyset$.

▶ Problem 5 (Intersecting 2-potato peeling). Given a set $S \subset \mathbb{Z}^2$ of n lattice points given by their coordinates, determine the *largest* union of two digital convex sets $K_1 \cup K_2 \subseteq S$ such that $\mathrm{conv}(K_1) \cap \mathrm{conv}(K_2) \neq \emptyset$. In this case, largest means the maximum area of $\mathrm{conv}(K_1) \cup \mathrm{conv}(K_2)$.

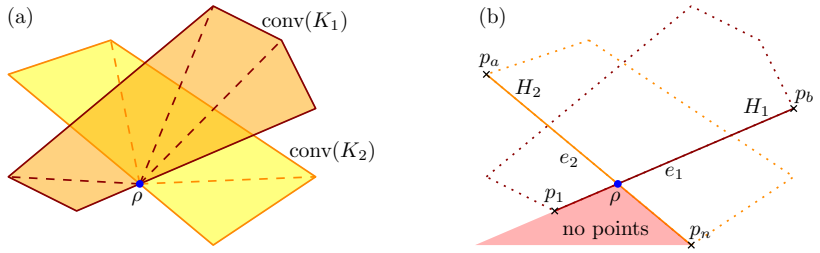


Figure 3 (a) A fan triangulation of two intersecting convex polygons from a point ρ. (b) Definitions used to solve Problem 6.

3.1 Disjoint Convex Polygons

It is well known that any two disjoint convex shapes can be separated by a straight line. Moreover two convex polygons can be separated by a supporting line of an edge of one of the convex polygons that contains no vertex of the other convex polygon (Figure 2(b)).

For each ordered pair of distinct points $p_1, p_2 \in S$, we define two subsets S_1, S_2. The set S_1 contains the points on the line p_1, p_2 or to the left of it (according to the direction $p_2 - p_1$). The set S_2 contains the remaining points of S.

For each pair of sets S_1, S_2, we independently solve Problem 1 for S_1 and S_2. Since there are $O(n^2)$ pairs and each pair takes $O(n^3 \log r)$ time, we solve Problem 4 in $O(n^5 \log r)$ time.

3.2 Intersecting Convex Polygons

The more interesting case is when the two convex polygons intersect (Problem 5). Note that it is possible to triangulate the union of two convex polygons that share a common boundary point ρ using a fan triangulation around ρ (Figure 3). Hence we consider the following rooted version of the problem.

▶ Problem 6 (Rooted 2-potato peeling). Given a set $S \subset \mathbb{Z}^2$ of n lattice points represented by their coordinates and two edges $e_1, e_2 \in S^2$ that cross at a point ρ, determine the *largest* union of two digital convex sets $K_1, K_2 \subseteq S$ such that e_1 is an edge of conv(K_1) and e_2 is an edge of conv(K_2).

Let ρ be the intersection point of e_1, e_2. To solve Problem 6 we encode the problem into a DAG (V, E) whose longest path corresponds to the solution. To avoid confusion, we use the terms *node* and *arc* for the DAG and keep the terms *vertex* and *edge* for the polygons.

Let $\mathcal{T}$ be the set of valid triangles with two vertices from S and ρ as the remaining vertex. The nodes $V = \mathcal{T}^2 \cup \{v_0\}$ are ordered pairs of valid triangles and a starting node v_0. The number of nodes is $|V| = O(n^4)$.

Each node $(\triangle_1, \triangle_2) \in V$ is such that $\triangle_1$ (resp. $\triangle_2$) is used to build the fan triangulation of conv(K_1) (resp. conv(K_2)). The arcs are defined in a way such that, at each step as we go through a path of the DAG, we add one triangle either to conv(K_1) or to conv(K_2). The arcs enforce the convexity of both conv(K_1) and conv(K_2). Furthermore, we enforce that we always append a triangle to the triangulation that is the least advanced of the two (in clockwise order), unless we have already reached the last triangle of conv(K_1). This last condition allows us to define the arc lengths in a way that it corresponds to the area of the union of the two convex polygons. Figure 4 illustrates the result of following a path on the DAG. As there is $O(n^4)$ pairs of starting edges and each DAG has $O(n^5)$ arcs, the total running time is roughly $O(n^9)$.

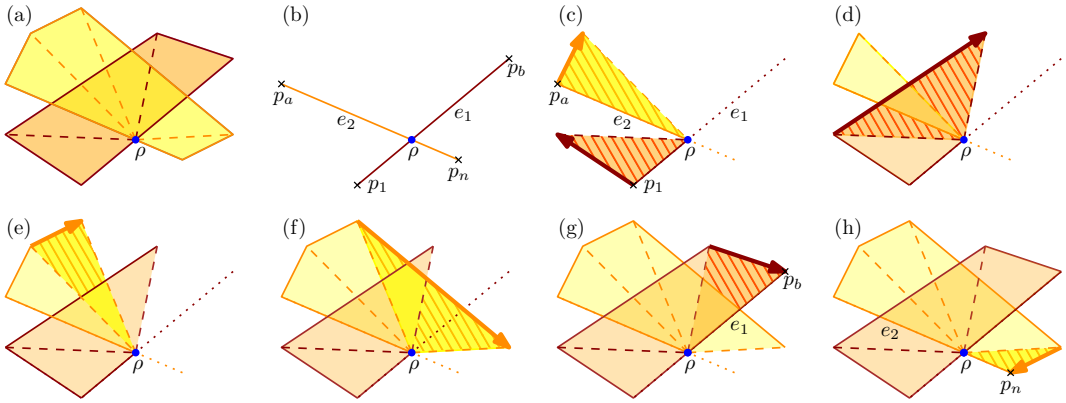


Figure 4 Steps of the algorithm from Section 3.2. Figure (a) represents the solution, while Figures (b) to (h) represent the triangulation obtained at each node of a path. The newly covered area that is assigned as the length of the corresponding arc is marked. In (b), we have the initial pair of edges e_1, e_2 which corresponds to the starting vertex. A first pair of triangles with vertices p_1 and p_a is obtained in (c). From (c) to (d) and from (f) to (g), the triangle $\triangle_1$ (less advanced than triangle $\triangle_2$) advances. From (d) to (e) and from (e) to (f), triangle $\triangle_2$ advances. In (g), the triangle $\triangle_1$ has reached the final node p_b. $\triangle_2$ advances until it reaches the end.

4 Conclusion and Open Problems

The (continuous) potato peeling problem is a very peculiar problem in computational geometry. The fastest algorithms known have running times that are polynomials of substantially high degree. Also, we are not aware of any algorithms (or difficulty results) for the natural extensions to higher dimensions (even 3d) or to a fixed number of convex bodies.

In this paper, we focused on a digital version of the problem. Many problems in the intersection of digital, convex, and computational geometry remain open. Our study falls in the following framework of problems, all of which receive as input a set of n lattice points $S \subset \mathbb{Z}^d$ and are based on a fixed parameter $k \geq 1$.

1. Is S the union of at most k digital convex sets?
2. What is the smallest superset of S that is the union of at most k digital convex sets?
3. What is the largest subset S that is the union of at most k digital convex sets?

In [12], the authors considered the first problem for $k = 1$, presenting polynomial time solutions (which may still leave room for major improvements for $d > 3$). We are not aware of any previous solutions for $k > 1$. In contrast, the continuous version of the problem is well studied. The case of $k = 1$ can be solved easily by a convex hull computation or by linear programming. Polynomial algorithms are known for $d = 2$ and $k \leq 3$ [5, 22], as well as for $d = 3$ and $k \leq 2$ [6]. The problem is already NP-complete for $d = k = 3$ [6]. Hence, the continuous version remains open only for $d = 2$ and fixed $k > 3$.

It is easy to obtain polynomial time algorithms for the second problem when $k = 1$, since the solution consists of all points in the convex hull of S. The continuous version for $d = k = 2$ can be solved in $O(n^4 \log n)$ time [2]. Also, the orthogonal version of the problem is well studied (see for example [15]). We know of no results for the digital version.

The third problem for $d > 2$ or $k > 2$ remains open. The DAG approach that we used for $d = 2$ is unlikely to generalize to higher dimensions, since there is no longer a single order by which to transverse the boundary of a convex polytope. Surprisingly, even the continuous version seems to be unresolved for $d > 2$ or $k \geq 2$.

References

1 Boris Aronov, Marc Van Kreveld, Maarten Löffler, and Rodrigo I. Silveira. Peeling meshed potatoes. *Algorithmica*, 60(2):349–367, 2011.

2 Sang Won Bae, Hwan-Gue Cho, William Evans, Noushin Saeedi, and Chan-Su Shin. Covering points with convex sets of minimum size. *Theoretical Computer Science*, 718:14–23, 2018.

3 Alexander I. Barvinok. A polynomial time algorithm for counting integral points in polyhedra when the dimension is fixed. *Mathematics of Operations Research*, 19(4):769–779, 1994.

4 Matthias Beck and Sinai Robins. Explicit and efficient formulas for the lattice point count in rational polygons using Dedekind-Rademacher sums. *Discrete & Computational Geometry*, 27(4):443–459, Jan 2002.

5 Patrice Belleville. On restricted boundary covers and convex three-covers. In *5th Canadian Conference on Computational Geometry (CCCG)*, pages 467–472, 1993.

6 Patrice Belleville. Convex covers in higher dimensions. In *7th Canadian Conference on Computional Geometry (CCCG)*, pages 145–150, 1995.

7 Gunilla Borgefors and Robin Strand. An approximation of the maximal inscribed convex set of a digital object. In *13th International Conference on Image Analysis and Processing (ICIAP)*, pages 438–445, 2005.

8 Sergio Cabello, Josef Cibulka, Jan Kyncl, Maria Saumell, and Pavel Valtr. Peeling potatoes near-optimally in near-linear time. *SIAM Journal on Computing*, 46(5):1574–1602, 2017.

9 Jyun S. Chang and Chee K. Yap. A polynomial solution for the potato-peeling problem. *Discrete & Computational Geometry*, 1(2):155–182, 1986.

10 Jean-Marc Chassery and David Coeurjolly. Optimal shape and inclusion: open problems. In *Mathematical Morphology: 40 Years On, International Symposium on Mathematical Morphology*, Computational Imaging and Vision. Springer Verlag, 2005.

11 Bernard Chazelle, Micha Sharir, and Emo Welzl. Quasi-optimal upper bounds for simplex range searching and new zone theorems. *Algorithmica*, 8(1-6):407–429, 1992.

12 Loïc Crombez, Guilherme D. da Fonseca, and Yan Gérard. Efficient algorithms to test digital convexity. In *21st International Conference on Discrete Geometry for Computer Imagery (DGCI)*, 2019. URL: `http://fc.isima.fr/~fonseca/digitalconvexity.pdf`.

13 Joseph Culberson and Robert A. Reckhow. Covering polygons is hard. *Journal of Algorithms*, pages 17:2–44, 1994.

14 Mousumi Dutt, Arindam Biswas, Partha Bhowmick, and Bhargab B. Bhattacharya. On finding an orthogonal convex skull of a digital object. *International Journal of Imaging Systems and Technology*, 21(1):14–27, 2011.

15 Cem Evrendilek, Burkay Genç, and Brahim Hnich. Covering points with minimum/maximum area orthogonally convex polygons. *Computational Geometry*, 54:32–44, 2016.

16 Jacob E. Goodman. On the largest convex polygon contained in a non-convex n-gon, or how to peel a potato. *Geometriae Dedicata*, 11(1):99–106, 1981.

17 Olaf Hall-Holt, Matthew J. Katz, Piyush Kumar, Joseph S.B. Mitchell, and Arik Sityon. Finding large sticks and potatoes in polygons. In *17th annual ACM-SIAM symposium on Discrete algorithm (SODA)*, pages 474–483, 2006.

18 Joseph O'Rourke. The complexity of computing minimum convex covers for polygons. In *20th Allerton Conference on Communication, Control, and Computing*, pages 75–84, 1982.

19 Joseph O'Rourke. The decidability of covering by convex polygons. Technical Report JHU-EECS 82-4, Johns Hopking University, 1982.

20 Joseph O'Rourke. Some NP-hard polygon decomposition problems. *IEEE Transactions on Information Theory, IT-30*, pages 181–190, 1983.

21 Georg Pick. Geometrisches zur zahlenlehre. *Sitzungsberichte des Deutschen Naturwissenschaftlich-Medicinischen Vereines für Böhmen "Lotos" in Prag.*, v.47-48 1899-1900, 1899.

22 Thomas C. Shermer. On recognizing unions of two convex polygons and related problems. *Pattern Recognition Letters , 14(9)*, pages 737–745, 1993.

23 Tony C. Woo. The convex skull problem. Technical report, Department of Industrial and Operations Engineering, University of Michigan, 1986.

24 Derick Wood and Chee K. Yap. The orthogonal convex skull problem. *Discrete & Computational Geometry*, 3(4):349–365, 1988.

Consistent Digital Curved Rays

Jinhee Chun[1], Kenya Kikuchi[1], and Takeshi Tokuyama[1]

1 **GSIS Tohoku University**
jinhee/kikuchi/tokuyama@dais.is.tohoku.ac.jp

Abstract

Representing a family of geometric objects in the digital world where each object is represented by a set of pixels is a basic problem in graphics and computational geometry. One important criterion is the consistency, where the intersection pattern of the objects should be consistent with axioms of the Euclidean geometry, e.g., the intersection of two lines should be a single connected component. Previously, the set of linear rays and segments has been considered. In this paper, we extend this theory to families of digital curves going through the origin.

1 Introduction

In geometric computation, we often experience that finite-precision computation causes geometric inconsistency. This is because the representation of geometric objects in the pixel world does not always satisfy geometric properties such as Euclidean axioms. Figure 1 shows that a naive definition of digital lines may cause inconsistency, where the intersection of a pair of digital lines has more than one connected components.

Thus, it is important to seek for a digital represention of a family of geometric objects such that they satisfy a digital version of geometric properties. We propose the *consistent digital curved rays* in this paper, generalising consistent digital rays for straight lines [1, 4].

We consider the triangular region Δ defined by $\{(x, y) : x \geq 0, y \geq 0, x + y \leq n\}$ in the plane, and the integer grid $G = \{(i, j) : i, j \in \{0, 1, \ldots, n\}, i + j \leq n\}$ in the region. We can also handle a square region, but use Δ for ease of description of our method.

Each element of G is called a pixel (usually, a pixel is a square, but we represent it by its lower-left-corner grid point in this paper). We say a pixel is a boundary pixel if it lies on $x + y = n$. We consider an undirected graph structure under the four-neighbor topology such that $(i, j) \in G$ is connected to $(k, \ell) \in G$ if $(k, \ell) \in \{(i - 1, j), (i, j - 1), (i + 1, j), (i, j + 1)\}$.

A *digital ray* $S(p)$ is a path in G from the origin o to p, where $S(o) = \{o\}$ is a zero-length path. A family $\{S(p) : p \in G\}$ of digital rays uniquely assinged to each pixel is called *consistent* if the following three conditions hold:

1. If $q \in S(p)$, then $S(q) \subseteq S(p)$.
2. For each $S(p)$, there is a (not necessarily unique) boundary pixel r such that $S(p) \subseteq S(r)$.
3. Each $S(p)$ is a shortest path from o to p in G.

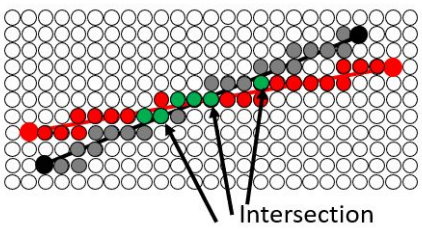


Figure 1 Inconsistency of intersection (green pixels) of two digital line segments

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 19–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

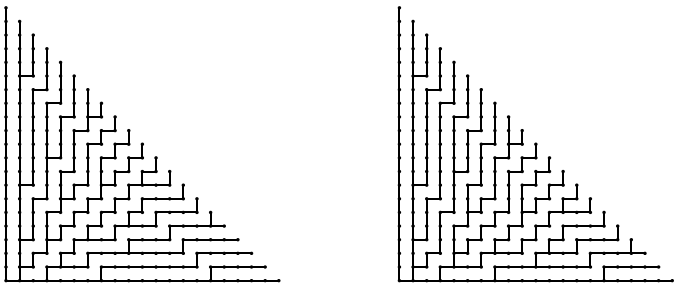


Figure 2 Consistent digital rays simulating linear rays (left) and parabolic rays(right)

The third condition is sometimes omitted in the literature, since it is not suitable for other types of grids such as a triangular grid, but we include it in this paper. The consistency implies that the union of paths $S(p)$ form a spanning tree T in G such that all leaves are boundary pixels as shown in Figure 2, and accordingly the intersection of two digital rays consists of single connected component. The tree T and also the family of digital rays are called CDR (*Consistent Digital Rays*).

Previously, the theory has been considered only for digital straightness [3]. Lubby [4] first gave a construction of CDR so that each $S(p)$ simulates a linear ray within Hausdorff distance $O(\log n)$, and showed that it is asymptotically tight. The construction was re-discovered by Chun *et al.*[1] to give further investigation, and Christ *et al.*[2] gave a construction of consistent digital line segments where the lines need not go through the origin.

We will extend the theory to families of curves with the same topology as linear rays.

A family $\mathcal{F}$ of nondecreasing curves in Δ is called *ray family* if each curve goes through the origin o, and for each point $(x, y) \in \Delta \setminus \{o\}$ there exists a unique curve of $\mathcal{F}$ going through it. We call an element of $\mathcal{F}$ a *ray*. Accordingly, each pair of rays intersect each other only at the origin. A typical example is the family of parabolas $y = ax^2$ for $a \geq 0$.

We give a construction method of CDR $T_{\mathcal{F}}$ in G such that the (unique) ray of $\mathcal{F}$ connecting o and a pixel p is simulated by the path $S(p)$ well, and show an $O(\sqrt{n \log n})$ bound of the Hausdorff distance for several ray families. Although the theoretical bound is much worse than the $\Theta(\log n)$ bound for the linear ray, it is the first nontrivial result for curved rays as far as the authors know, and experimentally the construction works better.

2 Consistent digital rays and their properties

Let us consider a CDR T of G. The set of pixels of G on the diagonal $x + y = k$ for $k = 0, 1, \ldots, n$ is called the level set $L(k)$. We call an edge from $L(k-1)$ towards $L(k)$ an incoming edge to (resp. outgoing edge from) a node in the level $L(k)$ (resp. $L(k-1)$). The following observation was given by Chun *et al.*[1](see Figure 3 for its illustration).

▶ **Lemma 2.1.** *In the level set $L(k)$ for $k \geq 1$, there exists a real value $0 < x(k) \leq k$ such that incoming edge of T to each node whose x-value is smaller than (resp. larger than or equal to) $x(k)$ is vertical (resp. horizontal). Accordingly, there exists a unique branching node of T in $L(k-1)$ (colored yellow in Figure 3).*

Thus, a CDR is completely characterized by the integer sequence $\lceil x(1) \rceil, \lceil x(2) \rceil, \ldots, \lceil x(n) \rceil$, where $1 \leq x(i) \leq i$. The following lemma is easy to verify.

▶ **Lemma 2.2.** *A (unique) CDR exists for each of $(n-1)!$ possible sequences as above.*

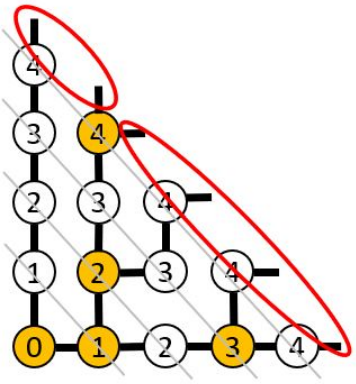


Figure 3 The branching nodes (colored yellow) and partition of incoming edges to the 5th level.

2.1 CDR for linear rays revisited

The CDR of linear rays given by Chun *et al.*[1] can be obtained by selecting $x(k)$ as uniformly as possible from $[1, k]$ by using a low-discreapancy pseudorandom sequence.

Let us consider the binary representation $k = \sum_{i=0}^{\infty} a(i)2^i$ of a natural number k. The van der Courput sequence (see [5]) is the sequence defined by a function $V(k) = \sum_{i=1}^{\infty} a(i)2^{-i}$ from natural numbers to $[0, 1]$. We remove $V(0) = 0$ from our consideration so that the range becomes $(0, 1]$. For example, for $6 = 2 + 4 = 110_2$, $V(6) = 0.11_2 = \frac{1}{2} + \frac{1}{4} = \frac{3}{4}$, where a sequence with a subscript 2 means the 2-adic representation of numbers.

The van der Courput sequence is known to be a low discrepancy sequence as shown in the following lemma (see e.g. [5]).

▶ **Lemma 2.3.** *Consider the set of points $S = \{k, V(k) : k = 0, 1, 2, \ldots, n\}$ in the region $X = [0, n] \times [0, 1]$. Then, for any axis parallel rectangle R in X, the difference from the number of points in $S \cap R$ and the area of R is $O(\log n)$.*

In particular, for each $m < n$, the set $\{V(i) : m \leq i \neq n\}$ gives an almost uniform distribution on $[0, 1]$ deterministically. We can set $x(k) = kV(k)$ to obtain a CDR that approximates the linear rays emanating from the origin with $O(\log n)$ distance bound. In order to generalize to the curved rays, we give the following interpretation.

Consider a line $y = ax$ intersecting $x + y = k$ at $q = (x_0, k - x_0)$. By definition, its slope is a, which is $\frac{k-x_0}{x_0}$. Naturally, we want to draw the line in the neighborhood of q with a segment of slope $\frac{k-x_0}{x_0}$, but we need to approximate it with a grid path. Therefore, ideally the ratio of vertical edges to the horizontal edges in the paths should be $\frac{k-x_0}{x_0}$ in a neighborhood of q.

By the definition of $x(k)$, the edge incoming to q is vertical if and only if q lies on the left of $x(k)$. If we take $x(k) = kV(k)$, the probability[1] that q is to the left of $x(k)$ is $\frac{x_0}{k}$, since $V(k)$ gives a uniform distribution. Thus, the incoming edge becomes horizontal and vertical with probabilities $\frac{x_0}{k}$ and $\frac{k-x_0}{k}$, respectively. Hence, the ratio between them is $\frac{k-x_0}{x_0}$ as desired.

We would like to extend this argument for other families of curves.

[1] Since the process is deterministic, we should say “ratio" rigorously, but we use the term “probability" for convenience’ sake.

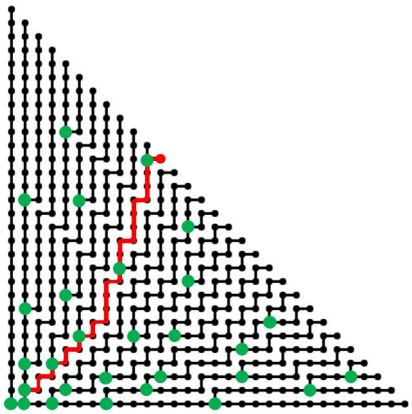


Figure 4 CDR T_{para}. Green nodes are branching nodes. Red path gives a digital parabolic ray.

3 CDR for families of curves

3.1 CDR for a family of parabolas

To improve readability, we start with the ray family $y = ax^2$ $(a \geq 0)$ of parabolas. We include the y-axis $x = 0$ in the family (this convention is applied to all other cases).

Consider a parabola $y = ax^2$ intersecting the level $x+y = k$ at $q = (x_0, k-x_0)$, The slope of tangent at q is $2ax_0$, which is $\frac{2(k-x_0)}{x_0}$. In order to approximate the parabola nicely, the tangent segment in the neighborhood of q should be approximated by a path that contains the horizontal edge with probability $\frac{x_0}{2k-x_0}$.

Thus, we should select $x(k)$ to be located on the left of q with probability $\frac{x_0}{2k-x_0}$. If we set $x_0 = kt$, this probability equals $\frac{t}{2-t}$. We consider a monotonically increasing function F in the range $[0, 1]$ and set $x(k) = kF(V(k))$. The probability that $x_0 = kt < x(k)$ is the probability that $F^{-1}(t) < V(k)$ from the monotonicity of F. Because of uniformity of $V(k)$, this probability equals $F^{-1}(t)$ (ignoring the small discrepancy).

Then, the probability (over k) that q is on the left of $x(k)$ is same as $t = x(k)/k \leq F(V(k))$. This probability is same as the probability that $F^{-1}(t) \leq V(k)$ from the monotonicity of F. Because of uniformity of $V(k)$, this means $F^{-1}(t) = \frac{t}{2-t}$ to meet our requirement, and $F(z) = \frac{2z}{z+1}$. Thus, we have $x(k) = \frac{2kV(k)}{V(k)+1}$ to define a CDR T_{para} illustrated in Figure 4.

The following theorem ensures that T_{para} approximate parabola rays well theoretically, and we will also demonstrate it works even better by implementation later.

▶ **Theorem 3.1.** *For each node $p = (i, j) \in G$, the Hausdorff distance between the parabola ray going through p and the path $S(p)$ from p towards the origin in T_{para} is $O(\sqrt{n \log n})$.*

The theorem is derived from the following lemma, which is obtained from Lemma 2.3. We omit proofs in this version.

▶ **Lemma 3.2.** *Consider the set of points $S = \{(k, V(k)) : k = 0, 1, 2, \ldots, n\}$. Let $f(x)$ be a nonincreasing or nondecreasing continuous function from $[0, n]$ to $[0, 1]$, and let $Q_I(f) = \{(x, y) : 0 \leq y \leq f(x), x \in I\}$ for any given interval $I \subset [0, 1]$. Then, the discrepancy (i.e., difference from the number of points in $S \cap Q_I(f)$ and the area $A(Q_I(f))$) is bounded by $c\sqrt{n \log n}$ for a suitable constant c.*

Note that for the discrepancy discussed in the above lemma, an $\Omega(\sqrt{n})$ lower bound is known even for a linear function [5].

3.2 Homogeneous polynomials

Le us consider the family $\mathcal{F}_j$ of curves defined by $y = f_a(x) = ax^j$ for $a \geq 0$. Here, the slope of the tangent of a curve at (x, y) is jax^{j-1}, which is jy/x. Thus, analogously to the parabola case, we have $F^{-1}(t) = \frac{t}{j-(j-1)t}$ and $F(z) = \frac{jz}{1+(j-1)z}$. We set $x(k) = \frac{jkV(k)}{(j-1)V(k)+1}$ for $k = 1, 2, \ldots, n$ to define a CDR $T_{\mathcal{F}_j}$. The following is obtained analogously to the parabola case.

▶ **Theorem 3.3.** *The path from p to the origin o in the CDR $T_{\mathcal{F}_j}$ approximates the curve in $\mathcal{F}_j$ going through p and o with an $O(\sqrt{n \log n})$ distance bound.*

3.3 Framework for a family of constant-multiplied curves

More generally, let us consider a nondecreasing differentiable function $y = f(x)$ for $x \in [0, n]$ such that $f(0) = 0$ and $f(x) > 0$ for $x > 0$. We define the family $\mathcal{F} = \{C_a : a \geq 0\}$ of curves, where C_a is defined by $y = af(x)$.

If C_a goes through (x_0, y_0), then $a = \frac{y_0}{f(x_0)}$. The slope of the curve C_a at (x_0, y_0) is $af'(x_0)$, which is (eliminating a) $\frac{f'(x_0)}{f(x_0)}(y_0)$. We consider the slope $T(x, k) = \frac{f'(x)}{f(x)}(k - x)$ along the diagonal $x + y = k$ for each k. We assume that it is monotonically decreasing in x for each fixed k.

Thus, we want to control so that the probability that the edge incoming to a pixel $(x, k - x)$ in $L(k)$ is horizontal with probability $\frac{1}{1+T(x,k)}$.

We consider a F such that $x(k) = kF(V(k))$ so that $(x, k - x)$ becomes horizontal with probability $\frac{1}{1+T(x,k)}$. Because of uniformity of $V(k)$, we set $F^{-1}(t) = \frac{1}{1+T(kt,k)}$. Note that we can show that F is monotone and the above argument holds. Although the explicit form of F might not be obtained, we can apply binary search to compute $F(z)$ for a given z utilizing the monotonicity. Thus, we can compute $x(k) = kF(V(k))$ within the pixel precision in $O(\log n)$ time.

3.3.1 Sigmoid curves and sine curves

Let $\mathcal{F}_{sig} = \{a\sigma(x) : 0 \leq a\}$, where $\sigma(x) = \frac{1}{1+e^{-x}} - \frac{1}{2}$ is the shifted sigmoid function. The curves $y = a\sigma(x)$ satisfy our conditions, and we have a CDR with distance bound $O(\sqrt{n \log n})$.

The sine curve $y = \sin(x)$ is not monotone, but we can define $\tilde{\sin}(x)$ by $\tilde{\sin}(x) = 0$ for $x < 0$, $\tilde{\sin}(x) = \sin x$ for $0 \leq x \leq \pi/2$ and $\tilde{\sin}(x) = 1$ for $x > \pi/2$. The curve $y = \tilde{\sin}(x)$ is monotonically nondecreasing and differentiable, and we can apply our CDR construction for the family of curves $y = a\,\tilde{\sin}(x)$ for $a \geq 0$.

The family of logarithmic functions $y = a \log(x + 1)$ can be also similarly handled.

The $O(\sqrt{n \log n})$ distance bound follows analogously to the parabola case for each family. Details are omitted in this version.

Figure 5 illustates CDR of families discussed above.

4 Experimental result and conclusion

For each grid width $n = 2^m$ up to $n = 2^{14}$, the worst-case Hausdorff distance between parabolas and digital rays in T_{para} is given in Figure 5, where it is about 12 for $n = 2^{14}$.

The experimental result suggests that the distance bound could be polylogarithmic, and our $O(\sqrt{n \log n})$ bound seems to be loose, although currently the lower bound mentioned for Lemma 3.2 prevents us to improve it beyond $\sqrt{n}$. On the other hand, Figure 6 suggests

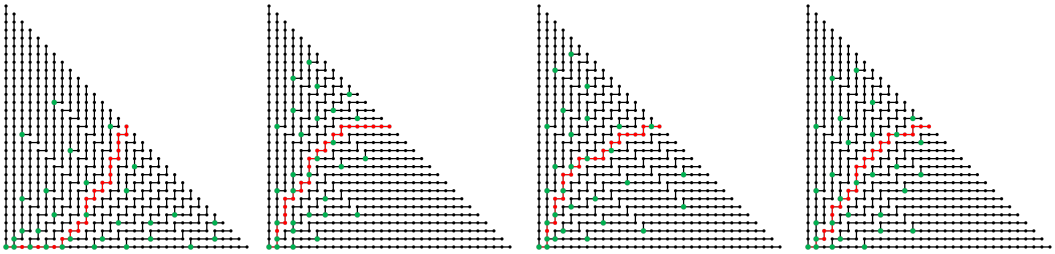


Figure 5 CDRs for (left to right) $y = ax^3$, $y = a\tilde{\sin}x$, $y = a\sigma(x)$, and $y = a\log(x+1)$. Green nodes are branching nodes. Each red path goes to the center boundary node.

the distance bound tends to be slightly larger than $O(\log n)$ for this construction, and investigation on both lower and upper bounds remains an interesting open problem. Another interesting problem is to find construction of consistent digital curves removing the ray condition. For example, it is curious to handle the set of all axis parallel parabolas.

Acknowledgement This work is supported by JSPS Kakenhi 17K19954 and 18H05291.

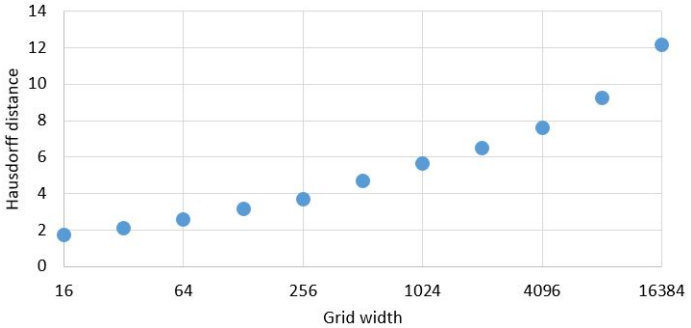


Figure 6 The largest distance from parabola ray and path in T_{para}.

References

1 J. Chun, M. Korman, M. Nöllenburg, T. Tokuyama, Consistent Digital Rays, *Discrete & Computational Geometry* 42-3 (2009) pp.359-378.

2 T. Christ, D. Pálvölgyi, M. Stojaković, Consistent Digital Line Segment, *Discrete & Computational Geometry* 47-4 (2012), pp. 691-710.

3 R. Klette, A. Rosenfeld, Digital straightness – a review *Discrete Applied Math.* 139 (2004), 197-230.

4 M.G.Lubby, Grid geometries which preserve properties of euclidean geometry: A study of graphics line drawing algorithms. In NATO Conference on Graphics/CAD, pages 397-432,1987.

5 J. Matoušek, *Geometric Discrepancy*, Springer Verlag 1991.

A Note on Universal Point Sets for Planar Graphs*

Manfred Scheucher[1], Hendrik Schrezenmaier[1], and Raphael Steiner[1]

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

Abstract

We investigate which planar point sets allow simultaneous straight-line embeddings of all planar graphs on a fixed number of vertices. We first show that at least $(1.293 - o(1))n$ points are required to find a straight-line drawing of each n-vertex planar graph (vertices are drawn as the given points); this improves the previous best constant 1.235 by Kurowski (2004).

Our second main result is based on exhaustive computer search: We show that no set of 11 points exists, on which all planar 11-vertex graphs can be simultaneously drawn plane straight-line. This strengthens the result by Cardinal, Hoffmann, and Kusters (2015), that all planar graphs on $n \leq 10$ vertices can be simultaneously drawn on particular "universal" sets of n points while there are no universal sets of size $n \geq 15$. Moreover, we provide a set of 23 planar 11-vertex graphs which cannot be simultaneously drawn on any set of 11 points. This, in fact, is another step towards a (negative) answer of the question, whether every two planar graphs can be drawn simultaneously – a question from Brass, Cenek, Duncan, Efrat, Erten, Ismailescu, Kobourov, Lubiw, and Mitchell (2007).

1 Introduction

A point set S in the Euclidean plane is called *n-universal for* a family $\mathcal{G}$ of planar n-vertex graphs if every graph G from $\mathcal{G}$ admits a plane straight-line embedding such that the vertices are drawn as points from S. A point set, which is n-universal for the family of all planar graphs, is simply called *n-universal*. We denote by $f_p(n)$ the size of a minimal n-universal set (for planar graphs), and by $f_s(n)$ the size of a minimal n-universal set for stacked triangulations, where stacked triangulations (a.k.a. planar 3-trees) are defined as follows:

▶ **Definition 1.1** (Stacked Triangulations)**.** Starting from a triangle, one may obtain any stacked triangulation by repeatedly inserting a new vertex inside a face (including the outer face) and making it adjacent to all the three vertices contained in the face.

Figures 2 and 3 show examples of stacked triangulations on 11 vertices.

De Fraysseix, Pach, and Pollack [10] showed that every planar n-vertex graph admits a straight-line embedding on a $(2n-4) \times (n-2)$ grid – even if the combinatorial embedding is prescribed. Moreover, the graphs are only embedded on a triangular subset of the grid. Hence, $f_p(n) \leq n^2 - O(n)$. This bound was further improved to the currently best known bound $f_p(n) \leq \frac{n^2}{4} - O(n)$ [4] (cf. [19, 5]). Also various subclasses of planar graphs have been studied intensively: Any stacked triangulation on n vertices (with a fixed outer face) can be drawn on a particular set of $f_s(n) \leq O(n^{3/2} \log n)$ points [13]. The first lower bound on the size of n-universal sets substantially greater than n was also given by de Fraysseix, Pach, and Pollack [10], who showed a lower bound of $f_p(n) \geq n + (1 - o(1))\sqrt{n}$. This was further

* The full version is available online [18].

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

improved by Chrobak and Karloff [9], and later on Kurowski [16] obtained the previous best lower bound of $(1.235 - o(1))n$ for $f_s(n)$ and thus also $f_p(n)$.

Cardinal, Hoffmann, and Kusters [8] showed that n-universal sets of size n exist for every $n \leq 10$, whereas for $n \geq 15$ no such set exists – not even for stacked triangulations. Moreover, they found a collection of 7,393 planar graphs on $n = 35$ vertices which cannot be simultaneously drawn straight-line on a common set of 35 points. We call such a collection of graphs a *conflict collection*. This was a first big step towards an answer to the question by Brass and others [6], which can be reformulated as follows:

▶ Question 1. Is there a conflict collection of size 2?

2 Results

Our first result is the following theorem, which further improves the lower bound on $f_s(n)$. We present the sketch of the proof in Section 3; for a detailed proof, see the full version [18].

▶ **Theorem 2.1.** *It holds that $f_s(n) \geq (\alpha - o(1))n$, where $\alpha = 1.293\ldots$ is the unique real-valued solution of the equation $\frac{\alpha^\alpha}{(\alpha-1)^{\alpha-1}} = 2$.*

In Section 4 we present our second result, which is another step towards a (negative) answer of Question 1 and strengthens the results from [8]. Its proof is based on exhaustive computer search.

▶ **Theorem 2.2** (Computer-assisted)**.** *There is a conflict collection consisting of 23 stacked triangulations on 11 vertices. Furthermore, there is no conflict collection consisting of 16 triangulations on 11 vertices.*

▶ **Corollary 2.3.** *There is no 11-universal set of size 11 – even for stacked triangulations. Hence, $f_p(11) \geq f_s(11) \geq 12$.*

3 Proof of Theorem 2.1

To prove the theorem, we use a refined counting argument based on a construction of a set of labeled stacked triangulations that was already introduced in [8]. There it was used to disprove the existence of n-universal sets of $n \geq 15$ points for the family of stacked triangulations.

▶ **Definition 3.1** (Labeled Stacked Triangulations, cf. [8, Section 3])**.** For every integer $n \geq 4$, we define the family $\mathcal{T}_n$ of labeled stacked triangulations on the set of vertices $V_n := \{v_1, ..., v_n\}$ inductively as follows:

(i) $\mathcal{T}_4$ consists only of the complete graph K_4 with labels $v_1, \ldots, v_4$.
(ii) If T is a labeled graph in $\mathcal{T}_{n-1}$ with $n \geq 5$, and $v_iv_jv_k$ defines a face of T, then the graph obtained from T by stacking the new vertex v_n to $v_iv_jv_k$ (i.e., connecting it to v_i, v_j, and v_k) is a member of $\mathcal{T}_n$.

The following, which is a consequence of Lemmas 1 and 2 in [8], is the basis of the proof of the new lower bound.

▶ **Corollary 3.2.** *The following two statements hold:*

(i) For any $n \geq 4$, $\mathcal{T}_n$ contains exactly $2^{n-4}(n-3)!$ stacked triangulations.

(ii) Let $P = \{p_1, \dots, p_m\}$ be a set of $m \geq n \geq 4$ labeled points in the plane. Then for any injection $\pi : V_n \to P$, there is at most one $T \in \mathcal{T}_n$ such that the embedding of T, which maps each vertex v_i to the point $\pi(v_i)$, defines a straight-line-embedding of T.

Sketch of Proof for Theorem 2.1. Let $n \geq 4$ be arbitrary and $m := f_s(n) \geq n$. There exists an n-universal point set $P = \{p_1, \dots, p_m\}$ for all stacked triangulations, hence for every $T \in \mathcal{T}_n$ there exists a straight-line embedding of T on P, with (injective) vertex-mapping $\pi : V_n \to P$. By Corollary 3.2 (ii), we know that no two stacked triangulations from $\mathcal{T}_n$ (each of which has the same vertex set) yield the same injection π. We conclude that

$$2^{n-4}(n-3)! = |\mathcal{T}_n| \leq \frac{m!}{(m-n)!},$$

Reformulating this inequality using Stirling's approximation now yields with $\beta(n) := \frac{f_s(n)}{n}$

$$2 - o(1) \leq \frac{\beta(n)^{\beta(n)}}{(\beta(n)-1)^{\beta(n)-1}}.$$

Consequently, $\beta(n) \geq (1 - o(1))\alpha$, where α is the unique real-valued solution to $\frac{\alpha^\alpha}{(\alpha-1)^{\alpha-1}} = 2$. This proves $f_s(n) = n \cdot \beta(n) \geq (1 - o(1))\alpha n$, which is the claim. ◀

4 Proof of Theorem 2.2 and Corollary 2.3

In the following, we outline the strategy which we have used to find a conflict collection of 23 stacked 11-vertex triangulations. Some details are omitted in this extended abstract but can be found in the full version [18]. In particular, we there provide detailed descriptions of all our programs – source codes are available on our supplemental website [17].

It is not hard to see that the embeddability of a given planar graph on a point set does not depend on the exact positions of the points but only on its *order type*, which is a combinatorial encoding of the point set determined by the orientations of triples of points in the point set. Thus, when testing for universality, it suffices to check embeddability of the corresponding graphs only on one representative point set for each order type.

4.1 Enumeration of Order Types

The database of all order types of up to $n = 11$ points was developed by Aurenhammer, Aichholzer, and Krasser [2, 3] (see also Krasser's dissertation [15]). The file for all order types of up to $n = 10$ points (each represented by a point set) is available online, while the file for $n = 11$ requires almost 100GB of storage and is available on demand [1]. In the full version, we also present an alternative and independent approach to enumerate all abstract order types from scratch and provide the corresponding source code [17].

4.2 Enumeration of Planar Graphs

To enumerate all non-isomorphic maximal planar graphs on 11 vertices (i.e, triangulations), we have used the plantri graph generator [7]. For various computations on graphs, such as filtering stacked triangulations, we have used SageMath [20].

4.3 Deciding Universality using a SAT Solver

For a given point set S and a planar graph $G = (V, E)$ we model a propositional formula in conjunctive normal form (CNF) which has a solution if and only if G can be embedded on S.

We have used variables to describe the vertex-to-point mapping and variables to describe whether the straight-line segments are "active" in a drawing. It is not hard to use clauses to assert that such a vertex-to-point mapping is bijective. Also it is easy to assert that, if two adjacent vertices u and v are mapped to points p and q, then the straight-line segment pq is active. For each pair of crossing straight-line segments pq and rs (dependent on the order type of the point set) at least one of the two segments is not allowed to be active.

We have implemented a C++ routine which, given a point set and a graph as input, creates an instance of the above described model and then uses the solver MiniSat [11] (see also [12]) to decide whether the graph admits a straight-line embedding.

4.4 Finding Conflict Collections – A Quantitive Approach

Before we actually tested whether a set of 11 points is 11-universal or not, we discovered a few necessary criteria for the point set, which can be checked much more efficiently. These considerations allowed a significant reduction of the total computation times.

Phase 1: Obviously, 11-universal point sets – if they exist – have to have triangular convex hulls. Secondly, the planar graph depicted in Figure 1 asserts an 11-universal set S to have a certain structure. Using these and a couple of other properties not mentioned here, only 293,114,696 of the 2,343,203,071 abstract order types on 11 points remain as candidates.

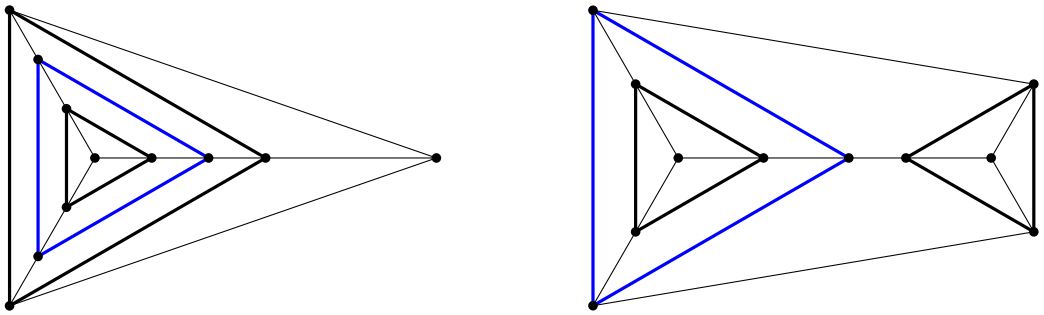


Figure 1 The two embeddings of a graph, which force the point set to have a certain layering.

Phase 2: For each of the remaining order types on 11 points from Phase 1, we have tested the embeddability of all maximal planar graphs on n vertices separately using a SAT-solver based approach. To speed up the computations we have used a priority queue: a graph which does not admit an embedding gets increased priority for other point sets to be tested first.

To keep the conflict collection as small as possible, we first filtered out all point sets which do not allow a simultaneous embedding of all planar graphs on 11 vertices with maximum degree 10. Only 278,530 of the 293,114,696 abstract order types remained (computation time about 100 CPU days).

At this point one can check with only a few CPU hours that the remaining 278,530 abstract order types are not 11-universal. Moreover, since some stacked triangulations on 11 vertices (e.g. the first graph from Figure 2) contain the graph from Figure 1 as a subgraph, the statement even applies to stacked triangulations and Corollary 2.3 follows.

Phase 3: We continued by testing the embeddability for each of the 434 stacked triangulations and each of the 278,530 remaining abstract order types (additional 35 CPU days). Based on this binary information, we formulated an integer program searching for a minimal set of triangulations without simultaneous embedding. Using the Gurobi solver [14], we managed to find a collection $\mathcal{G}$ of 11 stacked triangulations which cannot be embedded simultaneously; see Figure 2. By joining those stacked triangulations to the ones used in Phases 1 and 2, one already obtains a conflict collection of size 95.

Phases 4: To obtain smaller conflict collections, we again repeat the strategy from Phase 2, except that we test for the embeddability of the 11 stacked triangulations from the collection $\mathcal{G}$ obtained in Phase 3 instead of the 82 maximal planar graphs on 11 vertices with maximum degree 10. After 230 CPU days, our program had filtered out 17,533 of the 293,114,696 abstract order types obtained in Phase 1.

Phases 5: We proceeded as in Phase 3 and tested for each of the 434 stacked triangulations and each of the 17,533 order types from Phase 4, whether an embedding is possible (only 2 CPU days). Using the Gurobi solver, we managed to find a collection $\mathcal{H}$ of 12 stacked triangulations, which cannot be simultaneously embedded on those order types; see Figure 2.

Together with the 11 stacked triangulations from $\mathcal{G}$ we obtain a conflict collection of size 23, and the first part of Theorem 2.2 follows.

Phases 6: We have repeated our computations for the union of the two sets of point sets obtained in Phase 3 and Phase 5, respectively, in order to also improve the lower bounds. Using Gurobi, we obtained that any conflict collection consisting of 11-vertex planar graphs has size at least 17. This completes the proof of the second part of Theorem 2.2.

5 Discussion

In Section 3, we provided an improved lower bound for $f_p(n)$ and $f_s(n)$. However, the best known general upper bounds remain far from linear.

One could further proceed with the strategy from Section 4 to find even smaller conflict collection (if such exist). Also one could simply test whether all elements from the conflict collection are indeed necessary, or whether certain elements can be removed.

We also adapted our program to find all n-universal order types on n points for every $n \leq 10$, and hence could verify the results from [8, Table 1].

Unfortunately, we do not have an inductive argument for subsets/supersets of n-universal point sets, and thus the question for $n = 12, 13, 14$ remains open. However, based on computational evidence (see also [8, Table 1]), we strongly conjecture that no n-universal set of n points exists for $n \geq 11$.

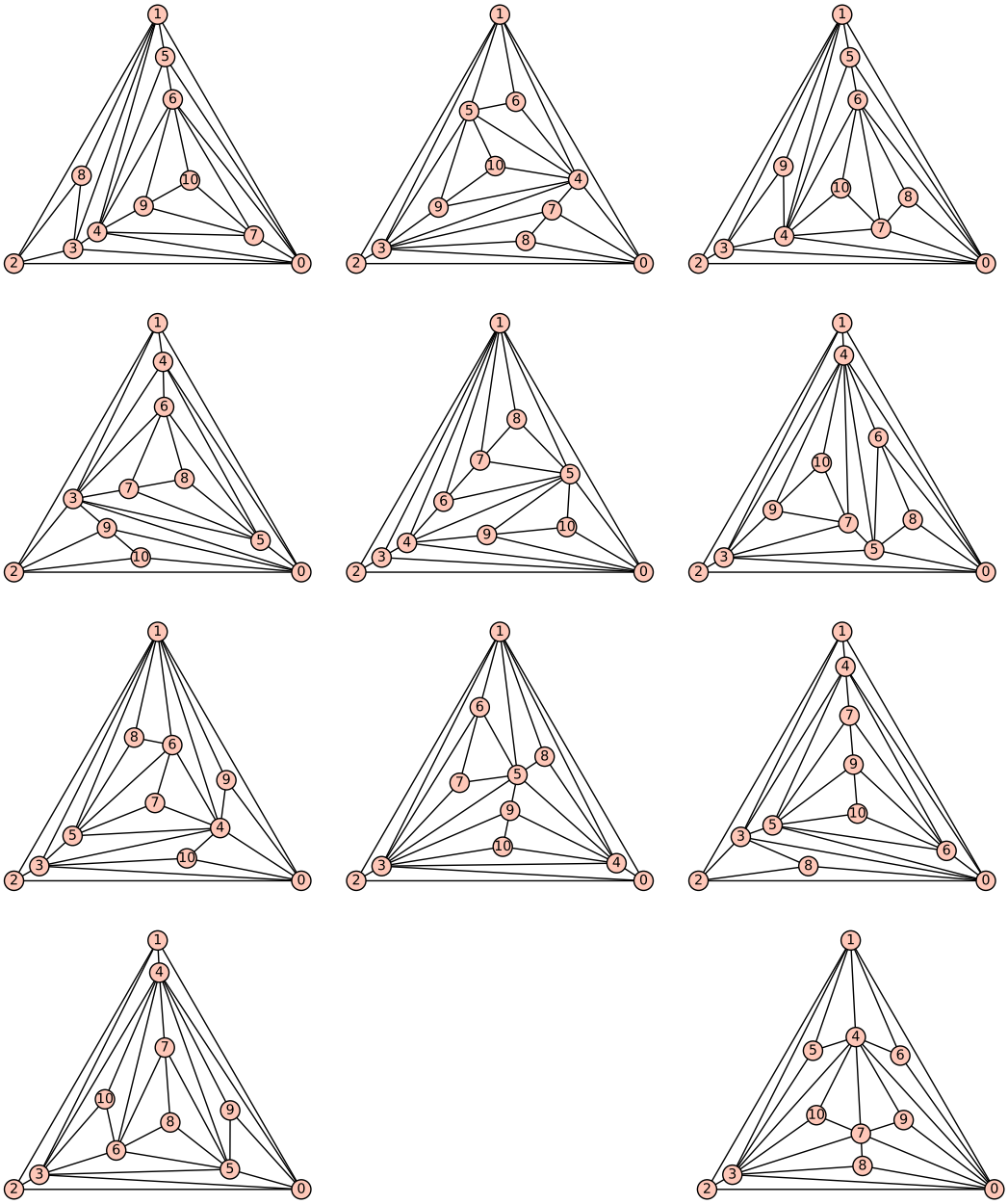


Figure 2 The 11 stacked triangulations from the conflict collection $\mathcal{G}$ obtained in Phase 3.

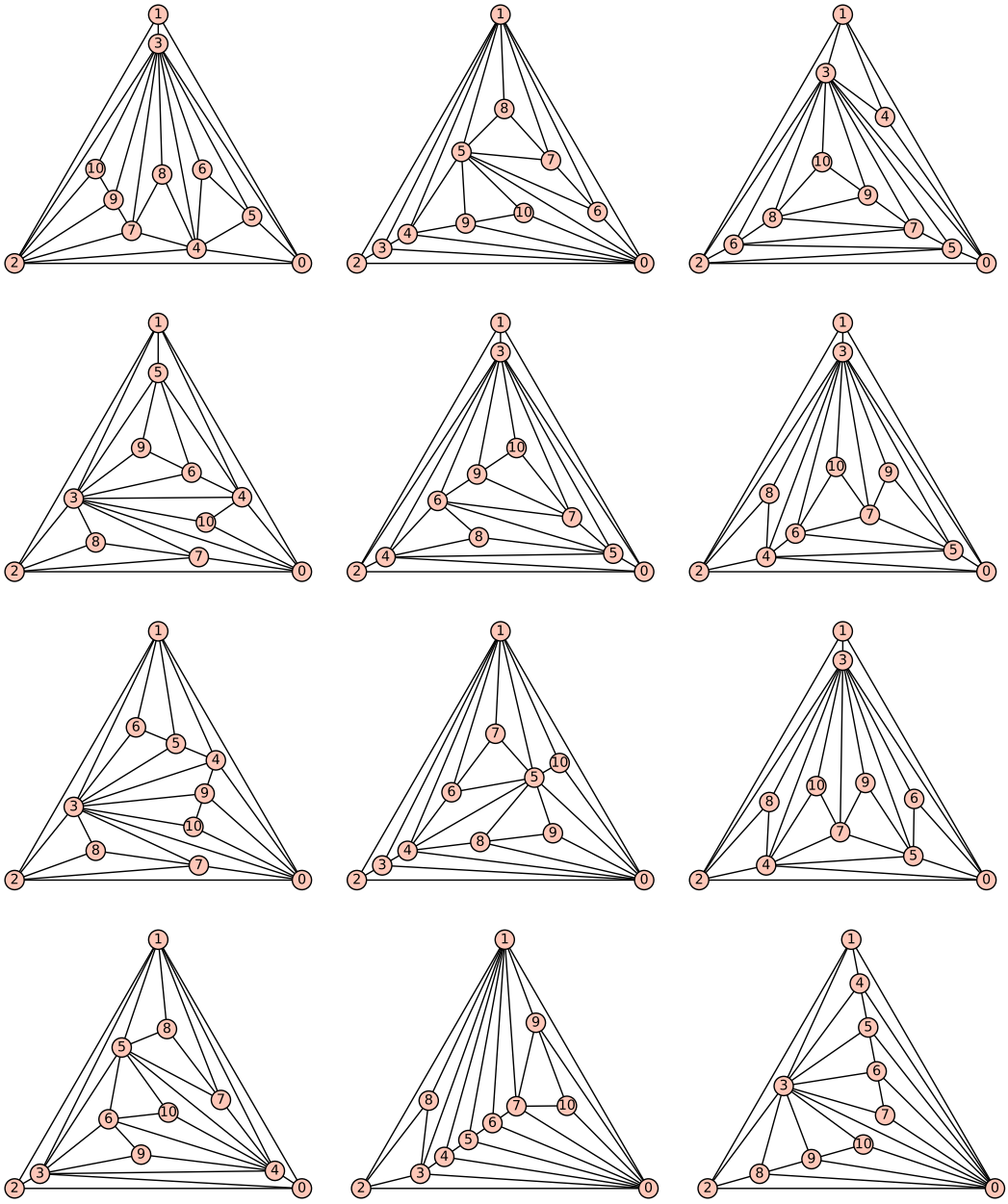


Figure 3 The 12 stacked triangulations from the conflict collection $\mathcal{H}$ obtained in Phase 5.

References

1 Oswin Aichholzer. Enumerating Order Types for Small Point Sets with Applications. http://www.ist.tugraz.at/aichholzer/research/rp/triangulations/ordertypes/.

2 Oswin Aichholzer, Franz Aurenhammer, and Hannes Krasser. Enumerating Order Types for Small Point Sets with Applications. *Order*, 19(3):265–281, 2002. doi:10.1023/A:1021231927255.

3 Oswin Aichholzer and Hannes Krasser. Abstract Order Type Extension and New Results on the Rectilinear Crossing Number. *Computational Geometry: Theory and Applications*, 36(1):2–15, 2006. doi:10.1016/j.comgeo.2005.07.005.

4 Michael J. Bannister, Zhanpeng Cheng, William E. Devanny, and David Eppstein. Superpatterns and Universal Point Sets. *Journal of Graph Algorithms and Applications*, 18(2):177–209, 2014. doi:10.7155/jgaa.00318.

5 Franz J. Brandenburg. Drawing planar graphs on $\frac{8}{9}n^2$ area. *Electronic Notes in Discrete Mathematics*, 31:37–40, 2008. doi:10.1016/j.endm.2008.06.005.

6 Peter Brass, Eowyn Cenek, Cristian A. Duncan, Alon Efrat, Cesim Erten, Dan P. Ismailescu, Stephen G. Kobourov, Anna Lubiw, and Joseph S.B. Mitchell. On simultaneous planar graph embeddings. *Computational Geometry*, 36(2):117–130, 2007. doi:10.1016/j.comgeo.2006.05.006.

7 Gunnar Brinkmann and Brendan D. McKay. Fast generation of some classes of planar graphs. *Electronic Notes in Discrete Mathematics*, 3:28–31, 1999. doi:10.1016/S1571-0653(05)80016-2.

8 Jean Cardinal, Michael Hoffmann, and Vincent Kusters. On Universal Point Sets for Planar Graphs. *Journal of Graph Algorithms and Applications*, 19(1):529–547, 2015. doi:10.7155/jgaa.00374.

9 Marek Chrobak and Howard J. Karloff. A Lower Bound on the Size of Universal Sets for Planar Graphs. *ACM SIGACT News*, 20(4):83–86, 1989. doi:10.1145/74074.74088.

10 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.

11 Niklas Eén and Niklas Sörensson. An extensible SAT-solver. In *Proceedings of Theory and Applications of Satisfiability Testing - SAT 2003*, pages 502–518, 2003. doi:10.1007/978-3-540-24605-3_37.

12 Niklas Eén and Niklas Sörensson. An Extensible SAT-solver. In Enrico Giunchiglia and Armando Tacchella, editors, *Theory and Applications of Satisfiability Testing: 6th International Conference, SAT 2003*, pages 502–518. Springer, 2004. doi:10.1007/978-3-540-24605-3_37.

13 Radoslav Fulek and Csaba D. Tóth. Universal point sets for planar three-trees. *Journal of Discrete Algorithms*, 30:101–112, 2015. doi:10.1016/j.jda.2014.12.005.

14 Gurobi Optimization, LLC. Gurobi Optimizer, 2018. http://www.gurobi.com.

15 Hannes Krasser. *Order Types of Point Sets in the Plane*. PhD thesis, Institute for Theoretical Computer Science, Graz University of Technology, Austria, 2003.

16 Maciej Kurowski. A $1.235n$ lower bound on the number of points needed to draw all n-vertex planar graphs. *Information Processing Letters*, 92(2):95–98, 2004. doi:10.1016/j.ipl.2004.06.009.

17 Manfred Scheucher. Webpage: Source Codes and Data for Universal Point Sets. http://page.math.tu-berlin.de/~scheuch/supplemental/universal_sets.

18 Manfred Scheucher, Hendrik Schrezenmaier, and Raphael Steiner. A Note On Universal Point Sets for Planar Graphs. http://arXiv.org/abs/1811.06482, 2018.

19 Walter Schnyder. Embedding Planar Graphs on the Grid. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 138–148. Society for Industrial and Applied Mathematics, 1990.

20 William A. Stein et al. *Sage Mathematics Software (Version 8.1)*. The Sage Development Team, 2018. `http://www.sagemath.org`.

On Disjoint Holes in Point Sets*†

Manfred Scheucher[1]

1 **Institut für Mathematik, Technische Universität Berlin, Germany**
{scheucher}@math.tu-berlin.de

Abstract

Given a set of points $S \subseteq \mathbb{R}^2$, a subset $X \subseteq S$, $|X| = k$, is called *k-gon* if all points of X lie on the boundary of the convex hull $\text{conv}(X)$, and *k-hole* if, in addition, no point of $S \setminus X$ lies in $\text{conv}(X)$. We use computer assistance to show that every set of 17 points in general position admits two *disjoint* 5-holes, that is, holes with disjoint respective convex hulls. This answers a question of Hosono and Urabe (2001).

In a recent article, Hosono and Urabe (2018) present new results on interior-disjoint holes – a variant, which also has been investigated in the last two decades. Using our program, we show that every set of 15 points contains two interior-disjoint 5-holes. Moreover, our program can also be used to verify that every set of 17 points contains a 6-gon within significantly smaller computation time than the original program by Szekeres and Peters (2006).

1 Introduction

A set of points in the Euclidean plane $S \subseteq \mathbb{R}^2$ is *in general position* if no three points lie on a common line. Throughout this paper all point sets are considered to be in general position. A subset $X \subseteq S$ of size $|X| = k$ is a *k-gon* if all points of X lie on the boundary of the convex hull of X. A classical result from the 1930s by Erdős and Szekeres asserts that, for fixed $k \in \mathbb{N}$, every sufficiently large point set contains a k-gon [12, 25]. They also constructed point sets of size 2^{k-2} with no k-gon. Recently, Suk [31] significantly improved the upper bound by showing that every set of $2^{k+o(k)}$ points contains a k-gon. However, the precise minimum number $g(k)$ of points needed to guarantee the existence of a k-gon is still unknown for $k \geq 7$ (cf. [32]).

In the 1970s, Erdős [11] asked whether every sufficiently large point set contains a *k-hole*, that is, a k-gon with no other points of S lying inside its convex hull. Harborth [17] showed that every set of 10 points contains a 5-hole and Horton [18] introduced a construction of large point sets without 7-holes. The question, whether 6-holes exist in sufficiently large point sets, remained open until 2007, when Nicolas [26] and Gerken [15] independently showed that point sets with large k-gons also contain a 6-hole (see also [33]). The currently best bound is by Koshelev [23], who showed that every set of 463 points contains a 6-hole. However, the largest set without 6-holes currently known has 29 points and was found by Overmars [27].

In 2001, Hosono and Urabe [19] started the investigation of disjoint holes, where two holes X_1, X_2 of a given point set S are said to be *disjoint* if their respective convex hulls are disjoint (that is, $\text{conv}(X_1) \cap \text{conv}(X_2) = \emptyset$). This led to the following question: What is the smallest number $h(k_1, \ldots, k_l)$ such that every set of $h(k_1, \ldots, k_l)$ points determines a k_i-hole for every $i = 1, \ldots, l$, such that the holes are pairwise disjoint [21]?

---

* The full version of this paper is available online [30].

† Research was supported by the DFG Grant FE 340/12-1. We thank Stefan Felsner, Linda Kleist, Felix Schröder, and Martin Balko for fruitful discussions and helpful comments. Thanks to Adrian Dumitrescu for pointing out the variant of interior-disjoint holes.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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 Sections 2 and 3, we summarize the current state of the art for two- and three-parametetric values and we present some new results that were obtained using computer-assistance. The basic idea behind our computer-assisted proofs is to encode point sets and disjoint holes only using triple orientations (see Section 4), and then to use a SAT solver to disprove the existence of sets with certain properties (see Section 5).

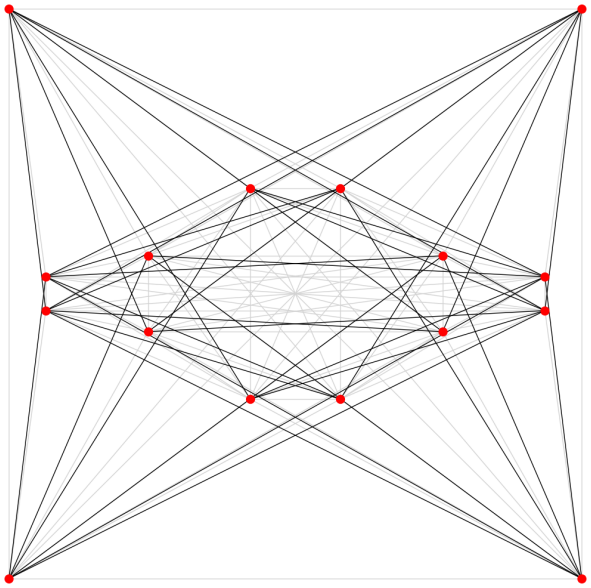
In the Final Remarks (Section 6) we outline how our SAT model can be adapted to tackle related questions on point sets. In particular, the program can be used to show that every set of 15 points contains two interior-disjoint 5-holes, and to prove $g(6) = 17$ with significantly smaller computation time than the original program from Szekeres and Peters [32].

2 Two Disjoint Holes

For two parameters, the value $h(k_1, k_2)$ has been determined for all $k_1, k_2 \leq 5$ except for $h(5, 5)$ [19, 20, 21, 5]. Table 1 summarizes the currently best bounds for two-parametric values. Concerning the value $h(5, 5)$, the best bounds are $17 \leq h(5, 5) \leq 19$. The lower bound $h(5, 5) \geq 17$ is witnessed by the set of 16 points with no two disjoint 5-holes (taken from Hosono and Urabe [21]), which is depicted Figure 1, and the upper bound $h(5, 5) \leq 19$ was shown by Bhattacharya and Das [6] by an elaborate case distinction.

	2	3	4	5
2	4	5	6	10
3		6	7	10
4			9	12
5				17*

Table 1 Values of $h(k_1, k_2)$ [19, 20, 21, 5]. The entry marked with star (*) is new.



```
  0   0
  0 270
140   0
140 270
  9 127
  9 143
131 127
131 143
 34 117
 34 153
106 117
106 153
 59  85
 59 185
 81  85
 81 185
```

Figure 1 A set of 16 points with no two disjoint 5-holes. This point set and the one by Hosono and Urabe [21, Figure 3] are of the same order type (see Section 4.1 for the definition of order type).

As our main result of this paper, we determine the precise value of $h(5,5)$. The proof is based on a SAT model which we later describe in Section 5.

▶ **Theorem 2.1** (Computer-assisted). *Every set of 17 points contains two disjoint 5-holes, hence* $h(5,5) = 17$.

We remark that the computations for verifying Theorem 2.1 take about two hours on a single 3GHz CPU using a modern SAT solver such as glucose [3] or picosat [7]. Moreover, we have verified the output of glucose and picosat with the proof checking tool DRAT-trim [34].

3 Three Disjoint Holes

For three parameters, most values $h(k_1,k_2,k_3)$ for $k_1,k_2,k_3 \leq 4$ and also the values $h(2,3,5) = 11$ and $h(3,3,5) = 12$ are known [21, 35]. Tables 2 and 3 summarize the currently best known bounds for three-parametric values.

	2	3	4
2	8	9	11
3		10	12
4			14

Table 2 Values of $h(k_1,k_2,4)$ [21, 35].

	2	3	4	5
2	10	11	11..14	17*
3		12	13..14	17..19*
4			15..17	17..23*
5				22*..27*

Table 3 Bounds for $h(k_1,k_2,5)$ [21, 35].

We now use Theorem 2.1 to derive new bounds on the value $h(k,5,5)$ for $k = 2,3,4,5$.

▶ **Corollary 3.1.** *We have*

$$h(2,5,5) = 17, \quad 17 \leq h(3,5,5) \leq 19, \quad 17 \leq h(4,5,5) \leq 23, \quad \text{and} \quad 22 \leq h(5,5,5) \leq 27.$$

Proof. To show $h(2,5,5) \leq 17$, observe that, due to Theorem 2.1, every set of 17 points contains two disjoint 5 holes that are separated by a line ℓ. By the pigeonhole principle there are at least 9 points on one of the two sides of such a separating line ℓ. Again, using a SAT instance similar to the one for Theorem 2.1, one can easily verify that every set of 9 points with a 5-hole also contains a 2-hole which is disjoint from the 5-hole. We remark that also the order type database of 9 points can be used to verify this statement.

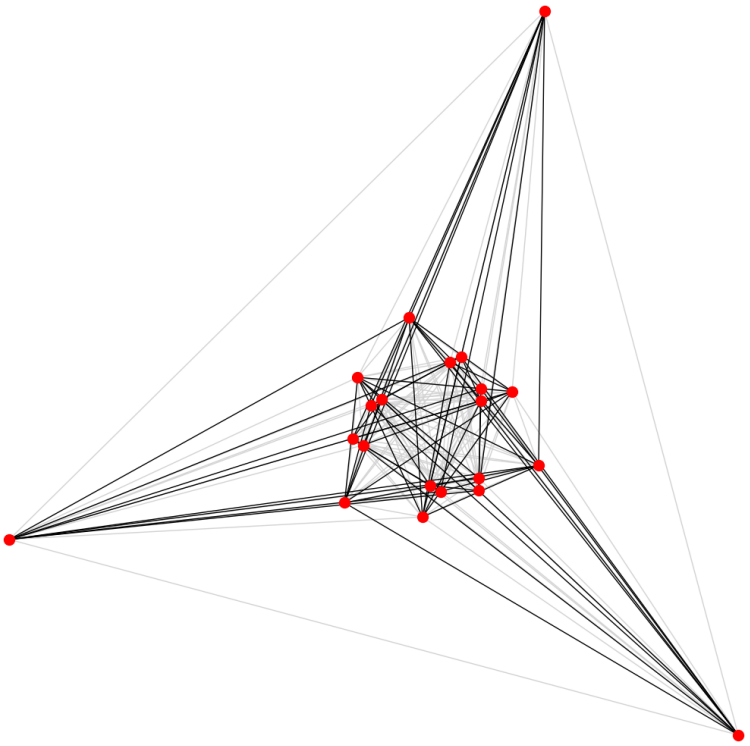
Similarly we show $h(3,5,5) \leq 19$: Every set of 19 points contains two disjoint 5-holes that are separated by a line ℓ. Now there are at least 10 points on one side of ℓ, and since $h(3,5) = 10$, there is a 3-hole and a 5-hole that are disjoint on that particular side.

An analogous argument shows $h(4,5,5) \leq 2 \cdot h(4,5) - 1 = 23$.

The point set from Figure 2 shows $h(5,5,5) > 21$, while $h(5,5,5) \leq h(5)+h(5,5) = 27$. ◀

4 Encoding with Triple Orientations

We describe how point sets and disjoint holes can be encoded only using triple orientations. This combinatorial description allows us to get rid of the actual point coordinates and to only consider a discrete parameter-space. This is essential for our SAT model of the problem.



0	161014
437034	595949
326347	343801
284425	294548
368806	311583
359850	306967
303825	276373
295136	271265
384946	285229
410465	282863
385025	275150
280383	244110
288858	238662
432159	221931
383508	211334
343366	205440
352134	200469
273710	191231
383027	201270
337326	179552
595182	0

Figure 2 A set of 21 points with no three disjoint 5-holes.

4.1 Triple Orientations

Given a set of points $S = \{s_1, \ldots, s_n\}$ with $s_i = (x_i, y_i)$, we say that the triple (a, b, c) is *positively (negatively) oriented* if

$$\chi_{abc} := \operatorname{sgn} \det \begin{pmatrix} 1 & 1 & 1 \\ x_a & x_b & x_c \\ y_a & y_b & y_c \end{pmatrix} \in \{-1, 0, +1\}$$

is positive (negative). Note that $\chi_{abc} = 0$ indicates collinear points, in particular, $\chi_{aaa} = \chi_{aab} = \chi_{aba} = \chi_{baa} = 0$. It is easy to see, that convexity is a combinatorial rather than a geometric property since k-gons can be described only by the relative position of the points: If the points $s_1, \ldots, s_k$ are the vertices of a convex polygon (ordered along the boundary), then, for every $i = 1, \ldots, k$, the cyclic order of the other points around s_i is $s_{i+1}, s_{i+2}, \ldots, s_{i-1}$ (indices modulo k). Similarly, one can also describe containment (and thus k-holes) only using relative positions: A point s_0 lies inside a convex polygon if the cyclic order around s_0 is precisely the order of the vertices along the boundary of the polygon.

To observe that the disjointness of two point sets can be described solely using triple orientations, suppose that a line ℓ separates point sets A and B. Then, for example by rotating ℓ, we can find another line ℓ' that contains a point $a \in A$ and a point $b \in B$ and separates $A \setminus \{a\}$ and $B \setminus \{b\}$. In particular, we have $\chi_{aba'} \leq 0$ for all $a' \in A$ and $\chi_{abb'} \geq 0$ for all $b' \in B$, or the other way round. Altogether, the existence of disjoint holes can be described solely using triple orientations.

Even though, for fixed $n \in \mathbb{N}$, there are uncountable possibilities to choose n points from the Euclidean plane, there are only finitely many equivalence classes of point sets when point

sets inducing the same orientation triples are considered equal. As introduced by Goodman and Pollack [16], these equivalence classes are called *order types*.

4.2 An Abstraction of Point Sets

Consider a point set $S = \{s_1, \dots, s_n\}$ where $s_1, \dots, s_n$ have increasing x-coordinates. Using the *unit paraboloid duality transformation*, which maps point $s = (a, b)$ to line $s^* : y = 2ax - b$, we obtain the arrangement of dual lines $S^* = \{s_1^*, \dots, s_n^*\}$, where the dual lines $s_1^*, \dots, s_n^*$ have increasing slopes. By the increasing x-coordinates and the properties of the unit paraboloid duality (see e.g. [24, Chapter 1.3]), the following three statements are equivalent:

(i) The points s_i, s_j, s_k are positively oriented.
(ii) The point s_k lies above the line $\overline{s_i s_j}$.
(iii) The intersection-point of the two lines s_i^* and s_j^* lies above the line s_k^*.

Due to Felsner and Weil [14] (see also [4]), 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 are in lexicographic order) changes its sign at most once. These conditions are the *signotope axioms*. It is worth to note that the signotope axioms are necessary conditions but not sufficient for point sets. There exist χ-configurations which fulfill the conditions above – so-called *abstract point sets*, *abstract order types*, *abstract oriented matroids (of rank 3)*, or *signotopes* – that are **not** induced by any point set, and in fact, deciding whether an abstract point set has a realizing point set is known to be $\exists\mathbb{R}$-complete (see e.g. [13]).

4.3 Increasing Coordinates and Cyclic Order

In the following, we see why we can assume, without loss of generality, that in every point set $S = \{s_1, \dots, s_n\}$ the following three conditions hold:

- the points $s_1, \dots, s_n$ have increasing x-coordinates,
- in particular, s_1 is an extremal point, and
- the points $s_2, \dots, s_n$ are sorted around s_1.

When modeling a computer program, one can use these constraints (which do not affect the output of the program) to restrict the search space and to possibly get a speedup. This idea, however, is not new and was already used for the generation of the *order type database*, which provides a complete list of all order types of up to 11 points [24, 1, 2].

▶ **Lemma 4.1.** *Let $S = \{s_1, \dots, s_n\}$ be a point set where s_1 is extremal and $s_2, \dots, s_n$ are sorted around s_1. Then there is a point set $\tilde{S} = \{\tilde{s_1}, \dots, \tilde{s_n}\}$ of the same order type as S (in particular, $\tilde{s_2}, \dots, \tilde{s_n}$ are sorted around $\tilde{s_1}$) such that $\tilde{s_1}, \dots, \tilde{s_n}$ have increasing x-coordinates.*

Proof. We can assume $s_1 = (0, 0)$ and $x_i, y_i > 0$ for $i \geq 2$ – otherwise we can apply an affine-linear transformation. Moreover, x_i/y_i is increasing for $i \geq 2$ since $s_2, \dots, s_n$ are sorted around s_1. Since S is in general position, there is an $\varepsilon > 0$ such that S and $S' := \{(0, \varepsilon)\} \cup \{s_2, \dots, s_n\}$ are of the same order type. We apply the projective transformation $(x, y) \mapsto (x/y, -1/y)$ to S' to obtain $\tilde{S}$. By the multilinearity of the determinant, we obtain

$$\det \begin{pmatrix} 1 & 1 & 1 \\ x_i & x_j & x_k \\ y_i & y_j & y_k \end{pmatrix} = y_i \cdot y_j \cdot y_k \cdot \det \begin{pmatrix} 1 & 1 & 1 \\ x_i/y_i & x_j/y_j & x_k/y_k \\ -1/y_i & -1/y_j & -1/y_k \end{pmatrix}.$$

Since the points in S' have positive y-coordinates, S' and $\tilde{S}$ have the same triple orientations. Moreover, as $\tilde{x_i} = {}^{x'_i}/_{y'_i}$ is increasing for $i \geq 1$, the set $\tilde{S}$ fulfills all desired properties. ◀

5 SAT Model

The basic idea to prove Theorem 2.1 is to assume – towards a contradiction – that a point set $S = \{s_1, \ldots, s_{17}\}$ with no two disjoint 5-holes exists. We formulate a SAT instance, where boolean variables indicate whether triples are positively or negatively oriented and clauses encode the necessary conditions described in Section 4. To be precise, we also have auxiliary variables, e.g., to indicate whether 4 points are in convex position and whether 3 points form a 3-hole. A detailled description of our SAT model can be found in the full version [30] and the source code of our python program is available online on our supplemental website [29].

Using a SAT solver we verify that the SAT instance has no solution and conclude that the point set S does not exist. This contradiction then completes the proof of Theorem 2.1.

It is folklore that satisfiability is NP-hard in general, thus it is challenging for SAT solvers to terminate in reasonable time for certain inputs of SAT instances. We now highlight the two crucial parts of our SAT model, which are indeed necessary for reasonable computation times: First, due to Lemma 4.1, we can assume without loss of generality that the points are sorted from left to right and also around the first point s_1. Second, we teach the solver that every set of 10 points gives a 5-hole, that is, $h(5) = 10$ [17]. By dropping either of these two constraints (which only give additional information to the solver and do not affect the solution space), none of the tested SAT solvers terminated within days.

6 Final Remarks

Interior-disjoint Holes: Two holes X_1, X_2 are called *interior-disjoint* if their respective convex hulls are interior-disjoint [10, 28, 9, 8, 22]. In a recent article, Hosono and Urabe [22] summarized the current status and presented some new results. By slightly adapting the SAT model from Section 5, we managed to show that every set of 15 points contains two interior-disjoint 5-holes; this further improves their result [22, Theorem 3].

Classical Erdős–Szekeres: The computation time for the computer assisted proof by Szekeres and Peters [32] for $g(6) = 17$ was about 1500 hours. By slightly adapting the model from Section 5 we have been able to confirm $g(6) = 17$ using glucose and DRAT-trim with about one hour of computation time.

References

1 Oswin Aichholzer, Franz Aurenhammer, and Hannes Krasser. Enumerating Order Types for Small Point Sets with Applications. *Order*, 19(3):265–281, 2002. doi:10.1023/A:1021231927255.

2 Oswin Aichholzer and Hannes Krasser. Abstract Order Type Extension and New Results on the Rectilinear Crossing Number. *Computational Geometry: Theory and Applications*, 36(1):2–15, 2006. doi:10.1016/j.comgeo.2005.07.005.

3 Gilles Audemard and Laurent Simon. Predicting Learnt Clauses Quality in Modern SAT Solvers. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, pages 399–404, 2009. http://ijcai.org/Proceedings/2009/.

4 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.

5 Bhaswar B. Bhattacharya and Sandip Das. On the Minimum Size of a Point Set Containing a 5-Hole and a Disjoint 4-Hole. *Studia Scientiarum Mathematicarum Hungarica*, 48(4):445–457, 2011. doi:10.1556/SScMath.2011.1173.

6 Bhaswar B. Bhattacharya and Sandip Das. Disjoint empty convex pentagons in planar point sets. *Periodica Mathematica Hungarica*, 66(1):73–86, 2013. doi:10.1007/s10998-013-9078-z.

7 Armin Biere. PicoSAT Essentials. *Journal on Satisfiability, Boolean Modeling and Computation (JSAT)*, 4:75–97, 2008. http://satassociation.org/jsat/index.php/jsat/article/view/45.

8 Ahmad Biniaz, Anil Maheshwari, and Michiel H. M. Smid. Compatible 4-Holes in Point Sets, 2017. arXiv:1706.08105.

9 Javier Cano, Alfredo García, Ferran Hurtado, Toshinori Sakai, Javier Tejel, and Jorge Urrutia. Blocking the k-Holes of Point Sets in the Plane. *Graphs and Combinatorics*, 31(5):1271–1287, 2015. doi:10.1007/s00373-014-1488-z.

10 Olivier Devillers, Ferran Hurtado, Gyula Károlyi, and Carlos Seara. Chromatic variants of the Erdős–Szekeres theorem on points in convex position. *Computational Geometry*, 26(3):193–208, 2003. doi:10.1016/S0925-7721(03)00013-0.

11 Paul Erdős. Some more problems on elementary geometry. *Australian Mathematical Society Gazette*, 5:52–54, 1978. http://www.renyi.hu/~p_erdos/1978-44.pdf.

12 Paul Erdős and George Szekeres. A combinatorial problem in geometry. *Compositio Mathematica*, 2:463–470, 1935. http://www.renyi.hu/~p_erdos/1935-01.pdf.

13 Stefan Felsner and Jacob E. Goodman. Pseudoline Arrangements. In Toth, O'Rourke, and Goodman, editors, *Handbook of Discrete and Computational Geometry*. CRC Press, third edition, 2018. doi:10.1201/9781315119601.

14 Stefan Felsner and Helmut Weil. Sweeps, Arrangements and Signotopes. *Discrete Applied Mathematics*, 109(1):67–94, 2001. doi:10.1016/S0166-218X(00)00232-8.

15 Tobias Gerken. Empty Convex Hexagons in Planar Point Sets. *Discrete & Computational Geometry*, 39(1):239–272, 2008. doi:10.1007/s00454-007-9018-x.

16 Jacob E. Goodman and Richard Pollack. Multidimensional Sorting. *SIAM Journal on Computing*, 12(3):484–507, 1983. doi:10.1137/0212032.

17 Heiko Harborth. Konvexe Fünfecke in ebenen Punktmengen. *Elemente der Mathematik*, 33:116–118, 1978. In German, http://www.digizeitschriften.de/dms/img/?PID=GDZPPN002079801.

18 Joseph D. Horton. Sets with no empty convex 7-gons. *Canadian Mathematical Bulletin*, 26:482–484, 1983. doi:10.4153/CMB-1983-077-8.

19 Kiyoshi Hosono and Masatsugu Urabe. On the number of disjoint convex quadrilaterals for a planar point set. *Computational Geometry*, 20(3):97–104, 2001. doi:10.1016/S0925-7721(01)00023-2.

20 Kiyoshi Hosono and Masatsugu Urabe. On the Minimum Size of a Point Set Containing Two Non-intersecting Empty Convex Polygons. In *Proceedings of the Japanese Conference on Discrete and Computational Geometry (JCDCG 2004)*, volume 3742 of *LNCS*, pages 117–122. Springer, 2005. doi:10.1007/11589440_12.

21 Kiyoshi Hosono and Masatsugu Urabe. A Minimal Planar Point Set with Specified Disjoint Empty Convex Subsets. In *Kyoto International Conference on Computational Geometry and Graph Theory (KyotoCGGT 2007)*, volume 4535 of *LNCS*, pages 90–100. Springer, 2008. doi:10.1007/978-3-540-89550-3_10.

22 Kiyoshi Hosono and Masatsugu Urabe. Specified holes with pairwise disjoint interiors in planar point sets. *AKCE International Journal of Graphs and Combinatorics*, 2018. In press. doi:10.1016/j.akcej.2018.08.003.

23 Vitalii A. Koshelev. On Erdős–Szekeres problem for empty hexagons in the plane. *Modelirovanie i Analiz Informatsionnykh Sistem*, 16(2):22–74, 2009. In Russian, http://mi.mathnet.ru/eng/mais52.

24 Hannes Krasser. *Order Types of Point Sets in the Plane*. PhD thesis, Institute for Theoretical Computer Science, Graz University of Technology, Austria, 2003.

25 Jiří Matoušek. Convex Independent Subsets. In *Lectures on Discrete Geometry*, pages 29–39. Springer, 2002. doi:10.1007/978-1-4613-0039-7_3.

26 Carlos M. Nicolas. The Empty Hexagon Theorem. *Discrete & Computational Geometry*, 38(2):389–397, 2007. doi:10.1007/s00454-007-1343-6.

27 Mark Overmars. Finding Sets of Points without Empty Convex 6-Gons. *Discrete & Computational Geometry*, 29(1):153–158, 2002. doi:10.1007/s00454-002-2829-x.

28 Toshinori Sakai and Jorge Urrutia. Covering the convex quadrilaterals of point sets. *Graphs and Combinatorics*, 23(1):343–357, 2007. doi:10.1007/s00373-007-0717-0.

29 Manfred Scheucher. Webpage: On Disjoint Holes in Point Sets. http://page.math.tu-berlin.de/~scheuch/supplemental/5holes/disjoint_holes/.

30 Manfred Scheucher. On Disjoint Holes in Point Sets, 2018. arXiv:1807.10848.

31 Andrew Suk. On the Erdős-Szekeres convex polygon problem. *Journal of the American Mathematical Society*, 30:1047–1053, 2017. doi:10.1090/jams/869.

32 George Szekeres and Lindsay Peters. Computer solution to the 17-point Erdős-Szekeres problem. *Australia and New Zealand Industrial and Applied Mathematics*, 48(2):151–164, 2006. doi:10.1017/S144618110000300X.

33 Pavel 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. http://bookstore.ams.org/conm-453.

34 Nathan Wetzler, Marijn J. H. Heule, and Warren A. Hunt. DRAT-trim: Efficient Checking and Trimming Using Expressive Clausal Proofs. In Carsten Sinz and Uwe Egly, editors, *Theory and Applications of Satisfiability Testing – SAT 2014*, pages 422–429. Springer, 2014. doi:10.1007/978-3-319-09284-3_31.

35 X. S. You and X. L. Wei. On the Minimum Size of a Point Set Containing a 5-Hole and Double Disjoint 3-Holes. *Mathematical Notes*, 97(5):951–960, 2015. doi:10.1134/S0001434615050314.

Erdős-Szekeres-Type Games*

Oswin Aichholzer[1], José-Miguel Díaz-Báñez[2], Thomas Hackl[1], David Orden[3], Alexander Pilz[1], Inmaculada Ventura[2], and Birgit Vogtenhuber[1]

1 **Institute of Software Technology, Graz University of Technology, Austria.**
[oaich|thackl|apilz|bvogt]@ist.tugraz.at
2 **Department of Applied Mathematics II, University of Seville, Spain.**
[dbanez|iventura]@us.es
3 **Departamento de Física y Matemáticas, Universidad de Alcalá, Spain.**
david.orden@uah.es

Abstract

We consider several combinatorial games, inspired by the Erdős-Szekeres theorem that states the existence of a convex k-gon in every sufficiently large point set. Two players take turns to place points in the Euclidean plane and the game is over as soon as the first k-gon appears. In the Maker-Maker setting the player who placed the last point wins, while in the Avoider-Avoider version this player loses. Combined versions like Maker-Breaker are also possible. Moreover, variants can be obtained by considering that (1) the points to be placed are either uncolored or bichromatic, (2) both players have their own color or can play with both colors, (3) the k-gon must be empty of other points, or (4) the k-gon has to be convex.

1 Introduction

A central topic in combinatorial game theory are sequential games with perfect information. These are often two-player games that have positions, in which the players take turns changing these positions (in a defined way) to eventually achieve a specific winning position. Perfect information means that the state of the game (the current position and usually also the history of all moves so far) and the set of all possible moves is known to both players at any time. This class includes Chess or Go, but also easy-to-analyze games like Tic-tac-toe. The formal analysis of concrete games sometimes reveals challenging mathematical problems while still having substantial recreational value. Along these lines, we study a class of combinatorial games related to a well-known result in combinatorics, the Erdős-Szekeres theorem.

▶ **Theorem 1.1** (Erdős and Szekeres [5]). *For every integer $k \geq 3$, there exists an $n(k)$ s.t. any set of at least $n(k)$ points in general position has a k-element subset in convex position.*

Decades later, Erdős [4] posed the problem of determining the smallest integer $h(k)$, if it exists, such that any set S of at least $h(k)$ points in general position contains a convex k-hole, that is, a convex k-gon that does not contain any point of S in its interior. Finding

* J.-M. D.-B. and I. V. supported by Project GALGO (Spanish Ministry of Economy and Competitiveness and MTM2016-76272-R AEI/FEDER,UE). D. O. partially supported by Project MTM2017-83750-P of the Spanish Ministry of Science (AEI/FEDER, UE). A. P. supported by a Schrödinger fellowship of the Austrian Science Fund (FWF): J-3847-N35. B. V. 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.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 19–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

exact lower bounds on the number of points to guarantee the existence of convex k-gons (see [6, 12, 16]) and k-holes (see [10, 11, 9, 15, 17]) has been a long line of research.

A simple polygon spanned by points of different color classes is called *monochromatic* if all its vertices have the same color. Two results of interest for our games are the following.

▶ **Theorem 1.2** (Devillers et al. [3]). *Every bichromatic set of* 9 *points in general position contains an empty monochromatic triangle.*

▶ **Theorem 1.3** (Devillers et al. [3]). *There exist arbitrarily large bichromatic point sets in general position without any convex monochromatic 5-hole.*

The question of whether every bichromatic point set of sufficiently many points contains a convex monochromatic 4-hole is still open, see [2, 8, 18]. However, we have:

▶ **Theorem 1.4** (Aichholzer et al. [1]). *Every bichromatic set of at least* 5044 *points contains a (possibly non-convex) monochromatic* 4*-hole.*

In this context, the following combinatorial game comes to mind immediately: Two players alternately place one point at a time in the plane until a k-gon appears, with the points required to be pairwise distinct and in general position. Different games arise depending on whether the points are colored or not. We will call the players *player A* and *player B* if both players can play both colors (say, red and green), and call them *Red* and *Green* if each player is only allowed to use a specific color. Players A and Red will always start a game, and are thus called *first player*. Consequently, we call players B and Green *second player*. In this paper we study the following scenarios:

1. The Erdős-Szekeres Avoider-Avoider game (ESAA), in which the goal is to avoid the formation of a convex monochromatic k-gon/k-hole. The player who places the last point that results in a convex k-gon/k-hole loses.
2. The Erdős-Szekeres Maker-Maker game (ESMM), in which the goal is to build a convex monochromatic k-gon/k-hole and the player who places the last point forming the convex k-gon/k-hole wins.
3. The Erdős-Szekeres Maker-Breaker game (ESMB), in which the Red player aims to make a red convex/general k-hole and the goal of the Green player is to prevent this.

Preliminaries. In combinatorial game theory, a sequential game is a game where one player performs their action before the next player performs theirs. The players act in turns, where one action of a player is called a *ply*. The term "ply" is used to avoid confusion when one might otherwise use the term move (or turn). All considered or constructed point sets will be in general position, i.e., no three points of a set are on a common line.

The Avoider-Avoider *uncolored* game for *convex* 5*-gons* and 5*-holes* was shown by Kolipaka and Govindarajan [13] to be a win of B. We provide a shorter proof via a simple strategy for B in Section 2. In Section 3, the *bichromatic* ESAA and ESMM games are addressed both for A–B and Red–Green players, as well as the ESMB game for Red–Green players.

2 Uncolored variants

In the setting in which the points do not have different colors, the symmetric game types Avoider-Avoider and Maker-Maker immediately come to mind. Observe that the Maker-Maker variant for convex k-gons/k-holes corresponds to the Avoider-Avoider variant for $k-1$. Here, we thus discuss the Avoider-Avoider variant. (For holes, it would also make sense to consider Maker–Breaker, at least for $k \geq 7$, as there are point sets without convex 7-hole [11].)

▶ **Game 1.** *Two players A and B alternate in placing a single (uncolored) point in the plane with the restriction that no three points are on a line. The player who places a point resulting in a convex k-gon/k-hole loses the game.*

The cases for $k \in \{3, 4\}$ are trivial. For $k = 5$, Kolipaka and Govindarajan [13] showed that player B wins at the 9th ply. We provide a simpler strategy leading to a shorter proof (the original proof results in a paper of 28 pages), which works for both 5-holes and 5-gons.

▶ **Theorem 2.1** (Kolipaka and Govindarajan [13]). *For $k = 5$, player B wins Game 1 at ply 9.*

Proof (sketch). We start by showing the result for 5-holes. Player B will use the *point reflection strategy*: After player A placed the first point, player B chooses a different point acting as center of symmetry, around which B mimics the plies of player A. We can argue that after 6 plies, this strategy leads to mostly equivalent configurations consisting of a parallelogram with two points inside (having central symmetry); see Figure 1 (a) and (b).

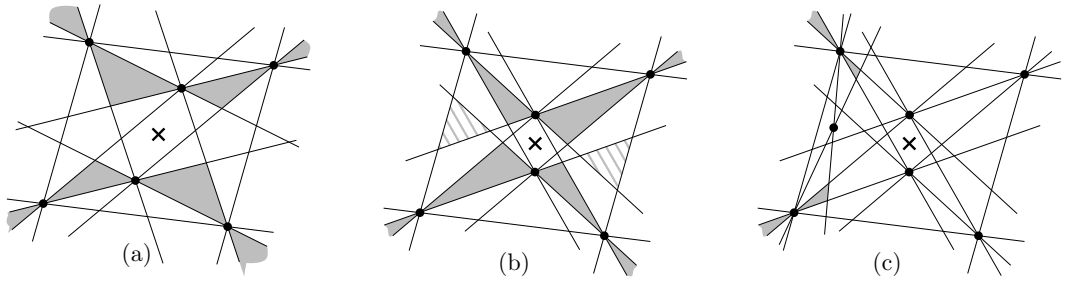


Figure 1 Configurations of six points with central symmetry and no convex 5-hole/5-gon.

The next point of player A has to be placed in a gray or hatched region, as otherwise a convex 5-hole occurs. If A plays in any gray region, then after ply 8 we get the situation depicted in Figure 2 (a). If A plays in a hatched region of Figure 1 (b), then the setting of Figure 2 (b) shows up. In both cases there is no convex 5-hole. Moreover, no additional point can be placed without generating a convex 5-hole (although there exist sets of 9 points in general position without any convex 5-hole [10]). Thus player B wins in ply 9.

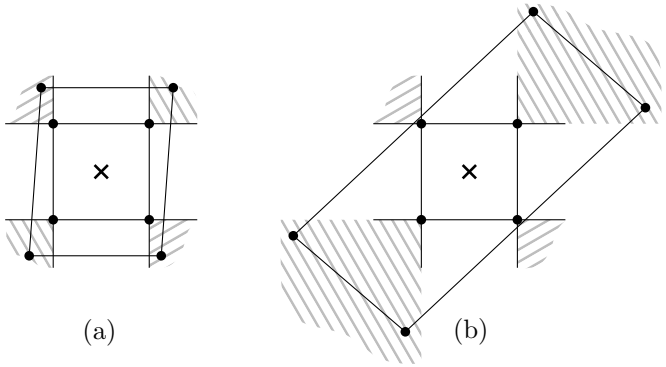


Figure 2 Any configuration without convex 5-hole must have the four points of the outer parallelogram in the indicated regions. Situation (b) cannot occur when avoiding a convex 5-gon.

For 5-gons, the arguments are analogous up to ply 6 and also until the end if player A places a point in a gray region of Figure 1 (a) or (b) in ply 7. So assume that player A places a point in a hatched region of Figure 1(b), as depicted in Figure 1(c). Then in ply 8

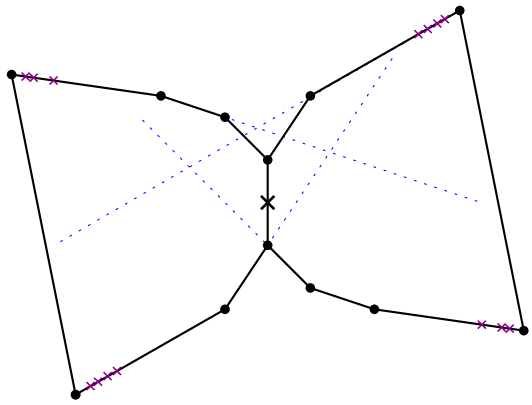


Figure 3 Counterexample for the point reflection strategy for k-holes for $k = 7$. The two points closest to the center are the last to be placed, where the last one causes the formation of two 7-holes. The small purple crosses indicate how to generalize this example for $k \geq 7$.

player B would produce a 5-gon with the point reflection strategy and hence has to make a different move; see again Figure 2 (b). However, in this case Player B can put a point in any of the grey regions indicated in Figure 1 (c). As every set of nine points in general position contains a 5-gon [12], player B wins in ply 9. ◀

For any even k, the point reflection strategy cannot work for player B: if player A places all points in convex position, then B creates a convex k-hole. For any odd k, a convex k-gon can never be centrally symmetric. Hence, in centrally symmetric point sets they come in pairs. Figure 3 shows that for odd $k > 5$ the strategy does not work either.

3 Bichromatic variants

In bichromatic games, either each player is assigned a color which they can use (then we call the players Red and Green), or both players (called A and B) may use both colors. In either case, the goal is to make or avoid a monochromatic k-gon or k-hole. A hole must not contain any points in its interior, regardless of their color. We consider the game with players Red and Green only for k-holes, as for k-gons the different colors do not influence each other.

3.1 Avoider–Avoider

We first consider the two-colored Avoider–Avoider setting in both versions, players A–B and Red–Green. We show that, for any version, the second player can avoid losing. For triangles, we provide upper bounds on the number of plies until the second player wins.

▶ **Game 2.** *In both versions (players Red–Green and A–B), the players alternate in placing a single point of one of two colors in the plane, avoiding collinear point triples. The goal for each player is to avoid the formation of a (general or convex) monochromatic k-hole. The player who placed the last point forming such a k-hole loses the game.*

The next theorem follows from the point reflection strategy with color-inversion.

▶ **Theorem 3.1.** *Players Green and B can avoid losing the respective variants of Game 2.*

By Theorem 1.3, there exist arbitrarily large sets without convex monochromatic 5-holes. It is open whether every sufficiently large bichromatic set has a convex monochromatic 4-hole [1], so we do not know if the game is finite for convex monochromatic k-holes, $k \geq 4$.

For $k = 3$, every bichromatic point set of at least 9 points contains an empty monochromatic triangle (Theorem 1.2). Thus, Theorem 3.1 implies a 9-ply winning strategy for the second player. We further show the following (proofs are omitted due to space constrains).

▶ **Theorem 3.2.** *Player B can win the bichromatic A–B Avoider–Avoider empty monochromatic triangle game at latest after the 9th ply, even if the point set must be in strong general position*[1]*. Player A can prevent to lose before the 9th ply.*

▶ **Theorem 3.3.** *Player Green can win the bichromatic Red–Green Avoider–Avoider empty monochromatic triangle game at latest after the 7th ply.*

For $k = 4$, the largest known point set not containing any convex monochromatic 4-hole has 46 points [14]. It can be modified (changing also the order type) to be centrally symmetric, with symmetry pairs having inverted colors [7]. From the latter set it follows that, if the second player mirrors the moves of the first player, the game might take at least 47 plies. For general monochromatic 4-holes, there is a set of 22 points not containing any of them [7], but without such a special symmetry. Both sets have the same number of red and green points. As the second player always has a strategy to not lose, the following questions arise.

▶ **Question 1.** *Does the second player have a winning strategy for monochromatic 4-holes by placing points centrally symmetric and color-inverted? What about for k-holes for $k > 4$?*

▶ **Question 2.** *Does every large enough centrally symmetric color-inverted bichromatic point set contain a convex monochromatic 4-hole?*

3.2 Maker–Maker

In the Maker–Maker variant the goal is to be the first to obtain a monochromatic k-hole. For players A and B, both must avoid a monochromatic $(k-1)$-hole; for convex k-holes, this is also sufficient. For general 4-holes, player A always makes a monochromatic triangle (empty or not) in the 5th ply. Then, player B can add another point inside this triangle as in Figure 4 (left) and produce a non-convex monochromatic 4-hole, i.e., wins in ply 6. For $k = 5$ we do not know any upper bound on the number of plies, convex or general.

Figure 4 Non-convex red 4-hole and Red–Green Maker–Maker after 7 plies for convex red 4-hole.

For players Red–Green, we argue that Red wins for $k = 3, 4$. A win in ply 5 for $k = 3$ is obvious. For $k = 4$, Red makes an empty red triangle Δ_R with the first three red points. In ply 6 player Green must place a green point inside Δ_R. Red can make two interior-disjoint empty red triangles by placing the fourth red point inside Δ_R and also inside the green triangle, see Figure 4 (right). This is already a general red 4-hole. For convex 4-holes, Green

[1] No two supporting lines of distinct point pairs are parallel; no three intersect in a common point; etc.

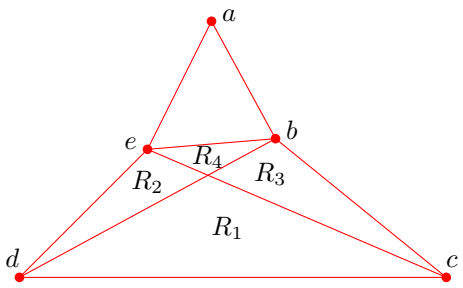


Figure 5 A can build a red general 5-hole in 9 plies.

can neither extend the green triangle nor block both of the two empty red triangles. Thus, Red wins in ply 9. We also show the following.

▶ **Lemma 3.4.** *Player Red can always build a red (general) 5-hole as depicted in Figure 5 in 9 plies, that is, with the minimum number of 5 red points.*

▶ **Question 3.** *For players Red–Green, is there a winning strategy for convex monochromatic 5-holes? For which of the two players? How about $k > 5$?*

3.3 Red–Green Maker–Breaker

In this section we consider an asymmetric game, where the two players have different goals.

▶ **Game 3.** *Two players Red and Green alternate in placing a single point of their color, avoiding collinearities. The goal for player Red is to make a red k-hole. Player Green wants to block Red from doing so. Green k-holes do not matter.*

The following result can be shown using the idea sketched in Figure 6.

▶ **Theorem 3.5.** *In a Red–Green Maker–Breaker game, the Maker can always build a red (general) k-hole by placing k points.*

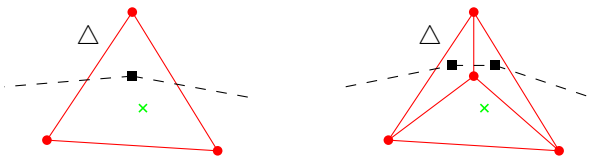


Figure 6 The Maker builds a chain of empty triangles (indicated by dashed lines and squares). When the Breaker places a green point (cross) inside such a triangle △, the Maker places a red point inside △, thereby repairing the chain and increasing its length by one.

From Lemma 3.4 we derive the following statements.

▶ **Proposition 3.6.** *The Maker can always build a red convex 4-hole by placing 5 red points.*

▶ **Theorem 3.7.** *The Maker can always build a red convex 5-hole by placing 8 red points.*

References

1 Oswin Aichholzer, Thomas Hackl, Clemens Huemer, Ferran Hurtado, and Birgit Vogtenhuber. Large bichromatic point sets admit empty monochromatic 4-gons. *SIAM J. Discrete Math.*, 23(4):2147–2155, 2010. doi:10.1137/090767947.

2 Peter Brass. Empty monochromatic fourgons in two-colored point sets. *Combinatorics*, 14, 2004.

3 Olivier Devillers, Ferran Hurtado, Gyula Károlyi, and Carlos Seara. Chromatic variants of the Erdsős-Szekeres theorem on points in convex position. *Comput. Geom.*, 26(3):193–208, 2003. doi:10.1016/S0925-7721(03)00013-0.

4 Paul Erdős. Some more problems on elementary geometry. *Austral. Math. Soc. Gaz.*, 5:52–54, 1978.

5 Paul Erdős and George Szekeres. A combinatorial problem in geometry. *Compositio Math.*, 2:463–470, 1935.

6 Paul Erdős and George Szekeres. On some extremum problems in elementary geometry. *Ann. Univ. Sci. Budapest. Eőtvős Sect. Math.*, 3–4, 1961. Reprinted in: Paul Erdős: The Art of Counting. Selected Writings (J. Spencer, ed.), pp. 680-689, MIT Press, Cambridge, MA, 1973.

7 EuroGiga. The Point Set Zoo. Retreived: Jan. 3, 2019. URL: http://www.eurogiga-compose.eu/posezo.php.

8 Erich Friedmann. 30 two-colored points with no empty monochromatic fourgons. *Combinatorics*, 14, 2004.

9 Tobias Gerken. Empty convex hexagons in planar point sets. *Discrete Comput. Geom.*, 39(1-3):239–272, 2008. doi:10.1007/s00454-007-9018-x.

10 Heiko Harborth. Konvexe Fünfecke in ebenen Punktmengen. *Elemente Math.*, 33:116–118, 1978. In German.

11 Joseph D. Horton. Sets with no empty convex 7-gons. *Canad. Math. Bull.*, 26:482–484, 1983.

12 J. D. Kalbfleisch, J. G. Kalbfleisch, and R. G. Stanton. A combinatorial problem on convex regions. pages 180–188, 1970. Congr. Numer. 1 (1970).

13 Parikshit Kolipaka and Sathish Govindarajan. Two player game variant of the Erdős-Szekeres problem. *Discrete Mathematics & Theoretical Computer Science*, 15(3):73–100, 2013. URL: http://dmtcs.episciences.org/620.

14 Vitaliy Koshelev. On Erdős–Szekeres problem and related problems. *ArXiv E-prints*, 2009. https://arxiv.org/abs/0910.2700.

15 Carlos M. Nicolas. The empty hexagon theorem. *Discrete Comput. Geom.*, 38(2):389–397, 2007. doi:10.1007/s00454-007-1343-6.

16 Géza Tóth and Pavel Valtr. Note on the Erdős–Szekeres theorem. *Discrete Comput. Geom.*, 19(3):457–459, 1998. doi:10.1007/PL00009363.

17 Pavel Valtr. On empty hexagons. In J. E. Goodman, J. Pach, and R. Pollack, editors, *Surveys on Discrete and Computational Geometry, Twenty Years Later*, pages 433–441. AMS, New York, 2008.

18 Rob Van Gulik. 32 two-colored points with no empty monochromatic convex fourgons. *Geombinatorics*, 15:32–33, 2004.

Approximating the Earth Mover's Distance between sets of points and line segments*

Marc van Kreveld[1], Frank Staals[1], Amir Vaxman[1], and Jordi L. Vermeulen[1]

1 **Department of Information and Computing Sciences, Utrecht University**
{m.j.vankreveld;f.staals;a.vaxman;j.l.vermeulen}@uu.nl

Abstract

We show that a $(1+\varepsilon)$-approximation algorithm exists for the Earth Mover's Distance between a set of n points and set of n line segments with equal total weight. Our algorithm runs in $O\left(\frac{n^6}{\varepsilon^2}\log^2\left(\frac{1}{\varepsilon}\right)\log^2\left(\frac{n^2}{\varepsilon}\log\frac{1}{\varepsilon}\right)\right)$ time.

1 Introduction

The Earth Mover's Distance (EMD) is a metric that is widely used in fields such as image retrieval [13], shape matching [5, 8, 16] and mesh reconstruction [3]. It models two sets A and B as distributions of mass, and takes their distance $\mathcal{D}(A,B)$ to be the minimum cost of transforming one distribution into the other, where cost is measured by the amount of mass moved multiplied by the distance over which it is moved. More formally,

$$\mathcal{D}(A,B) = \inf_{\mu\in M}\int_A\int_B d(a,b)\cdot\mu(a,b)\;da\,db \tag{1}$$

where M is the set of all mappings of mass between A and B. In the case where A and B are sets of (weighted) points, we can rewrite this as

$$\mathcal{D}(A,B) = \min_{\mu\in M}\sum_{a\in A}\sum_{b\in B} d(a,b)\cdot\mu(a,b) \tag{2}$$

For unweighted point sets, the solution can be obtained by solving an assignment problem; for weighted point sets, this is an instance of a min cost max flow problem.

In this work, we consider the case where A is a set of weighted points and B is a set of line segments in $\mathbb{R}^2$. We provide a polynomial-time algorithm that gives a $(1+\varepsilon)$-approximation to the Earth Mover's Distance between A and B, and also gives an assignment of mass that realises this cost. To our knowledge, this is the first combinatorial algorithm with a provable approximation ratio for the Earth Mover's Distance when the objects are continuous rather than discrete points.

2 Related work

The general problem of optimally moving a distribution of mass was first described by Monge in 1781 [11], and was reformulated by Kantorovich in 1942 [6]. It is known as the Earth Mover's Distance due to the analogy of moving piles of dirt around; it is also known as the 1-Wasserstein distance, and is a special case of the more general optimal transport problem.

* This research was supported by the Netherlands Organisation for Scientific Research under project number 612.001.651.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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 a full treatment of the problem's history and connections to other areas of mathematics, the reader is referred to Villani's book [17].

The Earth Mover's Distance has been studied in many geometric contexts. Cabello et al. [1] give a $(2+\varepsilon)$-approximation to minimising the EMD between two point sets under rigid transformations. For continuous distributions, rather than discrete point sets, many numerical algorithms are known (see e.g. De Goes et al. [2], Lavenant et al. [7], Mérigot [9, 10] and Solomon et al. [15]). For the case where one set contains weighted points and the other is a bounded set $C \subset \mathbb{R}^d$, Geiß et al. [4] give a geometric proof that there exists an additively weighted Voronoi diagram such that transporting mass from each point p to the part of C contained in its Voronoi cell is optimal. The weights of this Voronoi diagram can be determined numerically.

De Goes et al. [3] discuss a problem similar to our own in the context of the reconstruction and simplification of 2D shapes. Given a set of points, they want to reconstruct a simplicial complex of a given number of vertices that closely represents the shape of the point set. They start with computing the Delaunay triangulation of the point set, then iteratively collapse the edge that minimises the increase in the EMD between the point set and the triangulation. They use a variant of the EMD in which the cost is proportional to the square of the distance (2-Wasserstein distance). This allows them to calculate this variant of the EMD between a given set of points and a given edge of the triangulation exactly, as the squared distance can be decomposed into a normal and a tangential component. However, they determine the assignment of points to edges heuristically. In this work, we show how to obtain a $(1+\varepsilon)$-approximation to the true optimal solution.

3 Approximating the Earth Mover's Distance

We are given a set of points $P = \{p_1, \ldots, p_n\}$ with weights $\{w_1, \ldots, w_n\}$ and a set of segments $S = \{s_1, \ldots, s_n\}$, with lengths $\{l_1, \ldots, l_n\}$. We assume the mass associated with a segment is equal to its length, and that this mass is distributed uniformly over each segment. Given that $\sum w_i = \sum l_j = n$, we want to compute a "transport plan" of mass from P to S that minimises the cost according to the Earth Mover's Distance. We define for each pair $(p_i, s_j) \in P \times S$ a function $\mu_{i,j}(t)$, with $t \in [0, l_j]$, that describes the density of mass being moved from p_i to the point $s_j(t)$. All these functions together describe the function μ used in the definition of $\mathcal{D}(A, B)$. Such a set of functions needs to satisfy the following conditions to be a valid transport plan:

$$0 \leq \mu_{i,j}(t) \leq 1 \tag{3}$$

$$\forall i : \sum_{j=1}^{n} \int_t \mu_{i,j}(t)\, dt = w_i \tag{4}$$

$$\forall j, t : \sum_{i=1}^{n} \mu_{i,j}(t) = 1 \tag{5}$$

We can then define the cost of a given transport plan as

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \int_0^{l_j} \mu_{i,j}(t) \cdot d(p_i, s_j(t))\, dt \tag{6}$$

where $s_j(t)$ is the point on s_j associated with value t and $d(\cdot, \cdot)$ is any metric. Our problem is now to find a transport plan with minimal cost.

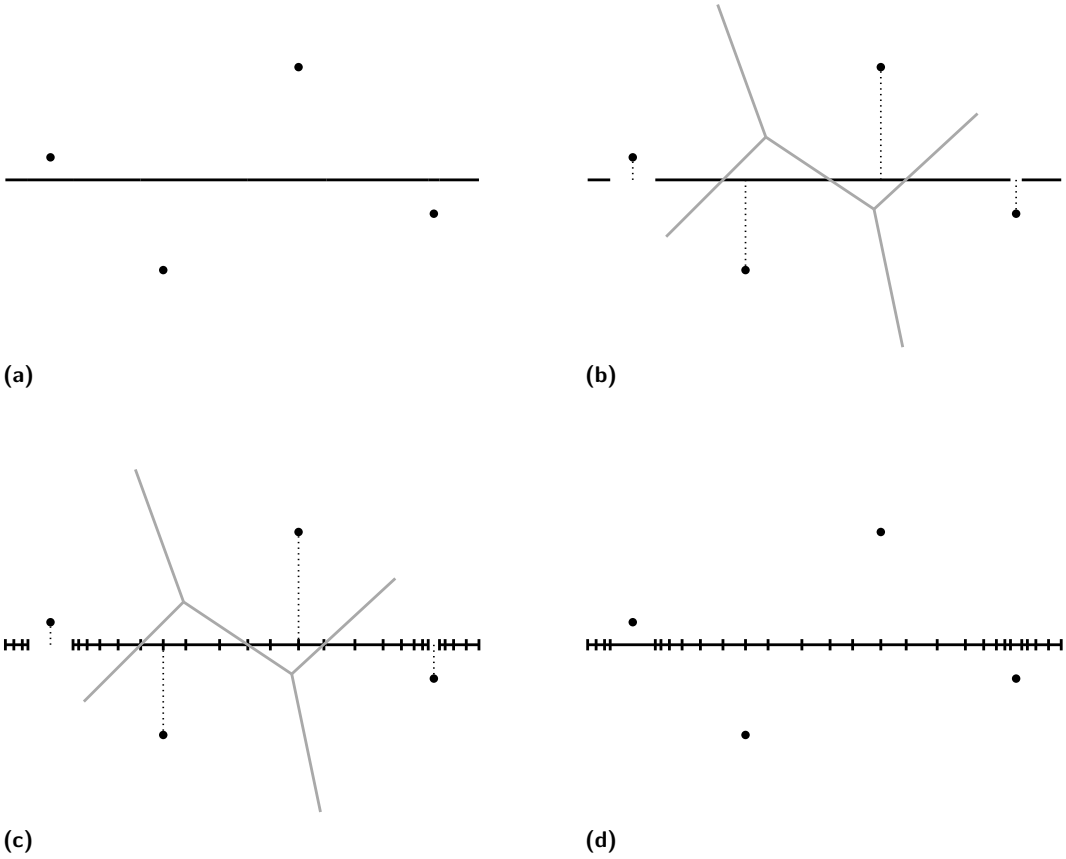


Figure 1 Our construction of Q for a single line segment under the Euclidean metric. (a) shows the input. (b) shows the Voronoi diagram of the points and the parts of the segment with distance at least δ/n. (c) shows the generated subsegments, and (d) all of Q, with the parts with distance less than δ/n added back in.

We now describe a polynomial-time algorithm that finds a transport plan with a cost that is at most $1+\varepsilon$ times the cost of the optimal transport plan. Our strategy is as follows: we subdivide each segment such that for each subsegment s' the ratio of the distance to the closest and furthest point on s' for any $p_i \in P$ is at most $1+\delta$. We then solve a min cost max flow problem on a bipartite graph between P and the subdivided segments, where the cost of any edge is equal to the shortest distance between a point and a subsegment. Finally, we use the solution to this flow problem to build a discrete transport plan. For an appropriate choice of δ, this gives a $(1+\varepsilon)$-approximation.

We begin by subdividing our segments as follows. First, we remove all parts of segments that lie within distance δ/n of any point in P (we will consider these parts separately later). Call the remaining set of segments S'; we subdivide S' by performing the following procedure: for each point in $p \in P$, we consider the part of S' that is within its Voronoi cell. Call this part S'_p. Now consider the closest point to p of any segment $s \in S'_p$, call their distance d. We create a subsegment s', starting at the closest point of s to p, with length $d \cdot (1+\delta)$ (or the length of s, if that is smaller). We remove this subsegment s' from S'_p, and iterate until S' is empty. Note that this way, the subsegments increase in size in both directions from the closest point to p. Call the set of all s' and all parts of the segments that lie within distance δ/n Q; this is our subdivision of S.

▶ **Lemma 1.** *Q has $O\left(\frac{n^2}{\delta}\log\frac{1}{\delta}\right)$ subsegments.*

Proof. Consider any segment $s_j \in S$. As the subsegments are created based on the closest point, we define two variables r_i and ℓ_i that denote the length of the part of s_j contained in p_i's Voronoi cell on either side of the perpendicular line from p_i to the supporting line of s_j. We also have at most one subsegment per point for the part that is within distance δ/n. The number of subsegments generated by p_i on s_j can then be expressed as $g(r_i,\ell_i) \leq \lceil \log_{1+\delta}(\frac{r_i}{\delta/n})\rceil + \lceil \log_{1+\delta}(\frac{\ell_i}{\delta/n})\rceil + 1 \leq \log_{1+\delta}(\frac{r_i}{\delta/n}) + \log_{1+\delta}(\frac{\ell_i}{\delta/n}) + 3$. Here we are counting the number of times we need to multiply the starting distance of δ/n by $1+\delta$ in order to reach length r_i or ℓ_i.

We are interested in the worst-case number of subsegments, so we want to find the maximum value of $\sum g(r_i,\ell_i)$ subject to the constraint that $\sum(r_i+\ell_i) \leq l_j$. As $\sum g(r_i,\ell_i)$ is a sum of logarithms, this is the same as maximising the product of all $g(r_i,\ell_i)$, which is achieved when all r_i and ℓ_i are equal (i.e. all are $l_j/2n$). By the same argument, the worst number of subsegments generated on all of S is upper bounded by making all l_j equal (i.e. all are 1). This gives us an upper bound on the total number of subsegments of

$$\sum_{i=1}^{n}\sum_{j=1}^{n} 2\cdot\log_{1+\delta}\left(\frac{1/2n}{\delta/n}\right) + 6 = 2n^2\cdot\frac{\ln\left(\frac{1}{2\delta}\right)}{\ln(1+\delta)} + 6n^2 = O\left(\frac{n^2}{\delta}\ \log\ \frac{1}{\delta}\right) \tag{7}$$

Note that we use the fact that $\ln(1+\delta) \approx \delta$ for small values of δ. We get that our number of subsegments is $O\left(\frac{n^2}{\delta}\log\frac{1}{\delta}\right)$ in the worst case. ◀

We now define a complete bipartite graph $G = (P \cup Q, E)$, with edges between each point-subsegment pair. The cost of each edge will simply be the shortest distance between the point and segment it connects. A solution to a flow problem in G can be transformed into a transport plan by assigning a piece of subsegment to a point with length equal to the amount of flow along the corresponding edge. We will show that the EMD between P and S is approximated by the cost of any transport plan derived from a min cost max flow in G.

First note the following general lower bound on the cost of an optimal solution:

▶ **Lemma 2.** *The Earth Mover's Distance between P and Q is bounded from below by the cost $\|\mathcal{W}\|$ of a min cost max flow $\mathcal{W}$ in G.*

Proof. Consider any transport plan that minimises the Earth Mover's Distance; call the cost associated with this plan OPT. If instead of spreading the mass equally over the whole segment, we move all the mass to the closest point on the segment, we obtain a plan with cost $\text{OPT}^* \leq \text{OPT}$. Such a plan is a solution to a maximum flow problem in G, as it moved all available mass. It follows that the cost $\|\mathcal{W}\|$ of a minimum cost maximum flow $\mathcal{W}$ in G satisfies $\|\mathcal{W}\| \leq \text{OPT}$. ◀

We also note the following lower bound on the value of $\|\mathcal{W}\|$:

▶ **Lemma 3.** $\|\mathcal{W}\| \geq \delta - 2\delta^2$.

Proof. For each point-segment pair, the part of the segment that has distance at most δ/n has length at most $2\delta/n$. The total length over all point-segment pairs is then $2\delta n$. This leaves $n - 2\delta n$ length with distance of at least δ/n, which gives a minimum cost of $(n-2\delta n)\cdot\delta/n = \delta - 2\delta^2$. Due to our construction of Q, we know that no subsegment crosses the distance boundary of δ/n. It follows that $\delta - 2\delta^2 \leq \|\mathcal{W}\|$. ◀

Using the lower bound from Lemma 2 and the way we constructed Q, we can derive a lower and upper bound on the solution obtained by the flow problem.

▶ **Lemma 4.** *For any transport plan $\mathcal{T}$ derived from $\mathcal{W}$ we have that its cost $\|\mathcal{T}\| \leq (1+\delta)\|\mathcal{W}\| + 2\delta^2$.*

Proof. We can upper bound $\|\mathcal{T}\|$ by measuring all distances to the furthest point in each subsegment. We constructed Q such that the ratio of the closest and furthest distance between any point-subsegment pair was $1+\delta$ when the closest distance was at least δ/n. We can therefore bound all parts of $\mathcal{T}$ where the distance is at least δ/n by $(1+\delta)\|\mathcal{W}\|$. The total mass being moved over a distance at most δ/n in $\mathcal{T}$ is at most δn, giving a cost of $2\delta^2$. The total cost when measuring to the furthest point is therefore $(1+\delta)\|\mathcal{W}\| + 2\delta^2$. ◀

▶ **Corollary 5.** $\|\mathcal{W}\| \leq \text{OPT} \leq (1+\delta)\|\mathcal{W}\| + 2\delta^2$.

Putting this all together, we can show that $\|\mathcal{T}\|$ approximates OPT.

▶ **Theorem 6.** *The cost of any transport plan $\mathcal{T}$ derived from $\mathcal{W}$ is a $(1+5\delta)$-approximation to the Earth Mover's Distance between P and S for $0 < \delta \leq \frac{1}{4}$.*

Proof. The ratio between the upper and lower bound on $\|\mathcal{T}\|$ is

$$\frac{(1+\delta)\|\mathcal{W}\| + 2\delta^2}{\|\mathcal{W}\|}$$

This ratio is the largest for small values of $\|\mathcal{W}\|$, so we plug in the lower bound from Lemma 3:

$$\begin{aligned}
& \frac{(1+\delta)\|\mathcal{W}\| + 2\delta^2}{\|\mathcal{W}\|} \\
\leq\ & \frac{(1+\delta)(\delta - 2\delta^2) + 2\delta^2}{\delta - 2\delta^2} \\
=\ & \frac{1+\delta-2\delta^2}{1-2\delta} \\
=\ & 1 + \frac{3\delta - 2\delta^2}{1-2\delta} \\
=\ & 1 + \delta + \frac{2\delta}{1-2\delta} \\
\leq\ & 1 + 5\delta \qquad \text{(assuming } \delta \leq \tfrac{1}{4}\text{)}
\end{aligned}$$

As $\|\mathcal{W}\|$ is also a lower bound for OPT, and $\mathcal{T}$ can obviously not have lower cost than the optimal transport plan, this gives a $(1+5\delta)$-approximation. ◀

Setting $\delta = \varepsilon/5$ gives a $(1+\varepsilon)$-approximation.

3.1 Analysis

We can calculate $\mathcal{W}$ in $O(|E|\log|V|((|E|+|V|)\log|V|))$ time using Orlin's algorithm for minimum cost maximum flows [12]. In our case, $|V| = O\left(\frac{n^2}{\varepsilon}\log\frac{1}{\varepsilon}\right)$ and $|E| = O\left(\frac{n^3}{\varepsilon}\log\frac{1}{\varepsilon}\right)$; as $|V| \in O(|E|)$, we can simplify the running time to $O(|E|^2\log^2|V|)$. This gives us a total running time of $O\left(\frac{n^6}{\varepsilon^2}\log^2\left(\frac{1}{\varepsilon}\right)\log^2\left(\frac{n^2}{\varepsilon}\log\frac{1}{\varepsilon}\right)\right)$.

This time can be improved when the lengths of the segments in S are divisible by δ^2/n, by making all subsegments have the same length. When this is the case, $\mathcal{W}$ becomes a minimum cost matching rather than a minimum cost maximum flow. We can then use the algorithm by Sharathkumar and Agarwal to find a $(1+\delta)$-approximate bipartite matching in $O(|V| \text{ poly}(\log |V|, \frac{1}{\delta}))$ time [14]. We pay for this approximate rather than optimal matching by an extra 2δ in our approximation factor, giving a $(1+7\delta)$-approximation. Using this subdivision, the number of subsegments is n^2/δ^2, giving a total running time of $O\left(\frac{n^2}{\varepsilon^2} \text{ poly}\left(\log \frac{n^2}{\varepsilon^2}, \frac{1}{\varepsilon}\right)\right) = O\left(n^2 \text{ poly}\left(\log \frac{n}{\varepsilon}, \frac{1}{\varepsilon}\right)\right)$.

References

1 S. Cabello, P. Giannopoulos, C. Knauer, and G. Rote. Matching point sets with respect to the Earth Mover's Distance. *Computational Geometry*, 39(2):118–133, 2008.

2 F. de Goes, K. Breeden, V. Ostromoukhov, and M. Desbrun. Blue noise through optimal transport. *ACM Transactions on Graphics*, 31(6):171, 2012.

3 F. de Goes, D. Cohen-Steiner, P. Alliez, and M. Desbrun. An optimal transport approach to robust reconstruction and simplification of 2D shapes. *Computer Graphics Forum*, 30(5):1593–1602, 2011.

4 D. Geiß, R. Klein, R. Penninger, and G. Rote. Optimally solving a transportation problem using voronoi diagrams. *Computational Geometry*, 46(8):1009 – 1016, 2013.

5 K. Grauman and T. Darrell. Fast contour matching using approximate Earth Mover's Distance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 220–227, 2004.

6 L. V. Kantorovich. On the translocation of masses. *Doklady Akademii Nauk*, 37:199–201, 1942.

7 H. Lavenant, S. Claici, E. Chien, and J. Solomon. Dynamical optimal transport on discrete surfaces. In *SIGGRAPH Asia 2018 Technical Papers*, pages 250:1–250:16, 2018.

8 F. Mémoli. Spectral Gromov-Wasserstein distances for shape matching. In *Proceedings of the 12th IEEE International Conference on Computer Vision Workshops*, pages 256–263, 2009.

9 Q. Mérigot. A multiscale approach to optimal transport. *Computer Graphics Forum*, 30(5):1583–1592, 2011.

10 Q. Mérigot, J. Meyron, and B. Thibert. An algorithm for optimal transport between a simplex soup and a point cloud. *SIAM Journal on Imaging Sciences*, 11(2):1363–1389, 2018.

11 G. Monge. Mémoire sur la théorie des déblais et des remblais. *Histoire de l'Académie Royale des Sciences de Paris*, 1781.

12 J. B. Orlin. A faster strongly polynomial minimum cost flow algorithm. *Operations Research*, 41(2):338–350, 1993.

13 Y. Rubner, C. Tomasi, and L. J. Guibas. The Earth Mover's Distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

14 R. Sharathkumar and P. K. Agarwal. A near-linear time ε-approximation algorithm for geometric bipartite matching. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing*, pages 385–394, 2012.

15 J. Solomon, F. de Goes, G. Peyré, M. Cuturi, A. Butscher, A. Nguyen, T. Du, and L. Guibas. Convolutional Wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Transactions on Graphics*, 34(4):66, 2015.

16 Z. Su, Y. Wang, R. Shi, W. Zeng, J. Sun, F. Luo, and X. Gu. Optimal mass transport for shape matching and comparison. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(11):2246–2259, 2015.

17 C. Villani. *Optimal Transport: Old and New*. Springer Verlag, Heidelberg, 2008.

Balanced Covering Problem in Bicolored point Sets

Sujoy Bhore[1], Supantha Pandit[2], and Sasanka Roy[3]

1 **Algorithms and Complexity Group, TU Wien, Vienna, Austria**
`sujoy.bhore@gmail.com`
2 **Stony Brook University, Stony Brook, NY, USA**
`pantha.pandit@gmail.com`
3 **Indian Statistical Institute, Kolkata, India**
`sasanka.ro@gmail.com`

Abstract

We study a variation of the classical set cover problem called the *balanced covering (BC)* problem on a set of red and blue points in the Euclidean plane. Let P be a set of red and blue points in the plane. An object is called a *balanced* object with respect to P, if it contains an equal number of red and blue points from P. In the BC problem, the objective is to cover the points in P with a minimum number of homogeneous geometric objects (i.e., unit squares, intervals) such that each object is *balanced*. For points in the plane, we prove that the BC problem is NP-hard when the covering objects are unit squares. For points on a line, we show that if the ratio of the total numbers of reds and blues is more than 2 then, there exists no solution of the BC problem. Subsequently, we devise a linear time exact algorithm for the BC problem with intervals. Finally, we study the study the problem of computing a balanced object of maximum cardinality. For this, we give polynomial time algorithms with unit squares in the plane and intervals on a line.

1 Introduction

Set cover is a well-studied problem in computer science with numerous application in various fields. In this problem a set P of points and a set O of objects are given and the objective is to cover all the points in P with a minimum number of objects in O. We consider a variation of this problem on a bicolored (red and blue) point sets. Let $P = R \cup B$ be a set of bicolored points in the plane, where R denotes a set of red and B denotes a set of blue points. We say that a geometric object X is *balanced* if it covers an equal number of red and blue points. Here we consider two problems based on the coverage of the points in P.

Balanced Covering (BC) Problem
Given a set $P = R \cup B$ of bicolored points in the plane, the objective is to find a minimum collection set of balanced objects that covers P.

Maximum Balanced Object (MBO) Problem
Given a set $P = R \cup B$ of bicolored points in the plane, the objective is to find a balanced object that covers the maximum number of points in P.

A related problem to the BC problem is the class cover problem: given a set of red and a set of blue points and a set of objects, the goal is to cover all the blue points excluding the red points with minimum number of objects [2, 1]. Another related problem is the red-blue set cover problem [3]. Generalized versions of these problems are studied in [7]. Chan and Hu [4] considered this problem when the covering objects are unit squares, and proved the

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

NP-hardness and gave a PTAS. We would like to mention that, in the discrete setting, the BC problem is equivalent to the standard geometric set cover problem and therefore becomes NP-hard [6]. However, in the continuous setting, one would require to decide the particular placement of the objects due to the color constraint.

Our Results: In Section 2, for points in the plane, we prove that the BC problem is NP-hard when the covering objects are unit squares. In Section 3, for points on a line, we show that if the ratio of the total numbers of reds and blues is more than 2 then, there does not exist a solution of the BC problem. Subsequently, we devise a linear time exact algorithm for the BC problem with intervals. Finally, we study the problem of computing a single balanced object of maximum cardinality. For this, we give polynomial time algorithms with unit squares (in section 3) in the plane and intervals on a line.

2 NP-Hardness: *BC* problem with Unit Squares

We prove that the BC problem with unit squares is NP-hard by a reduction from the rectilinear planar monotone 3SAT (shortly RPM3SAT) problem that is known to be NP-complete [5]. We define this problem as follows. A clause is said to be a positive (resp. negative) clause if all the literals it contains are positive (resp. negative). We are given a 3SAT instance ϕ with n variables and m clauses either positive or negative. The variables are positioned on a horizontal line. The positive clauses are above this line, and they connect to its corresponding variables with three legs: (i) ⌜, (ii) ׀, and (iii) ⌝. The negative clauses are below the line, and they connect to its corresponding variables with three legs: (iv) ⌞, (v) ׀, and (vi) ⌟. Finally, these legs do not intersect each other. The objective is to find a satisfying assignment for ϕ. See Figure 1 for an instance of the RPM3SAT problem.

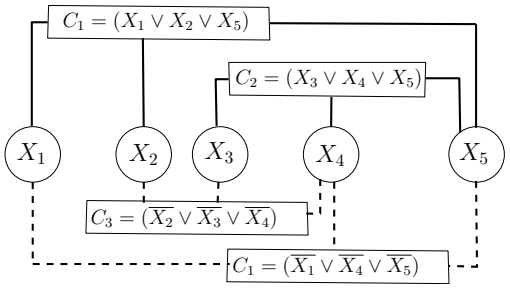


Figure 1 An instance of the RPM3SAT problem.

We construct an instance I_ϕ of the BC problem from an instance ϕ of the RPM3SAT problem. Let $\{u, v\}$ be a pair of two differently colored points that are ϵ $(0 < \epsilon \ll 1)$ distance apart. We call such a pair a *site*. We assume that the given points are a collection of sites.

Variable Gadget: The gadget for the variable x_i has two parts: a cycle and a set of chains that are attached to the cycle. Let d be the maximum number of clauses in which x_i is present, i.e, the number of legs attached to x_i. Then, x_i contains d chains one for each leg. Consider two imaginary axis-parallel horizontal lines ℓ_1 and ℓ_2 that are suitably placed such that any two sites placed on ℓ_1 and ℓ_2 cannot be covered by a unit square (see Figure 2). We place $6m + 2$ sites $s_1, s_2, \ldots, s_{6m+2}$ on ℓ_1 such that the distance between any two consecutive sites is exactly 1. In a similar fashion, we place $6m + 2$ sites $t_1, t_2, \ldots, t_{6m+2}$ sites on ℓ_2 as well (see Figure 2). We use 6 additional sites and place them in the following way; 3 sites each to the left and right of the above arrangement (see Figure 2). Thus as a total of $12m + 10$ sites form a cycle like structure. Notice that, due to this construction,

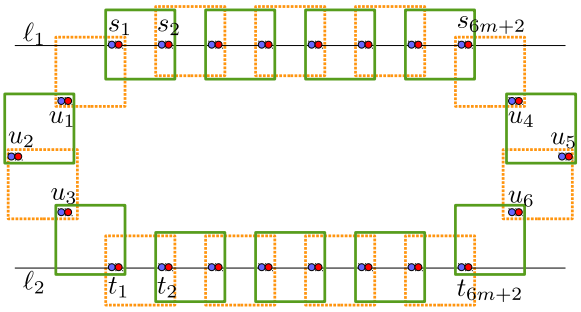


Figure 2 The cycle gadget corresponding to a variable x_i.

in the continuous balanced covering any unit square can cover at most two sites that are consecutive along the cycle. In Figure 2, we demonstrate a set of possible canonical unit squares that cover the sites. Note that in an optimal balanced covering exactly half of the squares are selected: either all solid or all dotted. There are three types of *chains* that are attached to the cycle of a variable gadget. Each chain is a specific geometric embedding of a set of sites. The gadgets of types (i) and (ii) chains are shown in Figure 3(a) and Figure 3(b) respectively. The other types ((iii)-(vi)) of chains are constructed by a simple modification of types (i) and (ii) chains.

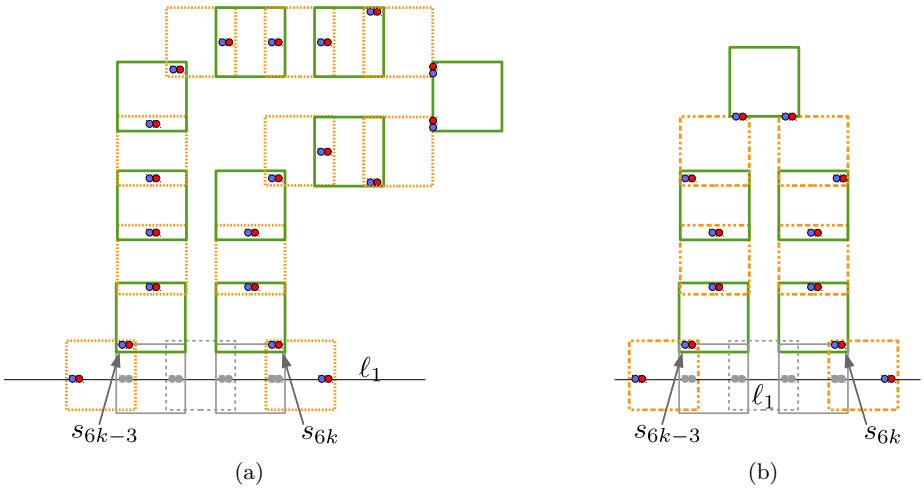


Figure 3 The chains of a variable (a)Type 1 (b)Type 2.

It needs to be mentioned that the number of sites is not fixed for every chain, even for similar chains of different clauses. Now we explain how the chains are attached to the variable cycle. Let C_1, C_2, ... be the order of the positive clauses that connect to a variable. We associate the four site s_{6k-3}, s_{6k-2}, s_{6k-1}, and s_{6k} in the cycle with the k-th clause in this order. We remove the two sites s_{6k-2} and s_{6k-1} from the cycle and perturb the other two sites s_{6k-3} and s_{6k} slight vertically up. See Figure 3 for detailed explanation.

▶ Observation 1. Let S_{x_i} be a set of unit squares associated with x_i. Exactly $\delta_i = |S_{x_i}|/2$ squares (either all solid or all dotted) are required for an optimal balanced-covering of x_i.

Clause: We describe a clause gadget and how it interacts with variable gadgets. Assume, w.l.o.g., that $C_i = (x_i, y_i, z_i)$ is a positive clause. For C_i, we take a special site w_i called the

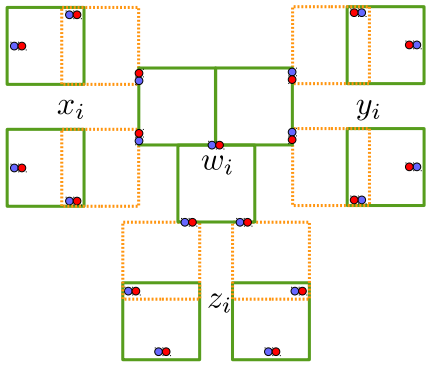


Figure 4 Positive clause interaction with the three variables it contains.

clause-site that connects the chains corresponding to the variables. The placement of w_i with respect to the three chains is shown in Figure 4. Notice that, no two sites from two different chains are covered by a single square. Clearly, I_ϕ can be constructed in polynomial time.

▶ **Lemma 2.1.** *ϕ is satisfiable if and only if I_ϕ has a balance cover with $\delta = \sum_{i=1}^{n} \delta_i$ squares.*

Thereby, we conclude the following theorem.

▶ **Theorem 2.2.** *The BC problem is* NP*-hard.*

3 Points on a Line

Let P be a set of of m red and n blue points on a line. We show that there is no solution for the BC problem when $\frac{m}{n} > 2$. Note, however it is not guaranteed that there is always a solution for BC problem if $\frac{m}{n} \leq 2$.

▶ **Theorem 3.1.** *Given a set P of m red points and n blue points on a real line $\mathcal{L}$, if $\frac{m}{n} > 2$ then, there does not exist a solution for the BC problem.*

Proof. For the sake of contradiction, let us assume that there is an optimal solution, $OPT = \{I_1, \ldots, I_j\}$ of the BC problem that covers P.

▶ Claim 1. Every blue point $b_i \in P$ is contained in at most two intervals in OPT.

Proof. For the sake of contradiction, let b_i be contained in at least three intervals (say I_i, I_j, I_k) in OPT. We select the two intervals from I_i, I_j, I_k; one whose left end point is left most and the other whose right end point is right most. Clearly, removing I_i, I_j, I_k from OPT and adding these two intervals in OPT still covers all the points, a contradiction. ◀

For each interval $I_i \in OPT$, we define a red-blue pairing in the following manner. Let $S(I_i) \subseteq P$ be the subset of points contained in I_i. Let $\{r_1, \ldots, r_p\}$ and $\{b_1, \ldots, b_p\}$ be the red and blue points in sorted order (x-coordinate wise) in I_i. Now, we consider the red-blue pairs $\{\{r_1, b_1\}, \ldots \{r_p, b_p\}\}$. For an interval I_k, consider a pair $\{r_i, b_i\}$ (for some i), we say b_i *balances* r_i. Now any blue point b_i that is contained in an interval I_k can *balance* exactly one red point in I_k. Using Claim 1, we conclude that each blue point b_i can *balance* at most two different red points. Hence, the ratio between reds and blues is at most 2. ◀

3.1 Exact Algorithms for Intervals

We give exact algorithms for *BC* and *MBO* problems while the covering objects are intervals. We denote them as *BCI* and *MBI* problem, respectively. Let $P = \{p_1, p_2, \ldots, p_n\}$ be a set of red and blue points on a line given in sorted order. The idea is to obtains a set of *candidate* intervals, and then choose intervals from this set with some modification. Observe that, in any optimal solution, a chosen balanced interval is not contained in any other chosen interval.

Finding candidate intervals: We maintain a counter c (initially it is empty) and an array $A[n]$. For each point $p_i \in P$ in the order, if p_i is red we increase the value of c by 1, otherwise decrease the value by 1, and set $A[i] = c$. Note that the values in A are integers and in the range $[-n, n]$. Let l and h be the minimum and maximum values of the counter c, respectively. Note that l and h can be negative. We construct a table T of $(|l| + |h| + 1)$ rows and 3 columns as follows. The first column stores the values in the range $[l, h]$ in order. The values of the second and third columns are initially empty. We update some of the entries during the following procedure. We go through each entry of A one by one. For the ith entry i.e., $A[i]$, if $T[A[i]][2]$ is empty then $T[A[i]][2] = i$. Else $T[A[i]][3] = i$. Now for each row i we generate an interval if $T[i][3]$ is non-empty. Therefore, we generates at most $n/2$ intervals based on the indices of A between the l and h.

▶ Observation 2. For each candidate interval the difference in reds and blues is at most 1.

Now we make each candidate interval (say I_c) that is balanced in the following way. Let (i, j) be the index of I_c. We consider the 9 intervals $\{i-1, i, i+1\} \times \{j-1, j, j+1\}$ and choose the maximum balance one. We make it for all candidate intervals. This process gives us a set of candidate balance intervals. Now to find the maximum balance interval just return the candidate balance interval that contains maximum number of points. For covering the whole point set P, we run the standard greedy algorithm on the candidate balance intervals and return the minimum cardinality subset of intervals that covers P.

▶ **Theorem 3.2.** *Given a set P of red and blue points in sorted order on the line,*

1. *finding the largest balanced interval requires $O(n)$ time and*
2. *finding minimum number of balanced intervals that cover P also requires $O(n)$ time.*

4 Exact Algorithm for Unit Squares

Let $P = \{p_1, p_2, \ldots, p_n\}$ be a set of red and blue points on a line given in sorted order. We study the *MBO* problem with unit square (shortly, *MBS*). Consider a unit square s, it is called a 2-anchored square if there is at least two distinct points on any of its two consecutive sides. Notice that, the output of the *MBS* problem is a 2-anchored square, otherwise we can always translate it to make such one without loosing any point (see Figure 5(a)). The naive algorithm for the *MBS* problem is following. First, we build a range tree T of P. Now, for each 2-anchored square we check in T, the points that is contained in that square, and report the maximum balanced square. This process takes $O(n^2 \log n)$ time. We give a simple $O(n^2)$ time algorithm for the *MBS* problem.

For each point $p_i \in P$, let $S(p_i)$ be the square centered at p_i. If p_i is a blue point (resp. red point) then, $S(p_i)$ is a blue square (resp. red square). Let $\mathcal{S}$ be the set of red and blue squares based on the points of P. Let E be the set that contains the left and right endpoints of the squares, and $|E| = 2n$. We consider the points in E in sorted order from left-to-right based on their x-coordinates.

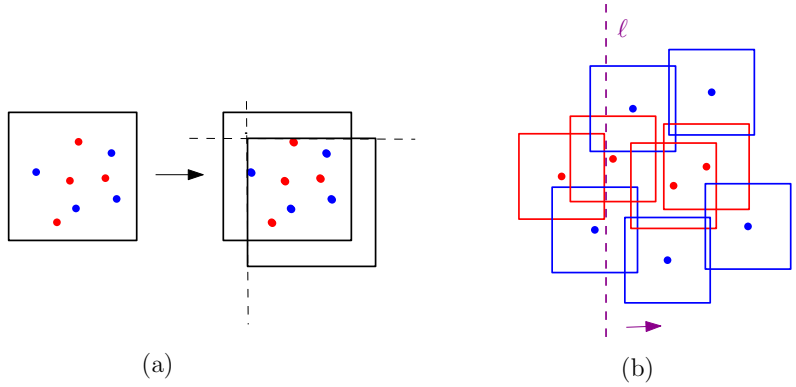


Figure 5 (a) Translation into a 2-anchored square. (b) A vertical line intersecting a set of bicolored squares.

▶ Observation 3. Consider any two points $p_i, p_j \in P$ such that $S(p_i)$ and $S(p_j)$ intersect. For an arbitrary point p_k that is in the intersecting region of $S(p_i)$ and $S(p_j)$, the square $S(p_k)$ contains both p_i and p_j.

We use the following procedure to find a maximum balanced square. Let ℓ be a vertical sweep line that considers the squares from left-to-right. Whenever, ℓ reaches to a point in E, we call it an event point and indeed we have $2n$ event points. For each $i \in [2n]$, the line ℓ passes through a point $p_i \in E$, and let $\mathcal{S}' \subseteq \mathcal{S}$ be a subset of squares intersecting ℓ (see Figure 5(b)). Beside that we maintain two counters (r, b) for each square. Whenever a square s enters or leaves ℓ we update in the range tree the (r, b) values of all the nodes whose corresponding squares intersecting s. Note that, at each event point, we basically have a set of red and blue unit intervals on ℓ. In linear time, we can compute the largest balanced subset clique. We know that, their corresponding squares have a common intersecting region. It is possible to place p_k in that region such that $S(p_k)$ is a balanced square (from observation 3). This process takes $O(n^2)$ time.

▶ **Theorem 4.1.** *Let P be a set of red and blue points on a line given in sorted order, there is an algorithm that computes maximum balanced unit square in $O(n^2)$ time.*

References

1 Sergey Bereg, Sergio Cabello, José Miguel Díaz-Báñez, Pablo Pérez-Lantero, Carlos Seara, and Inmaculada Ventura. The class cover problem with boxes. *Computational Geometry*, 45(7):294 – 304, 2012.

2 Adam Cannon and Lenore Cowen. Approximation algorithms for the class cover problem. *Ann. Math. Artif. Intell.*, 40(3-4):215–224, 2004.

3 Robert D. Carr, Srinivas Doddi, Goran Konjevod, and Madhav Marathe. On the red-blue set cover problem. In *SODA*, pages 345–353, 2000.

4 Timothy M. Chan and Nan Hu. Geometric red–blue set cover for unit squares and related problems. *Computational Geometry*, 48(5):380 – 385, 2015.

5 Mark de Berg and Amirali Khosravi. Optimal binary space partitions for segments in the plane. *Int. J. Comput. Geometry Appl.*, 22(3):187–206, 2012.

6 Robert J Fowler, Michael S Paterson, and Steven L Tanimoto. Optimal packing and covering in the plane are NP-complete. *Information processing letters*, 12(3):133–137, 1981.

7 Apurva Mudgal and Supantha Pandit. Generalized class cover problem with axis-parallel strips. In *Workshop on Algorithms and Computation, WALCOM*, pages 8–21, 2014.

Testing Transmission Graphs for Acyclicity*

Haim Kaplan[1], Katharina Klost[2], Wolfgang Mulzer[2], Liam Roditty[3], and Micha Sharir[1]

1 **Tel Aviv University, Israel**
{haimk, michas}@post.tau.ac.il
2 **Institut für Informatik, Freie Universität Berlin, Germany**
{kathklost,mulzer}@inf.fu-berlin.de
3 **Bar Ilan University, Israel**
liamr@macs.biu.ac.il

Abstract

Let S be a set of n *point sites* in the plane, such that each site $s \in S$ has an *associated radius* $r_s > 0$. The *transmission graph* on S, denoted $T(S)$, is the directed graph with vertex set S where st is a directed edge if and only if $|st| \leq r_s$, i.e., if t lies in the disk D_s with center s and radius r_s. A basic question is to decide whether $T(S)$ is *acyclic*, i.e., whether $T(S)$ does not contain a directed cycle. We show that if our notion of directed cycle also includes cycles with two edges, then this problem can be solved in $O(n \log n)$ expected time, independent of the number of edges in $T(S)$.

Along the way, we encounter a batched range searching problem that may be interesting in its own right: given $O(n)$ *query triples* of the form (p, r_1, r_2), with $p \in \mathbb{R}^2$ and $0 < r_1 < r_2$, report for every query (p, r_1, r_2) one site $s \in S$ with $p \in D_s$ and $r_s \in [r_1, r_2)$, if it exists. We show how to solve this range searching problem in $O(n \log n)$ expected time.

1 Introduction

Transmission graphs are a popular model for directed sensor networks with different transmission radii (see, e.g., [8] and the references therein). We are given a set S of n *point sites* in the plane, representing the locations of the sensors. Each site $s \in S$ has an *associated radius* $r_s > 0$ that models the transmission strength of the corresponding sensor. The *disk for the site* s, denoted D_s, is the disk with center s and radius r_s. The *directed transmission graph* $T(S)$ has vertex set S and a directed edge from a site s to a site t if and only if $t \in D_s$, i.e., if the sensor s can reach the sensor t. Throughout, we will assume that S is in *general position*, which means that no site lies on the disk boundary of any other site and that all associated radii are pairwise distinct. Even though transmission graphs may contain $\Omega(n^2)$ edges, it turns out that many problems on them can be solved without explicitly generating all those edges [7, 8].

Here, we consider the basic problem of testing whether $T(S)$ contains a *directed cycle*. This is a sequence $s_1, \ldots, s_k$ of $k \geq 2$ sites such that $s_1 \in D_{s_k}$ and such that $s_{i+1} \in D_{s_i}$, for $i = 1, \ldots, k-1$. There are two reasonable variants of this question, depending on whether we allow the case $k = 2$ or insist on $k \geq 3$. We show that if $k = 2$ is permitted, this problem can be solved in $O(n \log n)$ expected time, independent of the number of edges.

Our algorithm is based on the observation that if $T(S)$ contains a directed cycle, then it must contain a directed cycle with two edges. Furthermore, we identify a batched range query problem that may be interesting by itself. Given n disks in the plane, we want to efficiently answer a batch of n queries, each consisting of a point and a radius range. For

* Partially supported by ERC STG 757609 and GIF grant 1367/2015.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

each such query, we want to find a disk whose radius lies in the given range and that contains the given query point. Kaplan et al. [7] considered a specialized version of this range query, using a nearest neighbor data structure distributed on a balanced binary search tree. Their solution achieves worst-case running time $O(n \log^2 n)$. Another related query structure is due to Imai et al. [6]. Their structure stores a set of n disks. The preprocessing time is $O(n \log n)$, and it can find for any point $p \in \mathbb{R}^2$ in $O(\log n)$ time a disk that contains p, if it exists. Our structure uses a similar high-level approach as the structure of Kaplan et al. [7]. To achieve a better running time, we lift the problem to $\mathbb{R}^3$ and draw on previous results on convex hulls and the intersection of convex polyhedra.

2 Range Queries

We begin by describing our batched range searching problem. The setting is as follows: let S be a set of n sites in $\mathbb{R}^2$, each with an associated radius $r_s > 0$. Let D_s be the disk with radius r_s and center s. We will consider the following query problem:

(R) We are given $O(n)$ query triples (p, r_1, r_2), where $p \in \mathbb{R}^2$ and $r_2 \geq r_1 > 0$. For every query triple, we would like to find a site $s \in S$ with $r_s \in [r_1, r_2)$ and $p \in D_s$, if it exists.

2.1 Canonical Intervals, Paths, and Nodes

The queries in (R) concern sites whose associated radii lie in given intervals. Just as in range trees [1, Chapter 5], we subdivide each such *query interval* $[r_1, r_2)$ into $O(\log n)$ pieces from a fixed set of *canonical intervals*. For this, we build a balanced binary tree B on S. For an example of the tree, illustrating the concepts introduced in this section see Figure 1 The leaves of B are the sites of S, sorted from left to right by increasing associated radius. The tree B has n leaves, $O(n)$ vertices, and height $O(\log n)$. For each vertex $v \in B$, let $\mathcal{I}_v$ be the sorted list of sites in the leaves of the subtree rooted at v. We call the sets $\mathcal{I}_v$, for $v \in B$, the *canonical intervals* of S. There are $O(n)$ of them.

Next, we define *canonical paths* and *canonical nodes* for a query $q = (p, r_1, r_2)$. For a radius $r > 0$, the *predecessor* of r in S is the site $s \in S$ with the largest radius $r_s \leq r$. The *proper predecessor* of r is the site $s \in S$ with the largest radius $r_s < r$. The *successor* and *proper successor* of r are defined analogously. Let $[r_1, r_2)$ be the query interval of q. We consider the path π_1 in B from the root to the leaf with the proper predecessor t_1 of r_1 and the path π_2 in B from the root to the leaf with the successor t_2 of r_2. If t_1 does not exist, we take π_1 as the left spine of B, and if t_2 does not exist, we take π_2 as the right spine of B. Then, π_1 and π_2 are called the *canonical paths* for q. The set of *canonical nodes* for q is defined as follows: for each vertex v in $\pi_1 \setminus \pi_2$, we include the right child of v if it is not in π_1, and for each v in $\pi_2 \setminus \pi_1$, we include the left child of v if it is not in π_2. Furthermore, we include the last node of π_1 if t_1 does not exist, and the last node of π_2 if t_2 does not exist.

▶ **Lemma 2.1.** *The total size of the canonical intervals is $O(n \log n)$, and the tree B together with the sorted canonical intervals can be built in $O(n \log n)$ time. For any query q, there are $O(\log n)$ canonical nodes, and they can be found in $O(\log n)$ time. The canonical intervals for the canonical nodes of q constitute a partition of the query interval for q.*

Proof. Since a site $s \in S$ appears in $O(\log n)$ canonical intervals, the total size of the canonical intervals is $O(n \log n)$. To construct B, we sort S according to the associated radii r_s, and we build B on top of the sorted list. To find the sorted canonical intervals, we perform a bottom-up traversal of B, obtaining the canonical interval for each internal node by copying and joining the canonical intervals of its children.

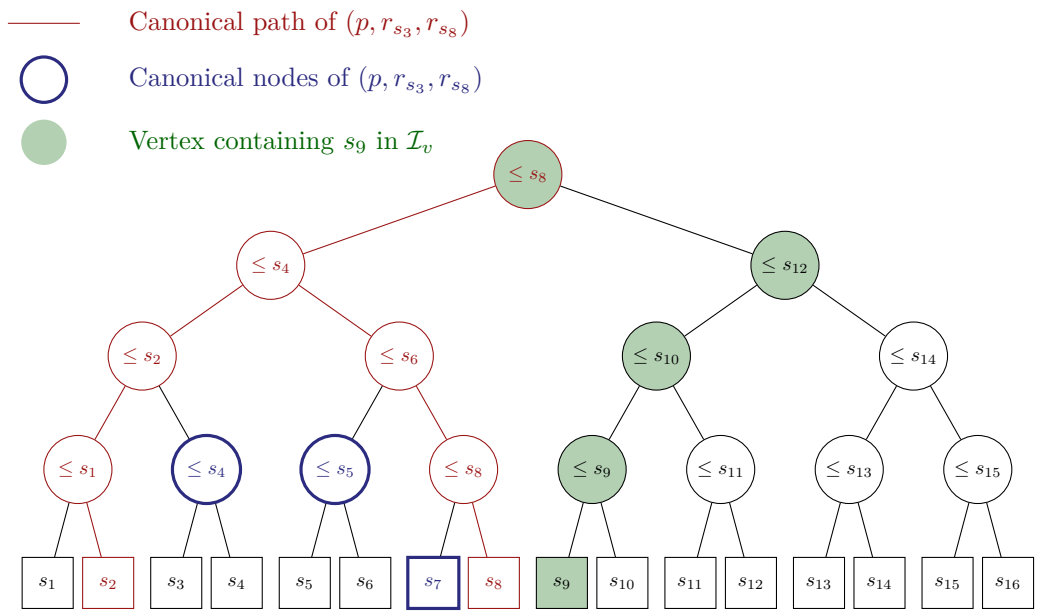


Figure 1 An example of canonical intervals, paths, and nodes.

The bound on the number of canonical nodes for q follows, since B has height $O(\log n)$. To find them, we trace the canonical paths for q in B. The partition property holds directly by construction. ◀

2.2 The Query Procedure

In order to solve a batch of queries of type (R), we build a dedicated search structure for each canonical interval, and we solve all queries that encompass one canonical interval in a single go. This means that we have a set of disks in the plane and a set of query points, and we need to determine for each query point whether it is contained in the union of the given disks. To solve this batched query efficiently, we exploit the well-known *lifting map*. Let $U = \{(x, y, z) \mid x^2 + y^2 = z\}$ be the three-dimensional unit paraboloid. For a point $p \in \mathbb{R}^2$, the lifted version $\hat{p}$ of p is its vertical projection onto U. Each disk D_s, for $s \in S$, is mapped to an upper halfspace $\widehat{D}_s$, so that the projection of $\widehat{D}_s \cap U$ onto the xy-plane is the set $\mathbb{R}^2 \setminus D_s$;[1] see Figure 2. Now, the union of a set of disks in $\mathbb{R}^2$ is represented as the intersection of the lifted upper halfspaces in $\mathbb{R}^3$.

▶ **Lemma 2.2.** *The range searching problem (R) can be solved in $O(n \log n)$ expected time.*

Proof. For each $v \in B$, we construct a three-dimensional representation of the union of the disks in the canonical interval $\mathcal{I}_v$. As explained, this is the intersection $\mathcal{E}_v$ of the lifted three-dimensional halfspaces $\widehat{D}_s$, for $s \in \mathcal{I}_v$. The intersection of two three-dimensional convex polyhedra with a total of m vertices can be computed in $O(m)$ time [2, 3]. Therefore, we can construct all the polyhedra $\mathcal{E}_v$, for $v \in B$, in overall $O(n \log n)$ time, by a bottom-up traversal of B (by Lemma 2.1, the total number of vertices of these polyhedra is $O(n \log n)$).

For the batched query processing, we computed a polytope $\widehat{Q}_v$ for each $v \in B$. The polytope $\widehat{Q}_v$ is obtained by determining all the points p that appear in a query (p, r_1, r_2)

[1] This halfspace is bounded by the plane $z = 2x_s x - x_s^2 + 2y_s y - y_s^2 + r_s^2$, where $s = (x_s, y_s)$.

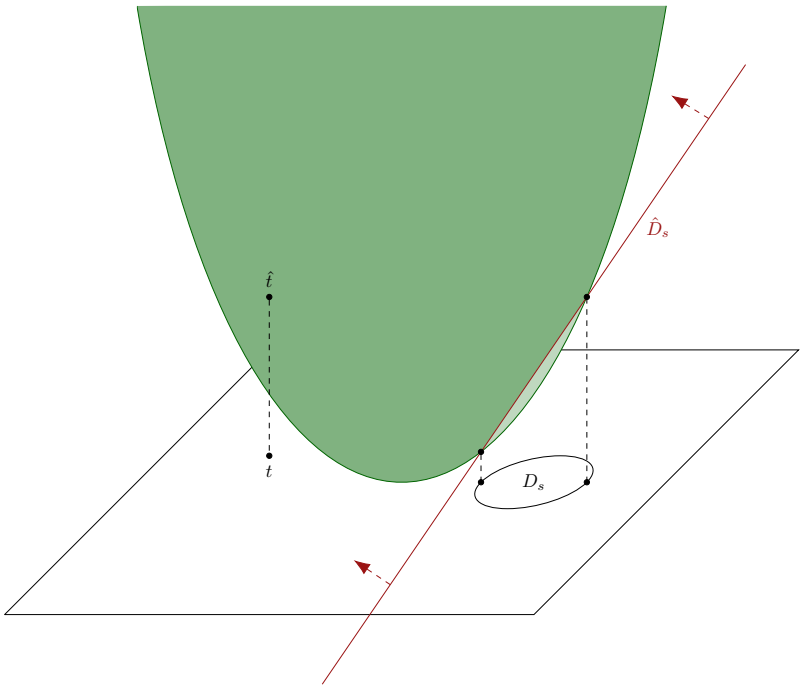


Figure 2 Lifting disks and points. For $\hat{D}$ only the bounding plane is shown.

that has v as a canonical node, lifting those point points p to their three-dimensional representations $\hat{p}$, and taking the convex hull of the resulting three-dimensional point set. The lifted query points all lie on the unit paraboloid U, so every lifted query point appears as a vertex on $\widehat{Q}_v$. To find all polytopes $\widehat{Q}_v$, for $v \in B$, efficiently, we proceed as follows: let A be the three-dimensional point set obtained by taking all points that appear in a query and by lifting them onto the unit paraboloid. We compute the convex hull of A in $O(n \log n)$ time. Then, for each $v \in B$, we find the convex hull of all lifted queries that have v in their canonical path. This can be done in $O(n \log n)$ total expected time by a top-down traversal of B. We already have the polytope for the root of B. To compute the polytope for a child node, given that the polytope for the parent node is available, we use the fact that for any polytope $\mathcal{E}$ in $\mathbb{R}^3$ with m vertices, we can compute the convex hull of any subset of the vertices of $\mathcal{E}$ in $O(m)$ expected time [4]. Once we have for each $v \in B$ the convex hull of the lifted query points that have v on their canonical *path*, we can compute for each $v \in B$ the polytope $\widehat{Q}_v$ that is the convex hull of the lifted query points that have v as a canonical *node*. For this, we consider the canonical path polytope stored at the parent node of v, and we again use the algorithm from [4] to extract the convex hull for the lifted query points that have v as a canonical node.

Now that the polyhedra $\widehat{Q}_v$ and the polytopes $\mathcal{E}_v$ are available, for all $v \in B$, we can answer the queries as follows: for each node $v \in B$, we must find the vertices of $\widehat{Q}_v$ that do not lie inside of $\mathcal{E}_v$. These are exactly the vertices of $\widehat{Q}_v$ that are not vertices of $\widehat{Q}_v \cap \mathcal{E}_v$. As mentioned, the intersections $\widehat{Q}_v \cap \mathcal{E}_v$ can be found in linear time for each node $v \in B$, for a total time $O(n \log n)$, and once the intersection is available, we can easily find all vertices $\hat{p}$ of $\widehat{Q}_v$ that are not vertices of $\widehat{Q}_v \cap \mathcal{E}_v$ (e.g., using radix sort). For any such vertex, we need to find an answer for the corresponding query, if is has not yet been found. For this, we create a data structure that supports ray shooting queries on $\mathcal{E}_v$ in $O(\log n)$ time. This

can be done in $O(|\mathcal{E}_v|)$ time [5], so the overall construction time is $O(n \log n)$. We answer a vertical ray shooting query for the query point p in $\mathcal{E}_v$ in $O(\log n)$ to find a disk in the given radius range that contains p. Since this is done at most once for each query triple, the overall time remains $O(n \log n)$, as desired. ◀

3 Testing for Acyclicity

Now we consider the problem of testing if a given transmission graph is acyclic, where we allow cycles with two edges. Let s, t be two sites such that the edges st and ts both exist in $T(S)$. We call s, t a *double edge*. Double edges in a transmission graph can be characterized by certain configurations between sites:

▶ **Lemma 3.1.** *Let $s, t \in S$ with $t \in D_s$ and $r_s \leq r_t$. Then s, t is a double edge in $T(S)$.*

Proof. The edge ts exists, since $t \in D_s$, or, equivalently, $|st| \leq r_s$. By the second assumption, we get $|st| \leq r_s \leq r_t$, and thus both edges exist. ◀

Now we use Lemma 3.1 to reduce the problem of deciding if $T(S)$ is acyclic to finding double edges in $T(S)$.

▶ **Lemma 3.2.** *A transmission graph $T(S)$ contains a cycle if and only if it contains a double edge.*

Proof. If $T(S)$ contains a double edge, we already have a cycle of length two. Now let $C = s_1, \ldots, s_k$ be a cycle in $T(S)$, such that s_1 is the site of minimum radius in C. We have $s_2 \in D_{s_1}$ by the definition of C and $r_{s_1} \leq r_{s_2}$ by assumption. By Lemma 3.1, we have that s_1, s_2 is the desired double edge. ◀

We use the range query data structure from Section 2.2 to check if a given transmission graph is acyclic.

▶ **Lemma 3.3.** *Let $T(S)$ be a transmission graph on a set S with n sites. In $O(n \log n)$ expected time, we can check if $T(S)$ is acyclic.*

Proof. By Lemma 3.2, it suffices to check if $T(S)$ contains a double edge. We can do this by using the range query data structure as follows. Let $r_{\min}$ be the minimum radius of any input disk. For each site t, we create a query triple $(t, r_{\min}, r_t)$ and perform a batched range query (R) with these triples on all sites. If there is a site t for which the query returns a value, we have, by the choice of the queries, that t is contained in the disk for some smaller site. This directly implies the existence of a double edge by Lemma 3.1. In the other case, we know that there are no double edges and we can conclude that the graph is acyclic. ◀

4 Conclusion

We showed how to check a transmission graph for acyclicity in $O(n \log n)$ time, when we include cycles of length two. Right now, we are working on the same setting, while disallowing these cycles.

Regarding the range query data structure, it would be interesting to derandomize it within the same time bound. Furthermore, at the moment we are only able to find one disk containing each given point. In particular with regard to applications in transmission graphs, it would be useful to be able to find all disks containing each query point, in overall $O((n + k) \log n)$ time, where k is the overall number of disks reported.

References

1 Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark H. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, third edition, 2008.

2 Timothy M. Chan. A simpler linear-time algorithm for intersecting two convex polyhedra in three dimensions. *Discrete Comput. Geom.*, 56(4):860–865, December 2016.

3 Bernard Chazelle. An optimal algorithm for intersecting three-dimensional convex polyhedra. *SIAM J. Comput.*, 21(4):671–696, 1992.

4 Bernard Chazelle and Wolfgang Mulzer. Computing hereditary convex structures. *Discrete Comput. Geom.*, 45(4):796–823, 2011.

5 David P. Dobkin and David G. Kirkpatrick. Fast detection of polyhedral intersections. In *Proc. 9th Internat. Colloq. Automata Lang. Program. (ICALP)*, pages 154–165, 1982.

6 Hiroshi Imai, Masao Iri, and Kazuo Murota. Voronoi diagram in the Laguerre geometry and its applications. *SIAM J. Comput.*, 14(1):93–105, 1985.

7 Haim Kaplan, Katharina Klost, Wolfgang Mulzer, and Liam Roditty. Finding the girth in disk graphs and a directed triangle in transmission graphs. In *Proc. 34th European Workshop Comput. Geom. (EWCG)*, pages 68:1–6, 2018.

8 Haim Kaplan, Wolfgang Mulzer, Liam Roditty, and Paul Seiferth. Spanners for directed transmission graphs. *SIAM J. Comput.*, 47(4):1585–1609, 2018.

Dynamic Maintenance of the Lower Envelope of Pseudo-Lines

Pankaj K. Agarwal[1], Ravid Cohen[2], Dan Halperin[2], and Wolfgang Mulzer[3]

1 **Department of Computer Science, Duke University, Durham, NC 27708, USA**
pankaj@cs.duke.edu

2 **School of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel**

3 **Institut für Informatik, Freie Universität Berlin, D-14195 Berlin, Germany**
mulzer@inf.fu-berlin.de

Abstract

We present a fully dynamic data structure for the maintenance of lower envelopes of pseudo-lines. The structure has $O(\log^2 n)$ update time and $O(\log n)$ vertical ray shooting query time. To achieve this performance, we devise a new algorithm for finding the intersection between two lower envelopes of pseudo-lines in $O(\log n)$ time, using *tentative* binary search; the lower envelopes are special in that at $x = -\infty$ any pseudo-line contributing to the first envelope lies below every pseudo-line contributing to the second envelope. The structure requires $O(n)$ storage space.

1 Introduction

A set of pseudo-lines in the plane is a set of infinite x-monotone curves each pair of which intersects at exactly one point. Arrangements of pseudo-lines have been intensively studied in discrete and computational geometry; see the recent survey on arrangements [7] for a review of combinatorial bounds and algorithms for arrangements of pseudo-lines. In this paper we consider the following problem: Given n pseudo-lines in the plane, dynamically maintain their lower envelope such that one can efficiently answer vertical ray shooting queries from $y = -\infty$. The dynamization is under insertions and deletions. If we were given n lines (rather than pseudo-lines) then we could have used any of several efficient data structures for the purpose [3–5, 9, 10]; these are, however, not directly suitable for pseudo-lines. There are several structures that rely on shallow cuttings and can handle pseudo-lines [2,6,8]. The solution that we propose here is, however, considerably more efficient than what these structures offer. We devise a fully dynamic data structure with $O(\log^2 n)$ update-time, $O(\log n)$ vertical ray-shooting query-time, and $O(n)$ space for the maintenance of n pseudo-lines. The structure is a rather involved adaptation of the Overmars-van Leeuwen structure [10] to our setting, which matches the performance of the original algorithm for the case of lines. The key innovation is a new algorithm for finding the intersection between two lower envelopes of planar pseudo-lines in $O(\log n)$ time, using *tentative* binary search (where each pseudo-line in one envelope is "smaller" than every pseudo-line in the other envelope in a sense to be made precise below). To the best of our knowledge this is the most efficient data structure for the case of pseudo-lines to date.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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 E be a finite family of pseudo-lines in the plane, and let ℓ be a vertical line strictly to the left of the left-most intersection point between lines in E (namely to the left of all the vertices of the arrangement $\mathcal{A}(E)$). The line ℓ defines a total order $\leq$ on the pseudo-lines in E, namely for $e_1, e_2 \in E$, we have $e_1 \leq e_2$ if and only if e_1 intersects ℓ below e_2. Since each pair of pseudo-lines in E crosses exactly once, it follows that if we consider a vertical line ℓ' strictly to the right of the right-most vertex of $\mathcal{A}(E)$, the order of the intersection points between ℓ' and E, from bottom to top, is exactly reversed.

The *lower envelope* $\mathcal{L}(E)$ of E is the x-monotone curve obtained by taking the pointwise minimum of the pseudo-lines in E. Combinatorially, the lower envelope $\mathcal{L}(E)$ is a sequence of connected segments of the pseudo-lines in E, where the first and last segment are unbounded. Two properties are crucial for our data structure: (A) every pseudo-line contributes at most one segment to $\mathcal{L}(E)$; and (B) the order of these segments corresponds exactly to the order $\leq$ on E defined above. In fact, our data structure works for every set of planar curves with properties (A) and (B) (with an appropriate order $\leq$), even if they are not pseudo-lines in the strict sense.

We assume a computational model in which primitive operations on pseudo-lines, such as computing the intersection point of two pseudo-lines or determining the intersection point of a pseudo-line with a vertical line can be performed in constant time.

3 Data structure and operations

The tree structure. Our primary data structure is a balanced binary search tree Ξ. Such a tree data structure supports insert and delete, each in $O(\log n)$ time. The leaves of Ξ contain the pseudo-lines, from left to right in the sorted order defined above. An internal node $v \in \Xi$ represents the lower envelope of the pseudo-lines in its subtree. More precisely, every leaf v of Ξ stores a single pseudo-line $e_v \in E$. For an inner node v of Ξ, we write $E(v)$ for the set of pseudo-lines in the subtree rooted at v. We denote the lower envelope of $E(v)$ by $\mathcal{L}(v)$. The inner node v has the following variables:

- f, ℓ, r: a pointer to the parent, left child and right child of v, respectively;
- max: the maximum pseudo-line in $E(v)$;
- Λ: a balanced binary search tree that stores the prefix or suffix of $\mathcal{L}(v)$ that is not on the lower envelope $\mathcal{L}(f)$ of the parent (in the root, we store the lower envelope of E). The leaves of Λ store the pseudo-lines that support the segments on the lower envelope, with the endpoints of the segments, sorted from left to right. An inner node of Λ stores the common point of the last segment in the left subtree and the first segment in the right subtree. We will need split and join operations on the binary trees, which can be implemented in $O(\log n)$ time.

Queries. We now describe the query operations available on our data structure. In a *vertical ray-shooting query*, we are given a value $x_0 \in \mathbb{R}$, and we would like to find the pseudo-line $e \in E$ where the vertical line $\ell : x = x_0$ intersects $\mathcal{L}(E)$. Since the root of Ξ explicitly stores $\mathcal{L}(E)$ in a balanced binary search tree, this query can be answered easily in $O(\log n)$ time.

▶ **Lemma 3.1.** *Let $\ell : x = x_0$ be a vertical ray shooting query. We can find the pseudo-line(s) where ℓ intersects $\mathcal{L}(E)$ in $O(\log n)$ time.*

Proof. Let r be the root of Ξ. We perform an explicit search for x_0 in $r.\Lambda$ and return the result. Since $r.\Lambda$ is a balanced binary search tree, this takes $O(\log n)$ time. ◀

Update. To insert or delete a pseudo-line e in Ξ, we follow the method of Overmars and van Leeuwen [10]. We delete or insert a leaf for e in Ξ using standard binary search tree techniques (the $v.\max$ pointers guide the search in Ξ). As we go down, we construct the lower envelopes for the nodes hanging off the search path, using split and join operations on the $v.\Lambda$ trees. Going back up, we recompute the information $v.\Lambda$ and $v.\max$. To update the $v.\Lambda$ trees, we need the following operation: given two lower envelopes $\mathcal{L}_\ell$ and $\mathcal{L}_r$, such that all pseudo-lines in $\mathcal{L}_\ell$ are smaller than all pseudo-lines in $\mathcal{L}_r$, compute the intersection point q of $\mathcal{L}_\ell$ and $\mathcal{L}_r$. In the next section, we will see how to do this in $O(\log n)$ time, where n is the size of E. Since there are $O(\log n)$ nodes in Ξ affected by an update, this procedure takes $O(\log^2 n)$ time. More details can be found in the literature [10, 11].

▶ **Lemma 3.2.** *It takes $O(\log^2 n)$ to insert or remove a pseudo-line in Ξ.*

4 Finding the intersection point of two lower envelopes

Given two lower envelopes $\mathcal{L}_\ell$ and $\mathcal{L}_r$ such that all pseudo-lines in $\mathcal{L}_\ell$ are smaller than all pseudo-lines in $\mathcal{L}_r$, we would like to find the intersection point q between $\mathcal{L}_\ell$ and $\mathcal{L}_r$. We assume that $\mathcal{L}_\ell$ and $\mathcal{L}_r$ are represented as balanced binary search trees. The leaves of $\mathcal{L}_\ell$ and $\mathcal{L}_r$ store the pseudo-line segments on the lower envelopes, sorted from left to right. We assume that the pseudo-line segments in the leaves are half-open, containing their right, but not their left endpoint in $\mathcal{L}_\ell$; and their left, but not their right endpoint in $\mathcal{L}_r$.[1] Thus, it is uniquely determined which leaves of $\mathcal{L}_\ell$ and $\mathcal{L}_r$ contain the intersection point q. A leaf v stores the pseudo-line $\mathcal{L}(v)$ that supports the segment for v, as well as an endpoint $v.p$ of the segment, namely the left endpoint if v is a leaf of $\mathcal{L}_\ell$, and the right endpoint if v is a leaf of $\mathcal{L}_r$.[2] An inner node v stores the intersection point $v.p$ between the largest pseudo-line in the left subtree $v.\ell$ of v and the smallest pseudo-line in the right subtree $v.r$ of v, together with the lower envelope $\mathcal{L}(v)$ of these two pseudo-lines. These trees can be obtained by appropriate split and join operations from the Λ trees stored in Ξ.

Let $u^* \in \mathcal{L}_\ell$ and $v^* \in \mathcal{L}_r$ be the leaves whose segments contain q. Let π_ℓ be the path in $\mathcal{L}_\ell$ from the root to u^* and π_r the path in $\mathcal{L}_r$ from the root to v^*. Our strategy is as follows: we simultaneously descend in $\mathcal{L}_\ell$ and in $\mathcal{L}_r$. Let u be the current node in $\mathcal{L}_\ell$ and v the current node in $\mathcal{L}_r$. In each step, we perform a local test on u and v to decide how to proceed. There are three possible outcomes:

1. $u.p$ is on or above $\mathcal{L}(v)$: the intersection point q is equal to or to the left of $u.p$. If u is an inner node, then u^* cannot lie in $u.r$; if u is a leaf, then u^* lies strictly to the left of u;
2. $v.p$ lies on or above $\mathcal{L}(u)$: the intersection point q is equal to or to the right of $v.p$. If v is an inner node, then v^* cannot lie in $v.\ell$; if v is a leaf, then v^* lies strictly to the right of v;
3. $u.p$ lies below $\mathcal{L}(v)$ and $v.p$ lies below $\mathcal{L}(u)$: then, $u.p$ lies strictly to the left of $v.p$ (since we are dealing with pseudo-lines). It must be the case that $u.p$ is strictly to the left of q or $v.p$ is strictly to the right of q (or both). In the former case, if u is an inner node, u^* lies in or to the right of $u.r$ and if u is a leaf, then u^* is u or a leaf to the right of u. In the latter case, if v is an inner node, v^* lies in or to the left of $v.\ell$ and if v is a leaf, then v^* is v or a leaf to the left of v; see Figure 1.

[1] We actually store both endpoints in the trees, but the intersection algorithm uses only one of them, depending on the role the tree plays in the algorithm.

[2] If the segment is unbounded, the endpoint might not exist. In this case, we use a symbolic endpoint at infinity that lies below every other pseudo-line.

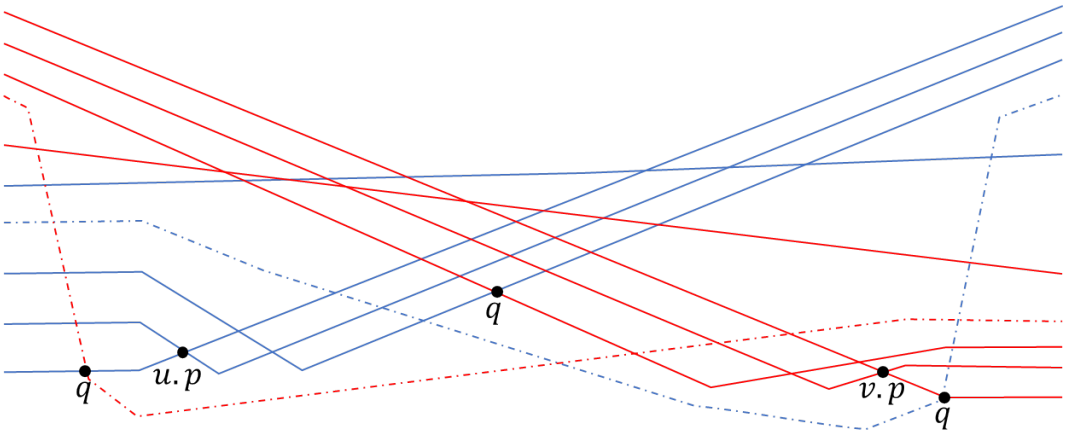


Figure 1 An example of Case 3. $\mathcal{L}_\ell$ is blue; $\mathcal{L}_r$ is red. The solid pseudo-lines are fixed. The dashed pseudo-lines are optional, namely, either none of the dashed pseudo-lines exists or exactly one of them exists. $u.p$ and $v.p$ are the current points; and Case 3 applies. Irrespective of the local situation at u and v, the intersection point can be to the left of $u.p$, between $u.p$ and $v.p$ or to the right of $v.p$, depending on which one of the dashed pseudo-lines exists.

Overmars and van Leeuwen [10, 11] describe a method for the case that $\mathcal{L}_\ell$ and $\mathcal{L}_r$ contain lines. Unfortunately, it is not clear how their strategy applies in the more general setting of pseudo-lines. The reason for this lies in Case 3: in this case, it is not immediately obvious how to proceed, because the correct step might be either to go to $u.r$ or to $v.\ell$. In the case of lines, Overmars and van Leeuwen can solve this ambiguity by comparing the slopes of the relevant lines. For pseudo-lines, however, this does not seem to be possible. For an example, refer to Figure 1, where the local situation at u and v does not determine the position of the intersection point q. Therefore, we present an alternative strategy.

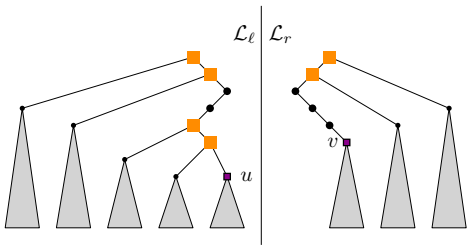


Figure 2 The invariant: the current search nodes are u and v. `uStack` contains all nodes on the path from the root to u where the path goes to a right child (orange squares), `vStack` contains all nodes from the root to v where the path goes to a left child (orange squares). The final leaves u^* and v^* are in one of the gray subtrees; and at least one of them is under u or under v.

We will maintain the invariant that the subtree at u contains u^* or the subtree at v contains v^* (or both). When comparing u and v, one of the three cases occurs. In Case 3, u^* must be in $u.r$, or v^* must be in $v.\ell$; see Figure 3.

We move u to $u.r$ and v to $v.\ell$. One of these moves must be correct, but the other move might be mistaken: we might have gone to $u.r$ even though u^* is in $u.\ell$ or to $v.\ell$ even though v^* is in $v.r$. To correct this, we remember the current u in a stack `uStack` and the current v in a stack `vStack`, so that we can revisit $u.\ell$ or $v.r$ if it becomes necessary. This leads to the general situation shown in Figure 2: u^* is below u or in a left subtree of a node on `uStack`,

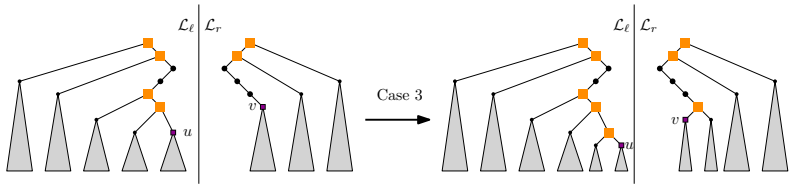


Figure 3 Comparing u to v: in Case 3, we know that u^* is in $u.r$ or v^* is in $v.\ell$; we go to $u.r$ and to $v.\ell$.

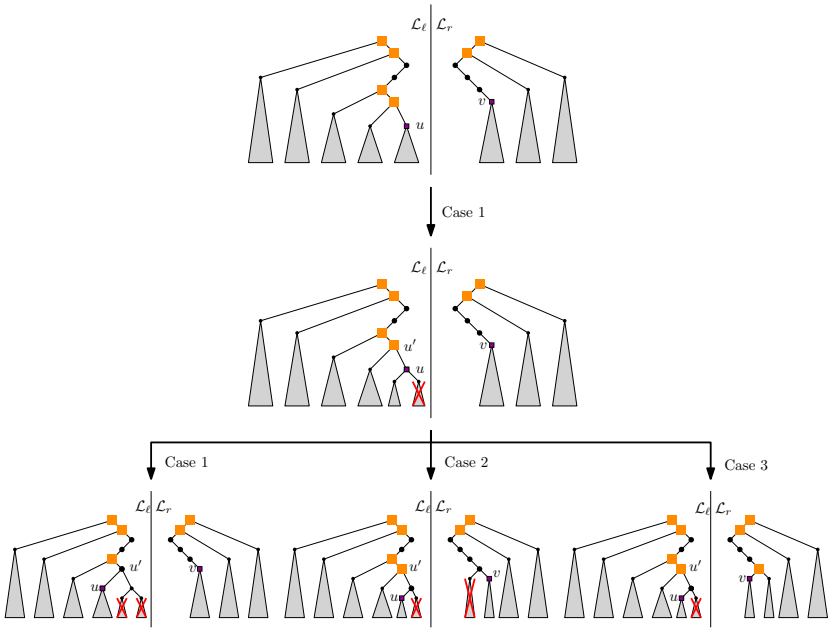
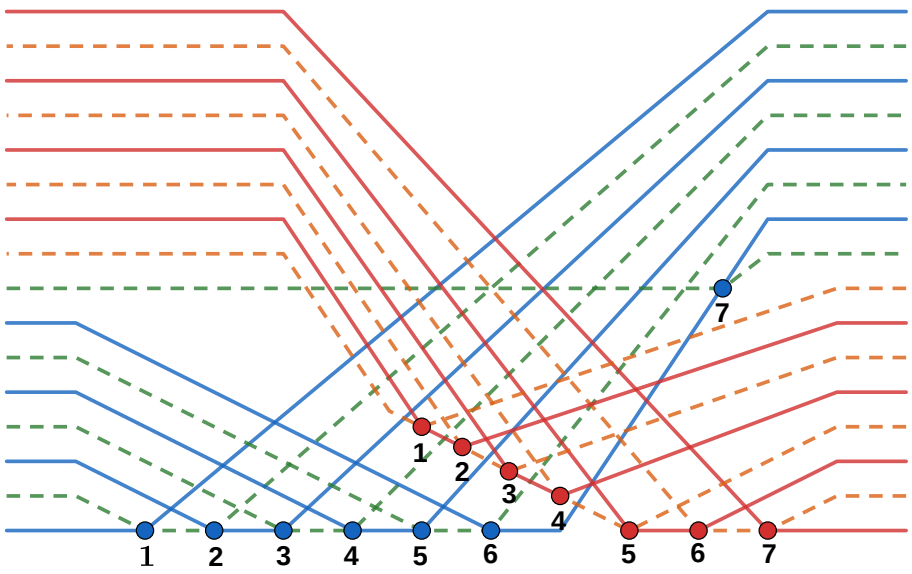
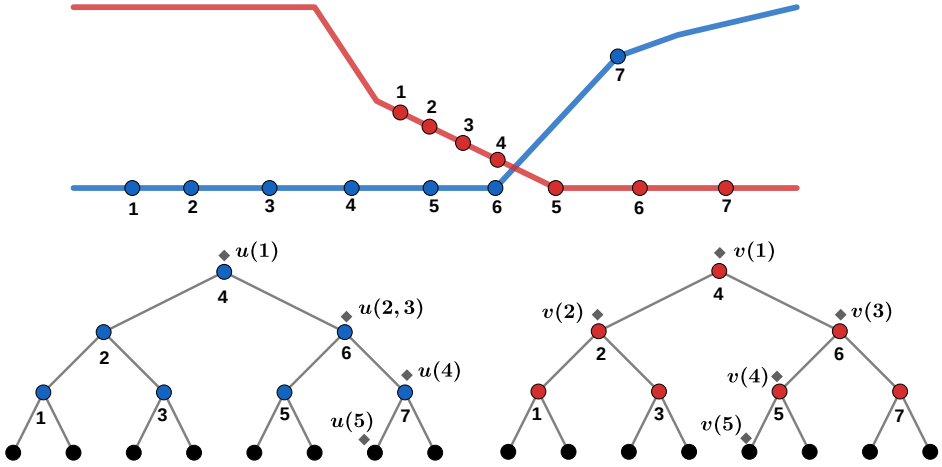


Figure 4 Comparing u to v: in Case 1, we know that u^* cannot be in $u.r$. We compare u' and v to decide how to proceed: in Case 1, we know that u^* cannot be in $u'.r$; we go to $u'.\ell$; in Case 2, we know that u^* cannot be in $u.r$ and that v^* cannot be in $v.\ell$; we go to $u.\ell$ and to $v.r$; in Case 3, we know that u^* is in $u'.r$ (and hence in $u.\ell$) or in $v.\ell$; we go to $u.\ell$ and to $v.\ell$. Case 2 is not shown as it is symmetric.

and v^* is below v or in a right subtree of a node on `vStack`, and at least one of u^* or v^* must be below u or v, respectively. Now, if Case 1 occurs when comparing u to v, we can exclude the possibility that u^* is in $u.r$. Thus, u^* might be in $u.\ell$, or in the left subtree of a node in `uStack`; see Figure 4. To make progress, we now compare u', the top of `uStack`, with v. Again, one of the three cases occurs. In Case 1, we can deduce that going to $u'.r$ was mistaken, and we move u to $u'.\ell$, while v does not move. In the other cases, we cannot rule out that u^* is to the right of u', and we move u to $u.\ell$, keeping the invariant that u^* is either below u or in the left subtree of a node on `uStack`. However, to ensure that the search progresses, we now must also move v. In Case 2, we can rule out $v.\ell$, and we move v to $v.r$. In Case 3, we move v to $v.\ell$. In this way, we keep the invariant and always make progress: in each step, we either discover a new node on the correct search paths, or we pop one erroneous move from one of the two stacks. Since the total length of the correct search paths is $O(\log n)$, and since we push an element onto the stack only when discovering a new correct node, the total search time is $O(\log n)$; see Figure 5 for an example run. For the full pseudocode and the formal proof see [1].



(a) Demonstration of two set of pseudo-lines and their lower envelope: (i) the blue and green pseudo-lines, (ii) the red and orange pseudo-lines. The blue and the red dots represents the intersection points on the lower envelopes.



(b) The top figure shows the lower envelope of (a). The bottom figure shows the the trees which maintain the lower envelopes. $u(i)$ and $v(i)$ shows the position of the pointers u and v at step i, during the search procedure.

Figure 5 Example of finding the intersection point of two lower envelopes:

Step	u	v	uStack	vStack	Procedure case
1	4	4	$\emptyset$	$\emptyset$	Case 3
2	6	2	4	4	Case 2 $\rightarrow$ Case 2
3	6	6	4	$\emptyset$	Case 3
4	7	5	4, 6	6	Case 1 $\rightarrow$ Case 3
5	7*	5*	4, 6	6, 5	Case 3 $\rightarrow$ End

Acknowledgments. We thank Haim Kaplan and Micha Sharir for helpful discussions. Work by P.A. has been supported by NSF under grants CCF-15-13816, CCF-15-46392, and IIS-14-08846, by ARO grant W911NF-15-1-0408, and by grant 2012/229 from the U.S.-Israel Binational Science Foundation. Work by D.H. and R.C. has been supported in part by the Israel Science Foundation (grant no. 825/15), by the Blavatnik Computer Science Research Fund, by the Blavatnik Interdisciplinary Cyber Research Center at Tel Aviv University, and by grants from Yandex and from Facebook. Work by W.M. has been partially supported by ERC STG 757609 and GIF grant 1367/2016.

References

1 Pankaj K. Agarwal, Ravid Cohen, Dan Halperin, and Wolfgang Mulzer. Dynamic maintenance of the lower envelope of pseudo-lines, 2019. `arXiv:arXiv:1902.09565`.

2 Pankaj K. Agarwal and Jiří Matoušek. Dynamic half-space range reporting and its applications. *Algorithmica*, 13(4):325–345, 1995.

3 Gerth Stølting Brodal and Riko Jacob. Dynamic planar convex hull with optimal query time. In *Algorithm Theory - SWAT 2000, 7th Scandinavian Workshop on Algorithm Theory, Bergen, Norway, July 5-7, 2000, Proceedings*, pages 57–70, 2000.

4 Gerth Stølting Brodal and Riko Jacob. Dynamic planar convex hull. In *Proceedings of the 43rd Symposium on Foundations of Computer Science (FOCS 2002)*, pages 617–626, 2002.

5 Timothy M. Chan. Dynamic planar convex hull operations in near-logarithmaic amortized time. *J. ACM*, 48(1):1–12, 2001.

6 Timothy M. Chan. A dynamic data structure for 3-d convex hulls and 2-d nearest neighbor queries. *J. ACM*, 57(3):16:1–16:15, 2010.

7 Dan Halperin and Micha Sharir. Arrangements. In Jacob E. Goodman, Joseph O'Rourke, and Csaba Tóth, editors, *Handbook of Discrete and Computational Geometry*, chapter 28. Chapman & Hall/CRC, 3rd edition, 2017.

8 Haim Kaplan, Wolfgang Mulzer, Liam Roditty, Paul Seiferth, and Micha Sharir. Dynamic planar Voronoi diagrams for general distance functions and their algorithmic applications. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 2495–2504, 2017.

9 Haim Kaplan, Robert Endre Tarjan, and Kostas Tsioutsiouliklis. Faster kinetic heaps and their use in broadcast scheduling. In *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms, January 7-9, 2001, Washington, DC, USA.*, pages 836–844, 2001.

10 Mark H. Overmars and Jan van Leeuwen. Maintenance of configurations in the plane. *Journal of Computer and System Sciences*, 23(2):166–204, 1981.

11 Franco P. Preparata and Michael Ian Shamos. *Computational Geometry. An Introduction.* Springer-Verlag, New York, 1985.

Computing α-Shapes for Temporal Range Queries on Point Sets

Annika Bonerath[1], Jan-Henrik Haunert[1], and Benjamin Niedermann[1]

1 **Institute of Geodesy and Geoinformation, University of Bonn**
(bonerath,haunert,niedermann)@igg.uni-bonn.de

Abstract

The interactive exploration of data requires data structures that can be repeatedly queried to obtain simple visualizations of parts of the data. In this paper we consider the scenario that the data is a set of points each associated with a time stamp and that the result of each query is visualized by an α-shape, which generalizes the concept of convex hulls. Instead of computing each shape independently, we suggest and analyze a simple data structure that aggregates the α-shapes of all possible queries. Once the data structure is built, it particularly allows us to query single α-shapes without retrieving the actual (possibly large) point set and thus to rapidly produce small previews of the queried data.

1 Introduction

In scientific projects that deal with large amounts of spatio- and temporal data, the data management is essential. As an example take a project dealing with a database of storm events of the United States; see Figure 1. Each storm event is a data point with a geo-location and a time stamp. Assuming a collection of storm events over several decades the amount of data becomes enormous. On the other hand, for certain scientific questions the user may not be interested in all data, but only in a subset in a pre-defined temporal range. Hence, before

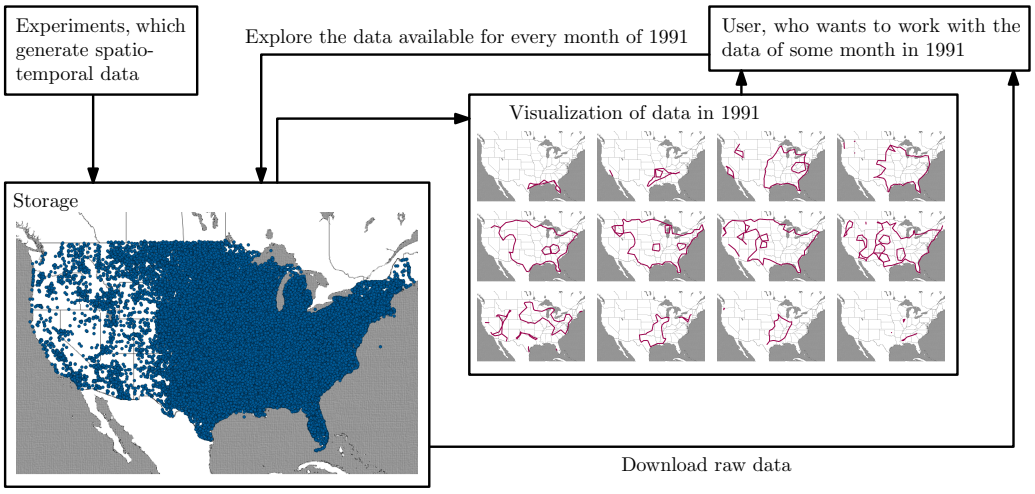


Figure 1 Scenario for the case that the user queries simplified visualizations for all storm events in the year 1991 broken down to months. The α-shapes (lilac) were generated with our approach. Data retrieved from Data.gov. Map tiles by Stamen Design, under CC BY 3.0. Data by OpenStreetMap, under ODbL.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

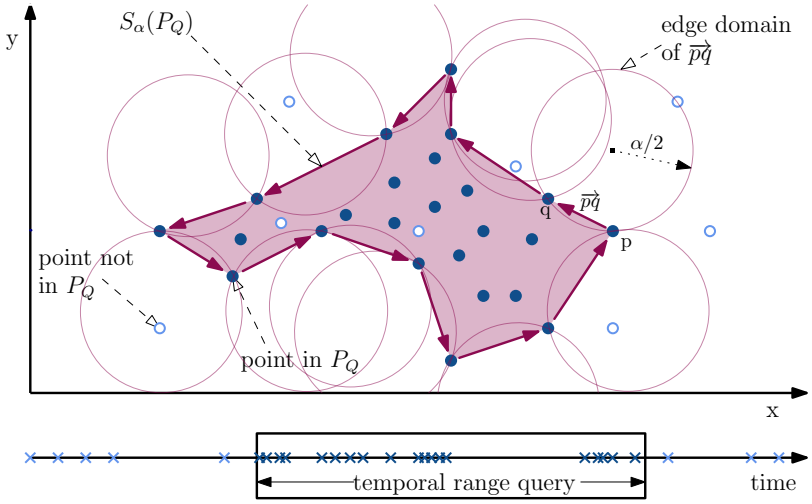


Figure 2 The α-shape (lilac arcs) for a point set (filled blue disks) of a temporal range query.

downloading the actual data for a thorough analysis, the user may be interested in exploring the data by querying simplified visualizations of the data within temporal ranges.

One approach to create a simplified visualization is to sketch the outline of the queried data set providing the user with the possibility of roughly assessing the spatial distribution of the data. For example, the convex hull is a simple polygonal representation for that purpose. However, for most data sets this representation is not adequate, because the convex hull may easily cover large areas that do not contain any points of the data set. A wide range of more sophisticated polygonal representations exists; some of these are based on Delaunay-triangulations and shortest-path graphs [5, 6, 9] while others use spatial grids to define the representation [1, 3, 10, 13]. In this paper we use α-shapes [8, 9] for representing point sets, which are a generalization of convex hulls and strongly related to Delaunay-triangulations. Among others, this technique finds its application in digital shape sampling and processing [2], in pattern recognition [14, 15] and micro-biology [7, 11].

An α-shape of a set $P \subset \mathbb{R}^2$ of n points in the plane is defined as follows. Let $\alpha > 0$. The *edge domain* of a directed edge $pq \in P \times P$ with $|q - p| \leq \alpha$ is the open disk D_{pq} with radius $\frac{\alpha}{2}$ whose center lies to the right of pq and whose boundary contains the points p and q. The set $S_\alpha(P) \subseteq P \times P$ of all edges that are shorter than α and do not contain any point of P in their edge domain is called *α-shape*; see Figure 2. It can be computed in $O(n \log n)$ time [9].

In our use-case each point $p \in P$ additionally is associated with a *time stamp* $t_p \in \mathbb{R}$; we assume that all points in P have pairwise distinct spatial and temporal coordinates. As described in the running example the point set P is queried frequently. Such a *query* Q is a temporal range $[t_Q^{\text{start}}, t_Q^{\text{end}}]$ and its result is the subset $P_Q = \{p \in P \mid t_p \in [t_Q^{\text{start}}, t_Q^{\text{end}}]\}$. We are then interested in visualizing P_Q by its α-shape. A straight-forward approach for a query Q first queries the set P obtaining P_Q and then computes the α-shape $S_\alpha(P_Q)$. Utilizing a balanced binary search-tree, finding P_Q takes $O(\log n + |P_Q|)$ time. Additionally computing the α-shape we obtain $O(\log n + |P_Q| \log |P_Q|)$ running time in total. For our use-case of frequently providing α-shapes for visualizing the query results, we aim at a better running time per query. In particular, for creating previews of the data, we only want to retrieve the α-shape of P_Q but not the entire set P_Q. In a pre-processing phase we compute a data structure that aggregates the α-shapes of all possible queries; we call it the *α-structure* of P. We use this data structure in the query phase to obtain the α-shapes of the incoming queries.

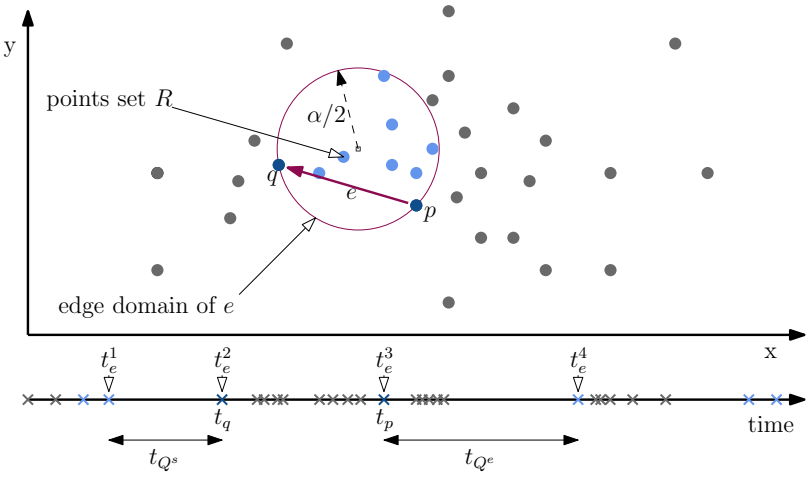


Figure 3 The edge domain with its contained point set R and the time attributes of an edge pq.

As we show in Section 3 the α-structure leads to quadratic memory consumption in the worst case. However, a detailed analysis for points sets whose spatial distribution is uniform and uncorrelated to their temporal distribution shows that the size of the α-structure is more complaisant. In Section 4 we present an algorithm that computes an α-structure in $O(n(\log n + m_{\mathrm{R}} \log m_{\mathrm{R}}))$ time utilizing linear and rotational sweeps, where m_{R} denotes the maximum number of points $p \in P$ in a square of width 2α. For the query phase we use a data structure for filtering search to answer a query in $O(\log n + k)$ time, where k is the size of the returned α-shape [4]. In Section 5 we present our initial experiments on real-world data showing that the α-structure is applicable in our concrete use case.

2 On α-Structures

In the following we define the α-structure of P. We say that an edge $pq \in P \times P$ is *active* for a temporal query Q if the α-shape $S_\alpha(P_Q)$ contains pq. We observe that an edge pq can be active for an infinite set of temporal queries, but it can only be active for $O(n^2)$ different subsets of P. To characterize this set, we introduce the following notation; see Figure 3.

Let $e = pq \in P \times P$ with $t_p < t_q$, and let $R \subseteq P$ be the set of points contained in the edge domain of pq. Further, let t_r with $r \in R$ be the largest time stamp that is smaller than t_p; if r does not exist, we set $t_r = -\infty$. The *minimal query start time* is $t_e^1 := t_r$ and the *maximal query start time* is $t_e^2 := t_p$. Similarly, let t_s with $s \in R$ be the smallest time stamp that is greater than t_q; if s does not exist, we set $t_s = \infty$. The *minimal query end time* is $t_e^3 := t_q$ and the *maximal query end time* is $t_e^4 := t_s$. We call t_e^1, t_e^2, t_e^3, t_e^4 the *time attributes* of pq. The next lemma characterizes for which queries a particular edge is active.

▶ **Lemma 2.1.** *The edge $e = pq$ is active for a query Q if and only if:*

1. *The distance between p and q is smaller than α, and*
2. *$\forall r \in R : t_r \notin [t_p, t_q]$, and*
3. *$t_Q^{\mathrm{start}} \in \left[t_e^1, t_e^2\right]$ and $t_Q^{\mathrm{end}} \in \left[t_e^3, t_e^4\right]$.*

Proof. Assume that e is active for Q. This is equivalent to the following three conditions; (i) p and q are contained in P_Q, which is equivalent to $t_p, t_q \in \left[t_Q^{\mathrm{start}}, t_Q^{\mathrm{end}}\right]$, (ii) e is shorter than

α (equivalent to Condition (1) of Lemma 2.1) and (iii) no point $r \in P_Q$ is contained in R, which is equivalent to $\forall r \in R : t_r \notin \left[t_Q^{\text{start}}, t_Q^{\text{end}}\right]$. Applying the definition of t_e^1, t_e^2, t_e^3, and t_e^4 the Conditions (i) and (iii) are equivalent to Condition (2) and (3) of Lemma 2.1. ◀

The *α-structure* $\mathcal{S}_\alpha(P) \subseteq P \times P$ of P is the set of all active edges over all possible temporal queries. We show that Condition (1) and (2) of Lemma 2.1 are necessary and sufficient for an edge pq to be contained in $\mathcal{S}_\alpha$.

▶ **Lemma 2.2.** *The edge $e = pq \in P \times P$ is contained in $\mathcal{S}_\alpha(P)$ if and only if:*

1. *The distance between p and q is smaller than α.*
2. $\forall r \in R : t_r \notin [t_p, t_q]$

Proof. Let $e \in \mathcal{S}_\alpha(P)$, and let Q be a temporal range query for which $e \in \alpha(P_Q)$. Then e fulfills the conditions of Lemma 2.1 and therefore the conditions of Lemma 2.2. Conversely, let e be shorter than α and all points $r \in R$ be temporally not in $[t_p, t_q]$. Then the α-shape of the query Q with $t_Q^{\text{start}} = t_p$ and $t_Q^{\text{end}} = t_q$ contains the edge e. ◀

3 Memory Consumption

For our use-case of a database the memory consumption of our approach is decisive for being deployed in practice. We first observe that $O(n^2)$ is an upper bound for the size of an α-structure. The following theorem shows that this is also a lower bound in the worst case.

▶ **Theorem 3.1.** *For a set P of n points the α-structure has size $\Omega(n^2)$ in the worst case.*

Proof. Let $P = \{p_1, p_2, \ldots, p_n\}$ be a point set with time stamps $t_1 < t_2 < \ldots < t_n$ such that the points lie on a circle C of radius $r < \frac{1}{2}\alpha$ ordered clockwise according to their time stamps; see Figure 4. Let $p_i, p_j \in P$ be two points with $t_i < t_j$. We show that $p_i p_j$ is contained in the α-structure $\mathcal{S}_\alpha(P)$ by proving the two conditions of Lemma 2.2. Due to $r < \frac{1}{2}\alpha$ the points p_i, p_j have distance smaller than α. Hence, Condition (1) of Lemma 2.2 is satisfied.

For the second condition let R_{ij} be the set of points contained in edge domain D_{ij} of $p_i p_j$. We observe that D_{ij} and C intersect in p_i and p_j. Since the radius of C is smaller than

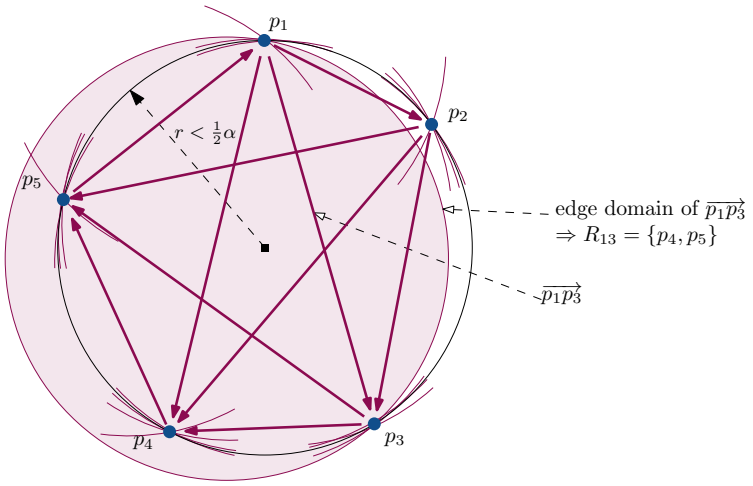


Figure 4 Worst-case example for the size of the α-structure as described in Theorem 3.1.

the radius of D_{ij}, the boundary of D_{ij} partitions C into two parts. One part is contained in D_{ij} and the other lies outside of D_{ij}. Since the points $p_1, p_2, \ldots, p_n$ appear in clockwise order on C, and since the center of D_{ij} lies to the right of $p_i p_j$ by definition, we obtain $R_{ij} = \{p_1, \ldots, p_{i-1}, p_{j+1}, \ldots, p_{n-1}\}$. Consequently, Condition (2) is satisfied. ◀

Hence, the database may exceed a size that is applicable in practice. However, the example is rather unlikely to occur in practice. The next theorem indicates that the data structure is more complaisant than the worst case example suggests. Following our use case we assume that the points are contained in a rectangle $\mathbb{B}$ of width and height at least 2α.

▶ **Theorem 3.2.** *For a finite set $P \subseteq \mathbb{B}$ of n points for which the spatial distribution is uniform in $\mathbb{B}$ and the spatial distance is uncorrelated to the temporal distance the α-structure has expected size $O(n)$.*

To prove Theorem 3.2 we show that the expected size of $\mathcal{S}_\alpha$ is in $O(nm/\kappa)$, where m is the expected number of points in the CPN of a point over all points and κ is the expected number of points in an edge domain over all possible edge domains. Since the area covered by a CPN and an edge domain has a fixed ratio together with a uniform density distribution we can show that m/κ is in $O(1)$. In our experiments on real-world data we also observe a linear relation between the number of points and the size of the α-structure; see Section 5.

4 Constructing and Querying α-Structures

We introduce an algorithm that computes an α-structure of a point set P in $O(n(\log n + m_{\mathrm{R}} \log m_{\mathrm{R}}))$ time and describe how to query this data structure. The construction algorithm applies two steps for each point $p \in P$; see Algorithm 1. The first step, which we call *CPN-Search*, computes all points $T_p \subseteq P$ that fulfill Condition (1) of Lemma 2.2, i.e., all points that lie in a circle with center at p and radius α. We call this circle the *circle of potential neighbors (CPN)* of p. We use the sweep line approach by Peng and Wolff [12] to find T_p in $O(\log n + m_{\mathrm{R}})$ time. The second step, which we call *CPN-Check*, checks for each point $q \in T_p$ whether the edge pq fulfills Condition (2) of Lemma 2.2. If this is the case it computes the time attributes of pq. To implement this efficiently, we use a rotational sweep. More precisely, we use a circle C of radius $\frac{\alpha}{2}$ which sweeps around p such that the center of C moves along the circle with center p and radius $\frac{\alpha}{2}$; see Figure 5. We call C the *sweep circle* of p. Let $\mathcal{R}$ be the points contained in C; we represent $\mathcal{R}$ using a binary search tree ordered by the time stamps of the points. The sweep circle C stops its rotation whenever its boundary intersects with a point $q \in T_p$. Two kind of events are possible; either the point q enters C, or it leaves C. Whenever a point q enters C, the sweep circle equals the edge domain of pq. Utilizing the properties of the binary search tree $\mathcal{R}$ Condition (2) of Lemma 2.2 can be checked in $O(\log m_{\mathrm{R}})$ time. If this is the case the time attributes of pq can be computed

Algorithm 1: Computation of the α-structure

Input: Point set P, parameter α
Output: α-structure $\mathcal{S}_\alpha(P)$
foreach $p \in P$ **do**
  CPN-Search: Find all points $T_p \subseteq P$ in the CPN of p
  CPN-Check: Check for each edge pq with $q \in T_p$ whether it fulfils Condition (2) of Lemma 2.2, possibly compute the time attributes and add to $\mathcal{S}_\alpha(P)$

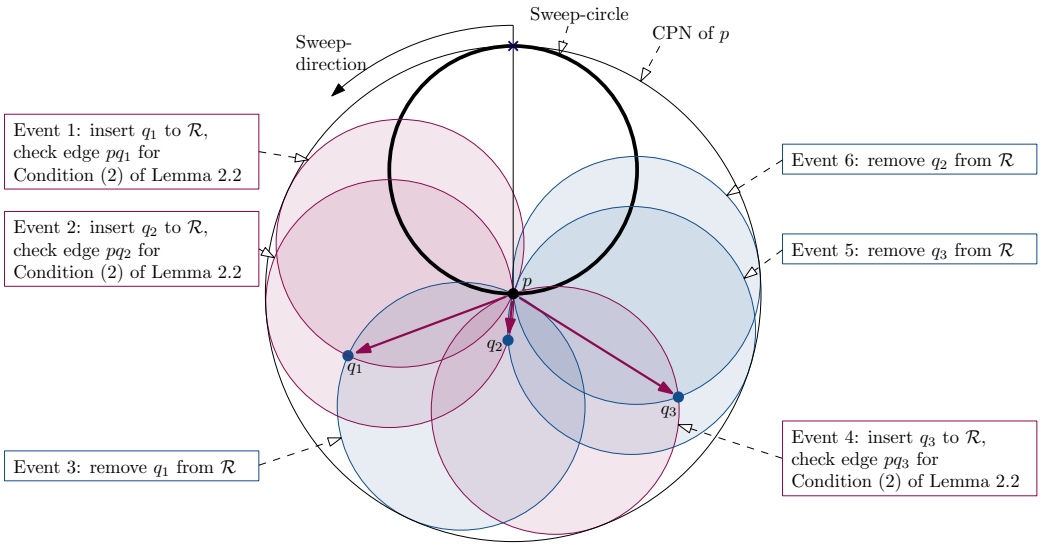


Figure 5 The rotational sweep CPN-Check method for point p and CPN points $T_p = \{q_1, q_2, q_3\}$.

using the temporal order of $\mathcal{R}$ in $O(\log m_{\mathrm{R}})$ time. This rotational sweep can be done in $O(m_{\mathrm{R}} \log m_{\mathrm{R}})$ time. Overall Algorithm 1 has running time $O(n(\log n + m_{\mathrm{R}} \log m_{\mathrm{R}}))$.

▶ **Theorem 4.1.** *For a set P of n points the α-structure can be computed in $O(n(\log n + m_{\mathrm{R}} \log m_{\mathrm{R}}))$ time, where m_{R} is the maximum number of points in a square with width 2α.*

For the query phase we represent each edge e with time attributes t_e^1, t_e^2, t_e^3 and t_e^4 of the α-structure by a rectangle $[t_e^1, t_e^2] \times [t_e^3, t_e^4]$. A query $[t_Q^{\mathrm{start}}, t_Q^{\mathrm{end}}]$ corresponds to finding all rectangles containing the point $(t_Q^{\mathrm{start}}, t_Q^{\mathrm{end}})$. Using a data structure for filtering search, we can solve this problem in $O(\log n + k)$ time per query, where k is the size of the α-shape [4].

5 Experimental Evaluation

We analyze the performance of α-structures using a data set of storm events in the United States in the years 1991–2000 obtained from Data.gov; see Figure 6 for the year 1991. The experiments[1] indicate that the memory consumption is linear in n; see Figure 7. The construction time for a point set of size $n = 70\,000$ varies between several seconds and hours depending on the value of α; see Figure 7. We assume this to be acceptable, since it is a pre-processing step. Applying the α-structure for temporal range queries the experiments indicate the query time to be nearly constant 200 [ms]; see Figure 8. In contrast an implemented straight forward approach yields results that indicate a dependency to the subset size.

6 Conclusion and Outlook

Overall we presented the design and construction of a data structure that provides the edges of α-shapes for temporal range queries on point sets. For future work we plan to consider other aggregated representations of geographic objects. Further, to reduce memory

[1] Implementation in Java, performed on a 4-core Intel Core i7-7700T CPU with 16 GiB RAM.

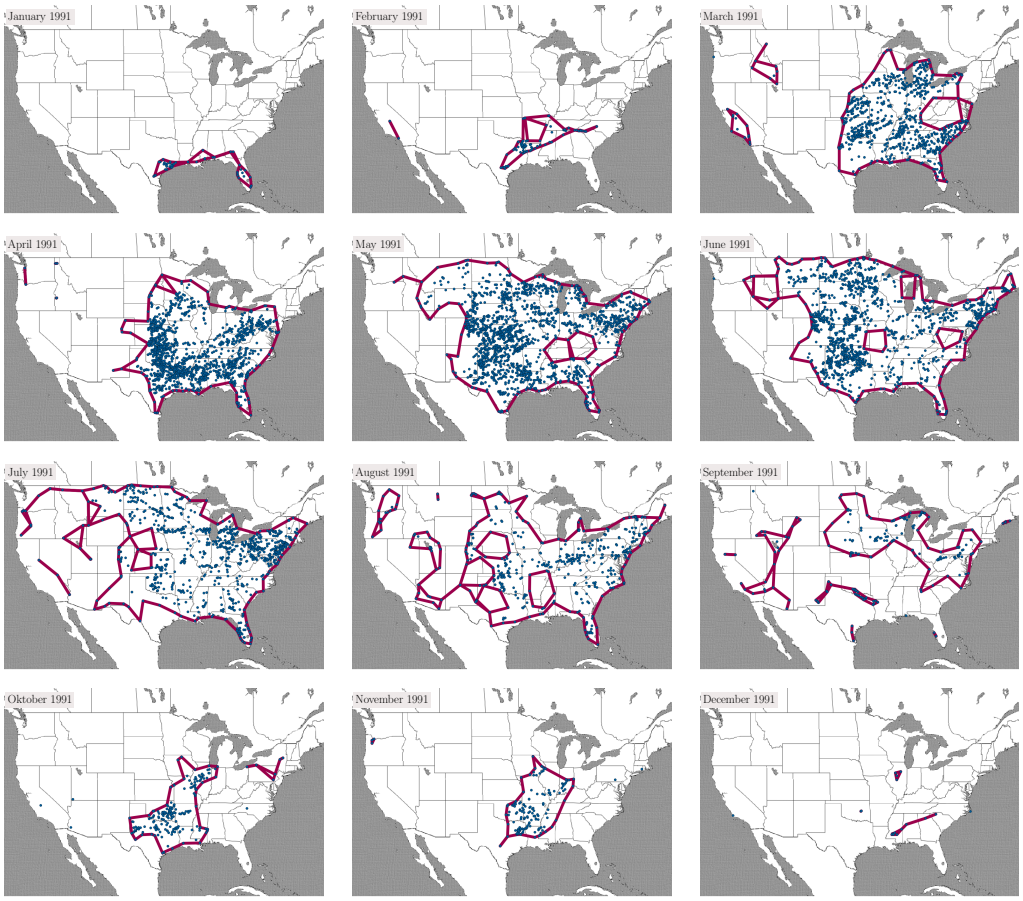


Figure 6 Storm events for the months of 1991 represented by α-shapes (lilac). The actual point set (blue) is drawn for illustration. Data retrieved from Data.gov. Map tiles by Stamen Design, under CC BY 3.0. Data by OpenStreetMap, under ODbL.

consumption we plan to work on an extension that only considers temporally long-lasting α-shape edges.

References

1 Dominique Attali. r-regular shape reconstruction from unorganized points. In *Proceedings of the thirteenth annual symposium on Computational geometry*, pages 248–253. ACM, 1997.

2 Fausto Bernardini and Chandrajit L. Bajaj. Sampling and reconstructing manifolds using alpha-shapes. Technical Report 97-013, Purdue University, Department of Computer Science, 1997.

3 A. Ray Chaudhuri, Bidyut Baran Chaudhuri, and Swapan K. Parui. A novel approach to computation of the shape of a dot pattern and extraction of its perceptual border. *Computer Vision and Image Understanding*, 68(3):257–275, 1997.

4 Bernard Chazelle. Filtering search: A new approach to query-answering. *SIAM Journal on Computing*, 15(3):703–724, 1986.

5 M.T. de Berg, W. Meulemans, and B. Speckmann. Delineating imprecise regions via shortest-path graphs. In *19th ACM SIGSPATIAL International Symposium on Advances*

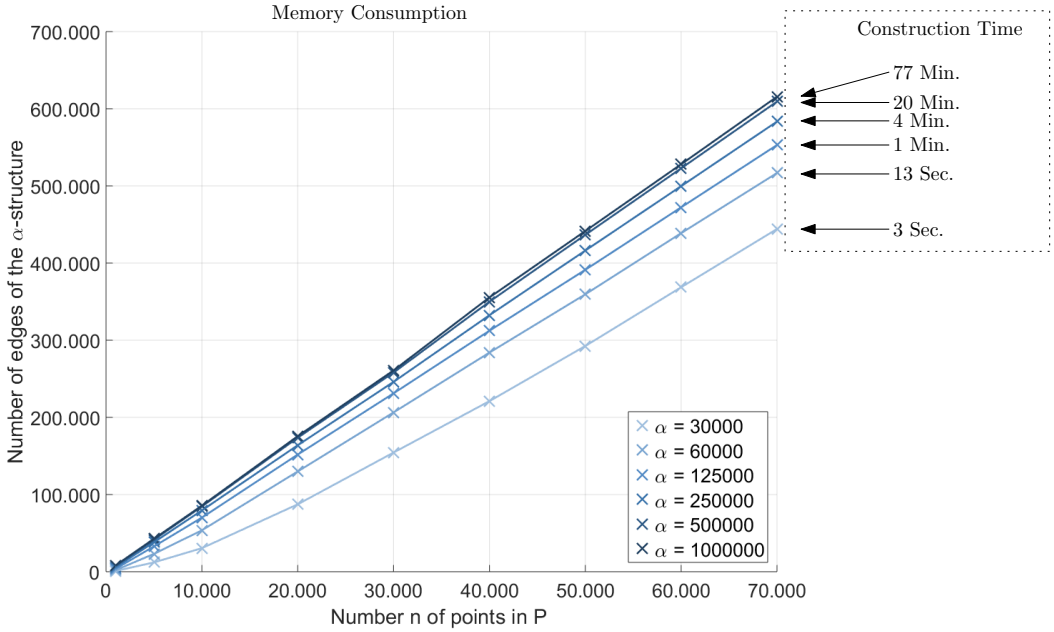


Figure 7 Memory consumption and construction time of an α-structure.

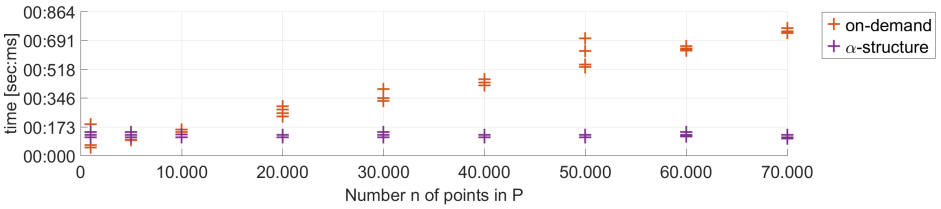


Figure 8 Query phase time of the α-structure compared to a straight forward implementation.

in Geographic Information Systems (ACM GIS), pages 271–280, United States, 2011. Association for Computing Machinery, Inc. doi:10.1145/2093973.2094010.

6 Matt Duckham, Lars Kulik, Mike Worboys, and Antony Galton. Efficient generation of simple polygons for characterizing the shape of a set of points in the plane. *Pattern recognition*, 41(10):3224–3236, 2008.

7 Herbert Edelsbrunner. Weighted alpha shapes. Technical Report UIUCDCS-R-92-1760, University of Illinois at Urbana-Champaign, Department of Computer Science, 1992.

8 Herbert Edelsbrunner. Alpha shapes - a survey. *Tessellations in the Sciences*, 27:1–25, 2010.

9 Herbert Edelsbrunner, David Kirkpatrick, and Raimund Seidel. On the shape of a set of points in the plane. *IEEE Transactions on information theory*, 29(4):551–559, 1983.

10 Christopher B. Jones, Ross S. Purves, Paul D. Clough, and Hideo Joho. Modelling vague places with knowledge from the web. *International Journal of Geographical Information Science*, 22(10):1045–1065, 2008.

11 Jie Liang, Herbert Edelsbrunner, Ping Fu, Pamidighantam Sudhakar, and Shankar Subramaniam. Analytical shape computation of macromolecules: I. molecular area and volume through alpha shape. *Proteins: Structure, Function, and Bioinformatics*, 33(1):1–17, 1998.

12 Dongliang Peng and Alexander Wolff. Watch your data structures. In *Proc. 22th Annu. Geograph. Inform. Sci. Research Conf. UK (GISRUK'14)*, Glasgow, 2014.

13 Ross Purves, Paul Clough, and Hideo Joho. Identifying imprecise regions for geographic information retrieval using the web. In *Proceedings of the 13th Annual GIS Research UK Conference*, pages 313–18, 2005.

14 Jari Vauhkonen, Ilkka Korpela, Matti Maltamo, and Timo Tokola. Imputation of single-tree attributes using airborne laser scanning-based height, intensity, and alpha shape metrics. *Remote Sensing of Environment*, 114(6):1263–1276, 2010.

15 Jari Vauhkonen, Timo Tokola, Petteri Packalén, and Matti Maltamo. Identification of scandinavian commercial species of individual trees from airborne laser scanning data using alpha shape metrics. *Forest Science*, 55(1):37–47, 2009.

Simplicial Depth for Multiple Query Points

Luis Barba[1], Stefan Lochau[2], Alexander Pilz*[3], and Patrick Schnider[4]

1 **Department of Computer Science, ETH Zürich, Switzerland**
`luis.barba@inf.ethz.ch`
2 **Department of Mathematics, ETH Zürich, Switzerland**
`stefan.andrea.lochau@alumni.ethz.ch`
3 **Institute of Software Technology, Graz University of Technology, Austria**
`apilz@ist.tugraz.at`
4 **Department of Computer Science, ETH Zürich, Switzerland**
`patrick.schnider@inf.ethz.ch`

Abstract

We consider the following generalization of *simplicial depth*: for a set of n data points P and a set of k query points Q, the simplicial depth of Q with respect to P is the number of simplices spanned by P that contain at least one point of Q. We study this generalization for point sets in the plane. For two query points we give bounds on the maximal simplicial depth, as well as an $O(n \log(n))$ time algorithm to compute the simplicial depth. For a general number of query points we prove a bound on on the maximal simplicial depth if the data point set is in convex position. Finally, we give an $O((n+k)^{7/3}\text{polylog}(n+k))$ time algorithm to compute the simplicial depth of arbitrary query point sets with respect to arbitrary data point sets.

1 Introduction

Suppose we are given a set of n data points in $\mathbb{R}^d$, which we would like to represent with just a few points. For just one representative in $\mathbb{R}^1$, this could be the median. One way to view the median is as the "deepest" point in the set of data points. Given a set $P = \{p_1, \dots, p_n\} \subseteq \mathbb{R}^1$ of n reals, it is quite intuitive to formalize a notion of "depth" w.r.t. P (the data points): for a given $q \in \mathbb{R}$, we merely count how many points of P are on each side of q and take the minimum of these two numbers. Then finding the median equals finding a point of maximal depth. Defining higher-dimensional medians requires to generalize not only the median itself, but the entire notion of depth. Several such depth measures have been introduced over time, most famously Tukey depth [21] (also called halfspace depth), simplicial depth [18], or convex hull peeling depth [5]; see, e.g., the survey by Aloupis [3]. Here, we consider simplicial depth:

▶ **Definition 1.1** (Simplicial depth)**.** For a finite point set $P \subset \mathbb{R}^d$ and a query point q, the *simplicial depth* $\sigma_P(q)$ is the number of open simplices with $d+1$ vertices in P that contain q.

The definition is attributed to Liu [18][1]; however, special cases were also addressed prior to her article (e.g., already in 1955 by Kárteszi [14]). In $\mathbb{R}^2$, Boros and Füredi [7] showed that for any set P of size n in general position there exists a point q with $\sigma_P(q) \geq n^3/27 + O(n^2)$, and there are sets where every point has simplicial depth of at most $n^3/27 + n^2$. Clearly, for n points in general position, not all triangles can hit a single point. The simplicial depth cannot be more than $\binom{\lfloor \frac{n+2}{2} \rfloor}{3} + \binom{\lceil \frac{n+2}{2} \rceil}{3} = \frac{n^3}{24} - \frac{n}{6}$, and there are sets that allow for

* Supported by a Schrödinger fellowship of the Austrian Science Fund (FWF): J-3847-N35.
[1] While in [18] the simplices are closed, we follow, e.g., [7, 10, 13, 16] and consider them open. Still, this will not make a significant difference herein, as we usually require the sets to be in general position.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 19–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

such a depth [7]. From an algorithmic point of view, Gil, Steiger, and Wigderson [13] and, independently, Khuller and Mitchell [17] showed that in $\mathbb{R}^2$ the simplicial depth of a query point can be computed in $O(n\log(n))$ time and that all simplicial depths of the points in P can be found in $O(n^2)$ time. For the simplicial depth of a query point in $\mathbb{R}^3$ and $\mathbb{R}^4$, Cheng and Ouyang [9] found $O(n^2)$ and $O(n^4)$ time algorithms, respectively. Improving on a previous general $O(n^d\log(n))$ time bound by Afshani, Sheehy, and Stein [1], Pilz, Welzl and Wettstein [20] gave an $O(n^{d-1})$ time algorithm for all $d \geq 3$. The problem is #P-complete and W[1]-hard if the dimension is part of the input [1]. The best known algorithm for finding a point with maximal simplicial depth in a given set in $\mathbb{R}^2$ takes $O(n^4)$ time [4].

In this work, we consider the use of multiple query points $q_1, \ldots, q_k$, instead of just one. The idea is to find a higher-dimensional analogue to quantiles, which further describe samples in $\mathbb{R}^1$. So we extend the definition by replacing the query point q by any of the points $q_i,\ i \in \{1, \ldots, k\}$. That is, we count the simplices containing at least one of the query points.

▶ **Definition 1.2** (Simplicial depth of multiple query points)**.** Let $P \subset \mathbb{R}^d$ and $Q \subset \mathbb{R}^d$ be two finite point sets. Then the *simplicial depth* of Q with respect to P is $\sigma_P(Q)$, the number of open simplices with $d+1$ vertices in P that contain at least one point $q \in Q$. A simplex that contains at least one of the query points Q *hits* Q.

Indeed, the two query points that maximize the simplicial depth for a one-dimensional data set are the 1/3 and 2/3-quantiles, which accounts for this way of generalization. The idea of generalizing quantiles by generalizing depth measures to several query points has already been considered for the Tukey depth [19].

The problem of stabbing triangles spanned by a point set was studied by Katchalski and Meir [16], as well as Czyzowicz, Kranakis and Urrutia [10], who independently proved that, for an n-point set P with h extreme points, $2n-2-h$ many points are sufficient and necessary to stab every triangle spanned by P (i.e., to have simplicial depth of $\binom{n}{3}$).

2 Two query points

In this section we focus on finite point sets P and two query points q_1, q_2 in the plane. We assume that all points in $P \cup Q$ are in general position, i.e., no three points are collinear.

2.1 Computing the depth of two query points

We argue that computing the simplicial depth of two query points is also in $O(n\log n)$. W.l.o.g., assume that the query points lie on the x-axis and that q_1 has a smaller x-coordinate than q_2. Rather than computing the simplicial depth directly, we consider all $\binom{n}{3}$ simplices and subtract the number of those which do *not* hit the query points.

We partition $P = U \dot\cup L$, where U and L are the points above and below the x-axis, respectively. We thus have $\binom{|U|}{3}$ triangles above and $\binom{|L|}{3}$ below the x-axis. The triangles intersecting the x-axis have one vertex on one side and two on the other side. For each point $p \in L$, let $s_{\mathrm{R}}(p)$ be the number of points in U to the right of the line pq_2, let $s_{\mathrm{L}}(p)$ be the number of points in U the left of pq_1, and let $s_{\mathrm{B}}(p) = |U| - s_{\mathrm{L}}(p) - s_{\mathrm{R}}(p)$. For a point $p \in U$ the functions are defined analogously with the roles of L and U swapped. Thus, we get

$$\sigma_P(q_1,q_2) = \binom{|P|}{3} - \binom{|U|}{3} - \binom{|L|}{3} - \sum_{p\in P}\left[\binom{s_{\mathrm{R}}(p)}{2} + \binom{s_{\mathrm{B}}(p)}{2} + \binom{s_{\mathrm{L}}(p)}{2}\right] . \qquad (1)$$

It remains to compute the values of s_{L} and s_{R} efficiently. To this end, we first sort the points radially around q_1 and q_2 in $O(n\log n)$ time. Then, for each point $p \in U$, we can

count the points of L to the left of q_1p in overall $O(n)$ time, by considering the points in clockwise order around q_1 and maintaining this number (i.e., starting with $|L|$, decreasing the number when reaching a point of L, and storing it when reaching a point of U). We thus obtain a table for s_{R} for all points in U, and do the analogous for the remaining values.

2.2 Bounds for two query points

We provide an upper bound for the simplicial depth of two query points. Let $m_1 = |L|$ and $m_2 = |U|$, i.e., $m_1 + m_2 = n$. First, consider a fixed point $l \in L$ and all triangles it forms with two points of U. For such a triangle lu_1u_2, we define C as the closed cone formed by the rays $\overrightarrow{lu_1}$ and $\overrightarrow{lu_2}$, and call $d := |C \cap U| - 1$ the *span* of the triangle. Then d is one plus the number of points strictly between u_1 and u_2 in the radial ordering around l. We now count how many of these triangles with a certain span d hit $\{q_1, q_2\}$.

We note that, for a fixed l, at most d triangles with span d can hit a query point, so at most $2d$ of these can be hitting. On the other hand, for each d there are at most $m_2 - d$ many triangles, hitting or not. We can do this for every lower and upper point, and sum up to get an upper bound on the simplicial depth as follows:

$$\sigma_P(q_1, q_2) \leq \sum_{l \in L} \sum_{d=1}^{m_2-1} \min\{2d, m_2 - d\} + \sum_{u \in U} \sum_{d=1}^{m_1-1} \min\{2d, m_1 - d\}$$

This implies a simpler bound of $\sigma_P(q_1, q_2) \leq \frac{1}{3} m_1 m_2 (m_1 + m_2 + 4)$, which is maximized for $m_1 = m_2 = n/2$ (see the full version), where we get the following:

▶ **Theorem 2.1.** *Let $P \subseteq \mathbb{R}^2$ be a set of n data points and q_1, q_2 be two query points, all in general position. Then*

$$\sigma_P(q_1, q_2) \leq n^3/12 + n^2/3.$$

There are indeed point sets in convex position that allow for a similar simplicial depth: consider P as the vertex set of a regular n-gon for $n = 2m$ (see Figure 1 for an illustration). Place both q_1 and q_2 on the intersection c of the long diagonals and move them slightly outwards such that they do not lie on any line through two points of P but such that the line through them contains c. Then $\{q_1, q_2\}$ has simplicial depth $\sigma_P(q_1, q_2) = \frac{m^3}{3} + m^2 - \frac{4m}{3}$. With $n = 2m$, this translates to $\frac{n^3}{24} + \frac{n^2}{4} - \frac{2n}{3}$, which is roughly half of the bound in Theorem 2.1. Further, recall that the maximal simplicial depth of a single point is $\frac{n^3}{24} - \frac{n}{6}$. Comparing to this, we only improve by addition of a quadratic term. Nevertheless, for up to 14 data points in convex position, computational experiments have shown that this construction is optimal. We conjecture that this holds for all sets in convex position. Note that while for one query point it is not hard to see that the simplicial depth is maximized for data point sets in convex position, the same cannot be said for two query points. In fact, the same question can be asked for any number of query points: is the simplicial depth of k query points always maximized by data point sets in convex position?

3 More query points

3.1 Upper bound for data points in convex position

In this section we give an upper bound to the simplicial depth of any set of k query points $Q = \{q_1, \ldots, q_k\}$ in a given point set $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$ when P is in convex position. We assume that $P \cup Q$ is in general position and Q is in $\mathrm{conv}(P)$, the convex hull of P.

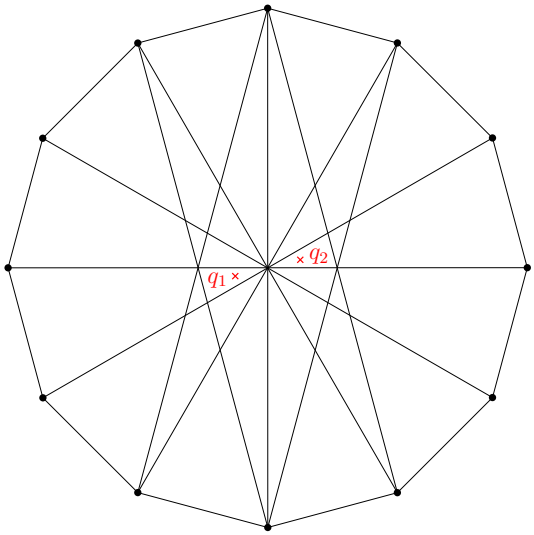


Figure 1 A point set of size $2m = 12$ with conjectured maximal simplicial depth 100 for two query points.

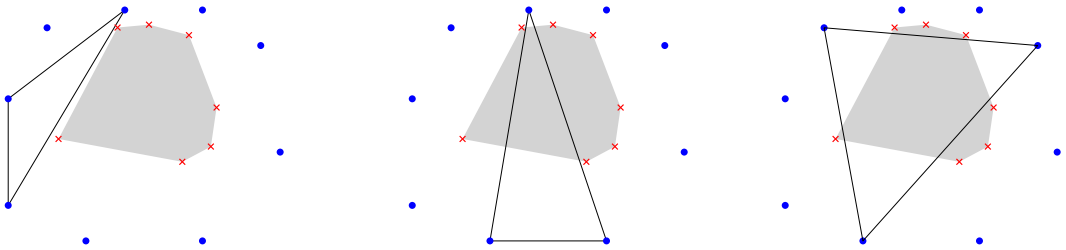


Figure 2 The different types of non-hitting triangles.

We again count the triangles not containing any points of Q. Let $\mathcal{S}$ be the set of triangles spanned by points of P. We partition $\mathcal{S}$ into those triangles that do hit Q, $\Sigma := \{S \in \mathcal{S} : S \cap Q \neq \emptyset\}$, and those which do not hit Q, $\Delta := \{S \in \mathcal{S} : S \cap Q = \emptyset\}$. Then $\sigma_P(Q) := |\Sigma| = \binom{n}{3} - |\Delta|$. We devise a lower bound on $|\Delta|$. Let $S \in \Delta$ be an arbitrary non-hitting triangle. Note that $S \setminus \text{conv}(Q)$ has at most three connected components, each of which contains at least one original vertex of S. We partition Δ into the triangles that get split into i parts by $\text{conv}(Q)$, Δ_i for $i \in \{1, 2, 3\}$. We see that (see Figure 2):

(1) If $S \setminus \text{conv}(Q)$ has only one component, then S and $\text{conv}(Q)$ are disjoint (as S does not hit Q). S has a unique pair of vertices w, w' whose supporting line separates S and Q.

(2) If $S \setminus \text{conv}(Q)$ has two components, we see that there are two vertices of S in one and a single vertex p in the other component. This vertex p is again unique.

(3) If $S \setminus \text{conv}(Q)$ has three components, we discard it and potentially worsen our bound.

Let $t_i(p)$ be the number of data points on the right side of the ray $\overrightarrow{pq_i}$, where the elements of Q are indexed according to their radial order around p. We get that:

$$\sigma_P(q_1,\dots,q_k) \le \binom{n}{3} - \sum_{p\in P}\left(\underbrace{\frac{1}{2}\binom{t_1(p)}{2} + \sum_{i=1}^{k-1}\binom{t_{i+1}(p)-t_i(p)}{2} + \frac{1}{2}\binom{n-1-t_k(p)}{2}}_{=:f_p(t_1,\dots,t_k)}\right).$$

We count each simplex that does not intersect $\mathrm{conv}(Q)$ "a half times" for w and w'. The function $f_p(t_1,\dots,f_k)$ is convex on $\{t\in\mathbb{R}^k \mid 0\le t_1\le\cdots\le t_k\le n-1\}$, so we can bound it from below separately using convex optimization techniques. (Intuitively, the value of f_p is small if the number of points between two consecutive points of Q is roughly the same; we provide a formal reasoning in the full version.) With this, we obtain an upper bound of

$$\sigma_P(q_1,\dots,q_k) \le \frac{n^3k}{6(k+3)} - \frac{3n^2}{2(k+3)} - \frac{5n}{24}.$$

For $k=2$, we can compare this to Theorem 2.1 – we go from the older $\frac{1}{12}n^3+\frac{1}{6}n^2$ bound to $\frac{1}{15}n^3-\frac{3}{10}n^2-\frac{5}{24}n$ and improve asymptotically by a factor of $\frac{5}{4}$. (But recall that the new bound is for P in convex position only.) Comparing thisto $\binom{n}{3}$ we get the following theorem.

▶ **Theorem 3.1.** *Let $P\subseteq\mathbb{R}^2$ be n points in convex position, and let $q_1,\dots,q_k\in\mathbb{R}^2$ be k query points. Then at most a fraction of $1-\frac{3}{k+3}+O\left(\frac{1}{n}\right)$ of the simplices with vertices in P can contain any of the k query points.*

3.2 Algorithmic aspects

▶ **Theorem 3.2.** *The simplicial depth of a set $Q\subset\mathbb{R}^2$ w.r.t. a set $P\subset\mathbb{R}^2$, all in general position, with $N=|P|+|Q|$ can be computed in $O(N^{7/3}\mathrm{polylog}(N))$ time.*

Proof. We use an approach similar to one of computing the number of empty triangles in a point set (see [12]). For a point $p\in P$, we define a simple polygon R_p that contains every triangle spanned by p and two other points of P not hitting Q. Let B be a bounding box of P. Shoot a ray from every point $q\in Q$ in the opposite direction of p until hitting B and add two edges for R_p in B starting at q with a small angle separated by the ray. See Figure 3. Now two points of $P\setminus\{p\}$ see each other in R_p iff they form a triangle with p not hitting Q.

Ben-Moshe et al. [6] construct the visibility graph of points inside a simple polygon. While enumerating the edges of this graph is too costly here, their method can be adapted to count them. They argue that edges of the visibility graph that cross an edge e separating the polygon (that is not necessarily a diagonal of the polygon) correspond to bichromatic crossings in an arrangement of red and blue segments: A point and the part on e which the point sees defines a (possibly empty) wedge, whose dual is a line segment; two points define an edge of the visibility graph crossing e iff their dual segments intersect; the red segments correspond to points in one sub-polygon defined by e, and the blue segments to points in the other sub-polygon. These segments can be given in $O(N\log(N))$ time using standard machinery for polygon visibility and point-line duality. Agarwal [2, Theorem 6.1] shows how to count bichromatic crossings of n red and blue segments in $O(n^{4/3}\mathrm{polylog}(n))$ time.

As in [6, Section 2.1], we divide R_p into two parts, each containing at least a constant fraction of the union of points and R_p's vertices (e.g., 1/3 is doable by adapting an approach from [8] not even using that R_p is star-shaped). Then, we recursively count the edges of the visibility graph in the two sub-polygons and add the number of edges crossing the diagonal using Agarwal's algorithm [2, Sect. 6]. As this is the dominating task in each iteration, we can use induction on the number of points to show that this algorithm requires $O(N^{4/3}\mathrm{polylog}(N))$ time for computing the number of triangles with a point p not hitting Q. ◀

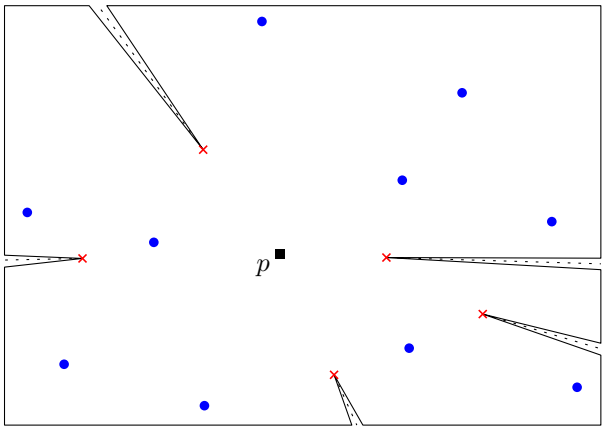


Figure 3 The polygon R_p of a point $p \in P$. Any two points of $P \setminus \{p\}$ (blue dots) that see each other correspond to a triangle with p not containing a point of Q (red crosses).

We do not know about the complexity of finding a set Q, $|Q| = k$, with maximal simplicial depth for a given integer k and data set P. Using a straight-forward reduction from monotone planar 3-SAT [11], we show in the full version that extending a given set to have a point in each triangle of P is NP-hard.

References

1 Peyman Afshani, Donald R. Sheehy, and Yannik Stein. Approximating the simplicial depth. *CoRR*, abs/1512.04856, 2015. URL: `http://arxiv.org/abs/1512.04856`.

2 Pankaj K. Agarwal. Parititoning arrangements of lines II: applications. *Discrete & Computational Geometry*, 5:533–573, 1990. `doi:10.1007/BF02187809`.

3 Greg Aloupis. Geometric measures of data depth. In Regina Y. Liu, Robert Serfling, and Diane L. Souvaine, editors, *Data Depth: Robust Multivariate Analysis, Computational Geometry and Applications*, pages 147–158. DIMACS/AMS, 2003.

4 Greg Aloupis, Stefan Langerman, Michael A. Soss, and Godfried T. Toussaint. Algorithms for bivariate medians and a Fermat-Torricelli problem for lines. *Comput. Geom.*, 26(1):69–79, 2003. `doi:10.1016/S0925-7721(02)00173-6`.

5 Vic Barnett. The ordering of multivariate data. *Journal of the Royal Statistical Society. Series A (General)*, 139(3):318–355, 1976. URL: `http://www.jstor.org/stable/2344839`.

6 Boaz Ben-Moshe, Olaf A. Hall-Holt, Matthew J. Katz, and Joseph S. B. Mitchell. Computing the visibility graph of points within a polygon. In Jack Snoeyink and Jean-Daniel Boissonnat, editors, *Proc. 20th ACM Symposium on Computational Geometry (SoCG 2004)*, pages 27–35. ACM, 2004. `doi:10.1145/997817.997825`.

7 E. Boros and Z. Füredi. The number of triangles covering the center of an n-set. *Geometriae Dedicata*, 17(1):69–77, Oct 1984. `doi:10.1007/BF00181519`.

8 Bernard Chazelle. A theorem on polygon cutting with applications. In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 339–349. IEEE Computer Society, 1982. `doi:10.1109/SFCS.1982.58`.

9 Andrew Cheng and Ming Ouyang. On algorithms for simplicial depth. In *Proc. 13th Canadian Conference of Computational Geometry (CCCG 2001)*, pages 53–56, 2001.

10 Jurek Czyzowicz, Evangelos Kranakis, and Jorge Urrutia. Guarding the convex subsets of a point-set. In *Proc. 12th Canadian Conference of Computational Geometry (CCCG 2000)*, pages 47–50, 2000.

11 Mark de Berg and Amirali Khosravi. Optimal binary space partitions for segments in the plane. *Int. J. Comput. Geometry Appl.*, 22(3):187–206, 2012.

12 David P. Dobkin, Herbert Edelsbrunner, and Mark H. Overmars. Searching for empty convex polygons. *Algorithmica*, 5(4):561–571, 1990. doi:10.1007/BF01840404.

13 Joseph Gil, William Steiger, and Avi Wigderson. Geometric medians. *Discrete Mathematics*, 108(1):37 – 51, 1992. doi:10.1016/0012-365X(92)90658-3.

14 Franz Kárteszi. Extremalaufgaben über endliche Punktsysteme. *Publ. Math. Debrecen*, 4:16–27, 1955.

15 William Karush. Minima of functions of several variables with inequalities as side constraint. *M.Sc. Dissertation. Dept. of Mathematics, Univ. of Chicago, Chicago, Illinois*, 1939. URL: https://dissexpress.proquest.com/dxweb/results.html?QryTxt=&By=&Title=&pubnum=TM11033.

16 Meir Katchalski and Amram Meir. On empty triangles determined by points in the plane. *Acta Mathematica Hungarica*, 51(3-4):323–328, 1988.

17 Samir Khuller and Joseph S.B. Mitchell. On a triangle counting problem. *Information Processing Letters*, 33(6):319 – 321, 1990. doi:10.1016/0020-0190(90)90217-L.

18 Regina Y. Liu. On a notion of data depth based on random simplices. *Ann. Statist.*, 18(1):405 – 414, 03 1990. doi:10.1214/aos/1176347507.

19 Alexander Pilz and Patrick Schnider. Extending the centerpoint theorem to multiple points. In Wen-Lian Hsu, Der-Tsai Lee, and Chung-Shou Liao, editors, *29th International Symposium on Algorithms and Computation (ISAAC 2018)*, volume 123 of *LIPIcs*, pages 53:1–53:13. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018. URL: http://www.dagstuhl.de/dagpub/978-3-95977-094-1, doi:10.4230/LIPIcs.ISAAC.2018.53.

20 Alexander Pilz, Emo Welzl, and Manuel Wettstein. From Crossing-Free Graphs on Wheel Sets to Embracing Simplices and Polytopes with Few Vertices. In Boris Aronov and Matthew J. Katz, editors, *33rd International Symposium on Computational Geometry (SoCG 2017)*, volume 77 of *LIPIcs*, pages 54:1–54:16. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.SoCG.2017.54.

21 John W. Tukey. Mathematics and picturing data. *Proc. Int. Congr. Mathematics, Vancouver*, 2:523 – 531, 1975.

Encoding 3SUM

Sergio Cabello[1], Jean Cardinal[2], John Iacono[2,3], Stefan Langerman[2], Pat Morin[4], and Aurélien Ooms[2]

1 **University of Ljubljana**
2 **Université libre de Bruxelles**
3 **New York University**
4 **Carleton University**

Abstract

We consider the following problem: given three sets of real numbers, output a word-RAM data structure from which we can efficiently recover the sign of the sum of any triple of numbers, one in each set. This is similar to a previous work by some of the authors to encode the order type of a finite set of points. While this previous work showed that it was possible to achieve slightly subquadratic space and logarithmic query time, we show here that for the simpler 3SUM problem, one can achieve an encoding that takes $\tilde{O}(N^{\frac{3}{2}})$ space for inputs sets of size N and allows constant time queries in the word-RAM.

1 The Problem

Given three sets of N real numbers $A = \{ a_1 < a_2 < \cdots < a_N \}$, $B = \{ b_1 < b_2 < \cdots < b_N \}$, and $C = \{ c_1 < c_2 < \cdots < c_N \}$, we wish to build a discrete data structure (using bits, words, and pointers) such that, given any triple $(i, j, k) \in [N]^3$ it is possible to compute the sign of $a_i + b_j + c_k$ by only inspecting the data structure (we cannot consult A, B, or C). We refer to the map $\chi : [N]^3 \to \{-, 0, +\}, (i, j, k) \mapsto \operatorname{sgn}(a_i + b_i + c_k)$ as the *3SUM-type* of the instance $\langle A, B, C \rangle$. Obviously, one can simply construct a lookup table of size $O(N^3)$, such that triple queries can be answered in $O(1)$ time. We aim at improving on this trivial solution.

2 Motivation

In the 3SUM problem, we are given an array of numbers as input and are asked whether any three of them sum to 0. In the mid-nineties, this problem was identified as a bottleneck of many important problems in geometry, such as detection of affine degeneracies or motion planning [5]. Since then, it has become a central problem in fine-grained complexity theory [9]. It has long been conjectured to require $\Omega(N^2)$ time. In 2014, it was shown to be solvable in $o(N^2)$ time, but no algorithm with running time $O(N^{2-\delta})$ with constant $\delta > 0$ is known [7].

Lower bounds exist in restricted models of computation. Most notably, $\Omega(N^2)$ 3-linear queries are needed to solve 3SUM [4], and nontrivial lower bounds have also been proven for slightly more powerful linear decision trees [1]. However, in a recent breakthrough contribution, Kane, Lovett, and Moran showed that 3SUM could be solved using $O(N \log^2 N)$ 6-linear queries [8], hence within a $O(\log N)$ factor of the information-theoretic lower bound.

Linear decision trees are examples of *nonuniform algorithms*, in which we are allowed to have different algorithms for different input sizes. Algebraic decision trees generalize linear decision trees by allowing decision based on the sign of constant-degree polynomials at each node [10].

Any decision tree identifying the 3SUM-type of a 3SUM instance yields a concise encoding of this 3SUM-type: just write down the outcome of the successive tests. Knowing the decision tree by convention, this sequence of bits is sufficient to recover the sign of any triple.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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 Table of results

	Query time	Space (in bits)	Preprocessing time
Trivial	$O(1)$	$O(N^3)$	$O(N^3)$
Almost trivial	$O(1)$	$O(N^2 \log N)$	$O(N^2)$
Order-type encoding [2]	$O(\log N)$	$O(\frac{N^2 \log^2 \log N}{\log N})$	$O(N^2)$
Order-type encoding [2]	$O(\frac{\log N}{\log \log N})$	$O(\frac{N^2}{\log^{1-\epsilon} N})$	$O(N^2)$
Numeric representation (§4)	$O(N)$	$O(N^2)$	$N^{O(1)}$
Space-optimal representation (§5)	$N^{O(1)}$	$O(N \log N)$	$N^{O(1)}$
Query-optimal (§6)	$O(1)$	$\tilde{O}(N^{1.5})$	$O(N^2)$

The question we consider here is how to make such a representation efficient, in the sense that not only does it use merely a few bits, but the answer to any triple query can be recovered efficiently. Understanding the interplay between nonuniform algorithms and such data structures hopefully sheds light on the intrinsic structure of the problem.

3 Results

See table 1 for a summary. As there are only $O(N^3)$ queries, a table of size $(\log_2 3)N^3 + O(1)$ bits suffices to give constant query time [3]. This can be improved to $O(N^2 \log N)$ bits of space by storing for each pair (i, j) the values $k_<(i, j) = \max\{0\} \cup \{k\colon\ a_i + b_j + c_k < 0\}$ and $k_>(i, j) = \min\{N + 1\} \cup \{k\colon\ a_i + b_j + c_k > 0\}$. For a query (i, j, k), we compare k against the values $k_<(i, j)$ and $k_>(i, j)$ to recover $\chi(i, j, k)$ in $O(1)$ time. All $k_<(i, j)$ and $k_>(i, j)$ can be computed in $O(N^2)$ time via the classic quadratic time algorithm for 3SUM.

One seemingly simple representation is to store the numbers in A, B and C; however these are reals and thus we need to make them representable using a finite number of bits. In Section 4 we show that a minimal integer representation of a 3SUM instance may require $\Theta(N)$ bits per value, which would give rise to a $O(N)$ query time and $O(N^2)$ space, which is far from impressive. In [2] the problem of given a set of N lines, to create an encoding of them so that the orientation of any triple (the *order type*) can be determined was studied; our problem is a special case of this where the lines only have three slopes. Can we do better for the case of 3SUM? We answer this in the affirmative. In Section 5 we show how to use an optimal $O(N \log N)$ bits of space with a polynomial query time. Finally, in section 6 we show how to use $\tilde{O}(N^{1.5})$ space to achieve $O(1)$-time queries.

4 Representation by numbers

A first natural idea is to encode the real 3SUM instance by *rounding* its numbers to integers. We show a tight bound of $\Theta(N^2)$ bits for this representation.

▶ **Lemma 4.1.** *Every 3SUM instance has an equivalent integer instance where all values have absolute value at most $2^{O(N)}$. Furthermore, there exists an instance of 3SUM where all equivalent integer instances require numbers at least as large as the Nth Fibonacci number and where the standard binary representation of the instance requires $\Omega(N^2)$ bits.*

Proof. Every 3SUM instance $A = \{\, a_1 < a_2 < \ldots < a_N \,\}$, $B = \{\, b_1 < b_2 < \cdots < b_N \,\}$, and $C = \{\, c_1 < c_2 < \cdots < c_N \,\}$ can be interpreted as the point $(a_1, \ldots, a_N, b_1, \ldots, b_N, c_1, \ldots, c_N)$ in $\mathbb{R}^{3N}$. Let us use the variables $x_1, \ldots, x_N$ to encode the first N dimensions of $\mathbb{R}^{3N}$,

$y_1, \ldots, y_N$ to encode the next N dimensions, and $z_1, \ldots, z_N$ for the remaining dimensions. Consider the subset of $\mathbb{R}^{3N}$

$$\Delta = \{(x_1, \ldots, x_N, y_1, \ldots, y_N, z_1, \ldots, z_N) \mid x_i < x_{i+1},\ y_j < y_{j+1},\ z_k < z_{k+1}\ \forall i, j, k \in [N-1]\}$$

and the set Π of N^3 hyperplanes $x_i + y_j + z_k = 0$, where $i, j, k \in [N]$. Let $\mathcal{A}$ be the arrangement defined by Π *inside* Δ. Instances of 3SUM correspond to points in Δ. Moreoever, two 3SUM instances have the same 3SUM-type if and only if they are in the same cell of $\mathcal{A}$.

Consider an instance $\langle A, B, C\rangle$ and let $\sigma = \sigma(A, B, C)$ be the cell of $\mathcal{A}$ that contains it. Then σ is the cell defined by the inequalities

$$\forall i, j, k \in [N]: \quad \begin{cases} x_i + y_j + z_k > 0 & \text{if } \chi(i,j,k) = +1, \\ x_i + y_j + z_k = 0 & \text{if } \chi(i,j,k) = 0, \\ x_i + y_j + z_k < 0 & \text{if } \chi(i,j,k) = -1. \end{cases}$$

$$\forall i, j, k \in [N-1]: \quad \begin{cases} x_i - x_{i+1} < 0, \\ y_j - y_{j+1} < 0, \\ z_k - z_{k+1} < 0. \end{cases}$$

Let σ' be the subset of $\mathbb{R}^{3N}$ defined by the following inequalities:

$$\forall i, j, k \in [N]: \quad \begin{cases} x_i + y_j + z_k \geq 1 & \text{if } \chi(i,j,k) = +1, \\ x_i + y_j + z_k = 0 & \text{if } \chi(i,j,k) = 0, \\ x_i + y_j + z_k \leq -1 & \text{if } \chi(i,j,k) = -1. \end{cases}$$

$$\forall i, j, k \in [N-1]: \quad \begin{cases} x_i - x_{i+1} \leq 1, \\ y_j - y_{j+1} \leq 1, \\ z_k - z_{k+1} \leq 1. \end{cases}$$

Clearly σ' is contained in σ. Moreover, for a sufficiently large $\lambda > 0$ the scaled instance $\langle \lambda A, \lambda B, \lambda C\rangle$ belongs to σ'. Therefore, σ' is nonempty.

Since σ' is defined by a collection of linear inequalities defining closed halfspaces, there exists a point p in σ' defined by a subset of at most $3N$ inequalities, where the inequalities are actually equalities. Let us assume for simplicity that exactly $3N$ equalities define the point p. Then, $p = (x, y, z)$ is the solution to a linear system of equations $M[x\ y\ z]^T = \delta$ where M and δ have their entries in $\{-1, 0, 1\}$ and each row of M has at most three non-zero entries. The solution p to this system of equations is an instance equivalent to $\langle \lambda A, \lambda B, \lambda C\rangle$.

Because of Cramer's rule, the system of linear equations has solution with entries $\det(M_i)/\det(M)$, where M_i is the matrix obtained by replacing the ith column of M by δ. We use the following simple bound on the determinant. Since $\det(M) = \sum_\pi \operatorname{sgn}(\pi) \prod_i m_{i,\pi(i)}$, where π iterates over the permutations of $[3N]$, there are at most 3^{3N} summands where π gives non-zero product $\prod_i m_{i,\pi(i)}$ (we have to select one non-zero entry per row), and the product is always in $\{-1, 0, 1\}$. Therefore $|\det(M)| \leq 3^{3N}$. Similarly, $|\det(M_i)| \leq 4^{3N}$ because each row of M_i has at most 4 non-zero entries. We conclude that the solution to the system $M[x\ y\ z]^T = \delta$ are rationals that can be expressed with $O(N)$ bits. This solution gives a 3SUM instance with rationals that is equivalent to $\langle A, B, C\rangle$. Since all the rationals have the common denominator $(\det(M))$, we can scale the result by $\det(M)$ and we get an equivalent instance with integers, where each integer has $O(N)$ bits.

The proof of the second statement is by implementing the Fibonacci recurrence in each of the arrays A, B, C. This can be achieved by letting:

$$\begin{aligned} a_i + b_1 + c_{N-i+1} &= 0, \text{for } i \in [N] \\ a_1 + b_i + c_{N-i+1} &= 0, \text{for } i \in [N] \\ a_{i-1} + b_{i-2} + c_{N-i+1} &< 0, \text{for } i \in \{3, 4, \ldots, N\}, \end{aligned}$$

The first two sets of equations ensure that the two arrays A and B are identical, while the array C contains the corresponding negated numbers, in reverse order. From the inequalities in the third group, and depending on the choice of the initial values a_1, a_2, each array contains a sequence growing at least as fast as the Fibonacci sequence. ◀

Note that this is a much smaller lower bound than for order types of points sets in the plane, the explicit representation of which can be shown to require exponentially many bits per coordinate [6].

5 Space-optimal representation

By considering the arrangement of hyperplanes defining the 3SUM problem, we get an information-theoretic lower bound on the number of bits in a 3SUM-type.

▶ **Lemma 5.1.** *There are* $2^{\Theta(N \log N)}$ *distinct 3SUM-types of size* N*.*

Proof. 3SUM-types of size N are in one-to-one correspondence with cells of the arrangement of N^3 hyperplanes in $\mathbb{R}^{3N}$. The number of such cells is $O(N^{9N})$ and at least $(N!)^2$. ◀

In order to reach this lower bound, we can simply encode the label of the cell of the arrangement in $\Theta(N \log N)$ bits. However, decoding the information requires to construct the whole arrangement which takes $N^{O(N)}$ time. An alternative solution is to store a vertex of the arrangement of hyperplanes $a_i + b_j + c_k \in \{-1, 0, 1\}$. There exists such a vertex that has the same 3SUM-type as the input point, as shown in the proof of Lemma 4.1. To answer any query, either recompute the vertex from the basis then answer the query using arithmetic, or use linear programming. Hence we can build a data structure of $O(N \log N)$ bits such that triple queries can be answered in polynomial time.

Note that we do not exploit much of the 3SUM structure here. In particular, the same essentially holds for k-SUM, and can also be generalized to a Subset Sum data structure of $O(N^2)$ bits, from which we can extract the sign of the sum of any subset of numbers.

6 Subquadratic space and constant query time

Our encoding is inspired by Grønlund and Pettie's $\tilde{O}(N^{1.5})$ non-uniform algorithm for 3SUM [7]. Our data structure stores three components, which we call the *differences*, the *staircase* and the *square neighbors*.

Differences. Partition A and B into *blocks* of $\sqrt{N}$ consecutive elements. Let D be the set of all differences of the form $a_i - a_j$ and $b_k - b_\ell$ where the items come from the same block. There are $O(N^{1.5})$ such differences. Sort D and store a table indicating for each difference in D its rank among all differences in D. This takes $O(\log N)$ bits for each of the $O(N^{1.5})$ differences, for a total of $O(N^{1.5} \log N)$ bits.

	1	2	10	14	17	22	32	33	40	91	92	97	98	110	120	127
1	2	3	11	15	18	23	33	34	41	92	93	98	99	111	121	128
11	12	13	21	25	28	33	43	44	51	102	103	108	109	121	131	138
13	14	15	23	27	30	35	45	46	53	104	105	110	111	123	133	140
19	20	21	29	33	36	41	51	52	59	110	111	116	117	129	139	146
24	25	26	34	38	41	46	56	57	64	115	116	121	122	134	144	151
34	35	36	44	48	51	56	66	67	74	125	126	131	132	144	154	161
51	52	53	61	65	68	73	83	84	91	142	143	148	149	161	171	178
57	58	59	67	71	74	79	89	90	97	148	149	154	155	167	177	184
59	60	61	69	73	76	81	91	92	99	150	151	156	157	169	179	186
114	115	116	124	128	131	136	146	147	154	205	206	211	212	224	234	241
119	120	121	129	133	136	141	151	152	159	210	211	216	217	229	239	246
127	128	129	137	141	144	149	159	160	167	218	219	224	225	237	247	254
128	129	130	138	142	145	150	160	161	168	219	220	225	226	238	248	255
133	134	135	143	147	150	155	165	166	173	224	225	230	231	243	253	260
138	139	140	148	152	155	160	170	171	178	229	230	235	236	248	258	265
142	143	144	152	156	159	164	174	175	182	233	234	239	240	252	262	269

Figure 1 Illustration of the staircase and square neighbors of the constant query time encoding. Here the 16×16 table is partitioned into a 4×4 grid of squares of size 4×4. If $c_k = 100$, the grey illustrates the squares that form the staircase, containing values both larger and smaller than 100. Predecessors and successors within each staircase square are shown in red and blue.

Staircase. Look at the table G formed by all sums of the form $a_i + b_j$, which is monotonic in its rows and columns due to A and B being sorted and view it as being partitioned into a grid G' of size $\sqrt{N} \times \sqrt{N}$ where each *square* of the grid is also of size $\sqrt{N} \times \sqrt{N}$. For each element $c \in C$, for each $i \in [1, \sqrt{N}]$ we store the largest j such that some elements of the square $G'[i, j]$ are $< c$, denote this as $V[c, i]$. We also store, for each $c \in C$, for each $j \in [1, \sqrt{N}]$ the smallest i such that some elements of the square $G'[i, j]$ are $\geq c$, denote this as $H[c, j]$. We thus store, in V and H, $\sqrt{N}$ values of size $O(\log N)$ for each of the N elements of C, for a total space usage of $O(N^{1.5} \log N)$ bits. We call this the *staircase* as this implicitly classifies, for each $c \in C$, whether each square has elements larger than c, smaller than c, or some larger and some smaller; only $O(\sqrt{N})$ can be in the last case, which we refer to as the *staircase* of c.

Square neighbors. For each element $c \in C$, for each of the $O(\sqrt{N})$ squares on the staircase, we store the location of the predecessor and successor of c in the squares $G'[i, V[c, i]]$ and $G'[H[c, j], j]$, for $i, j \in [1, \sqrt{N}]$. This takes space $O(N^{1.5} \log N)$.

To execute a query (a_i, b_j, c_k), only a constant number of lookups in the tables stored are needed. If $j < \sqrt{N} \cdot H[k, i]$, then we know $a_i + b_j > c_k$. If $i > \sqrt{N} \cdot V[k, j]$, then we know $a_i + b_j < c_k$. If neither of these is true, then the square $G'[\lceil i/\sqrt{N} \rceil, \lceil j/\sqrt{N} \rceil]$ is on the staircase of c_i and thus using the square neighbors table we can determine the location of the predecessor and successor of c_k in this square; suppose they are at $G[s_i, s_j]$ and $G[p_i, p_j]$ and thus $G[s_i, s_j] \leq c_k \leq G[p_i, p_j]$. One need only determine how these two compare to $G[i, j] = a_i + b_j$ to answer the query. But this can be done using the differences as follows: to compare $G[s_i, s_j]$ to $G[i, j]$ this would be determining the sign of $(a_i + b_j) - (a_{s_i} + b_{s_j})$ which is equivalent to determining the result of comparing $a_i - a_{s_i}$ and $b_j - b_{s_j}$, which since both are in the same square, these differences are in D and the comparison can be obtained by examining their stored ranks. By doing this for the predecessor and successor we will determine the relationship between $a_i + b_j$ and c_k.

References

1. Nir Ailon and Bernard Chazelle. Lower bounds for linear degeneracy testing. *J. ACM*, 52(2):157–171, 2005.
2. Jean Cardinal, Timothy M. Chan, John Iacono, Stefan Langerman, and Aurélien Ooms. Subquadratic encodings for point configurations. In *Symposium on Computational Geometry*, volume 99 of *LIPIcs*, pages 20:1–20:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.
3. Yevgeniy Dodis, Mihai Patrascu, and Mikkel Thorup. Changing base without losing space. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 593–602, 2010.
4. Jeff Erickson. Lower bounds for linear satisfiability problems. *Chicago J. Theor. Comput. Sci.*, 1999.
5. Anka Gajentaan and Mark H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Comput. Geom.*, 5:165–185, 1995.
6. Jacob E. Goodman, Richard Pollack, and Bernd Sturmfels. Coordinate representation of order types requires exponential storage. In *STOC*, pages 405–410. ACM, 1989.
7. Allan Grønlund and Seth Pettie. Threesomes, degenerates, and love triangles. *J. ACM*, 65(4):22:1–22:25, 2018.
8. Daniel M. Kane, Shachar Lovett, and Shay Moran. Near-optimal linear decision trees for k-sum and related problems. In *STOC*, pages 554–563. ACM, 2018.
9. Mihai Patrascu and Ryan Williams. On the possibility of faster SAT algorithms. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 1065–1075, 2010.
10. J. Michael Steele and Andrew Yao. Lower bounds for algebraic decision trees. *J. Algorithms*, 3(1):1–8, 1982.

3D Staged Tile Self-Assembly

Arne Schmidt[1]

1 **Department of Computer Science, TU Braunschweig, Germany.**
aschmidt@ibr.cs.tu-bs.de

Abstract

In the staged self-assembly model, building blocks, called tiles, are added to bins in stages. This results in multiple subassemblies attaching to each other at each stage. Previous work by Demaine et al. show that any polyomino can be created within $O(\log^2 n)$ stages. We improve this to $O(\log n)$ stages and also show that for any three-dimensional monotone polycube there is a staged self-assembly system with $O(\log n)$ stages, $O(1)$ glue types and a scale factor of five constructing this shape.

1 Introduction

In 1998, Erik Winfree [11] introduced the abstract Tile Self-Assembly Model (aTAM) in which unrotatable building blocks (called Wang tiles [10]) with specific *glue types* on their sides successively attach to a given tile, called *seed*. Each glue type has a strength and a tile can attach to an existing assembly through a matching glue type if the sum of corresponding glue types is at least a threshold value τ, called *temperature*.

Ten years later, Demaine et al. [3] introduced the staged Tile Self-Assembly Model (sTAM). Here, not only tiles but whole subassemblies can stick to each other rather than only attaching to a seed. In the sTAM, the assembly process is split up into several phases, called *stages*. In each stage, multiple subassemblies are created independently in various *bins*. The results can then be mixed together in a next stage and unwanted assemblies can be filtered out. Two subassemblies attach to each other if the sum of strengths of all matching glue types along the common boundary exceeds the temperature.

Demaine et al. [4] showed that, in contrast to aTAM, where a line can only be built with $\Omega(N)$ glue types, any shape can be built within $O(\log^2 n)$ stages using $O(1)$ glue types and $O(k)$ bins in sTAM. Here, N is the number of tiles of the shape, n is the side length of a smallest enclosing square, and k is the number of corners of the shape.

1.1 Our Contribution

We prove that there are staged self-assembly systems using $O(\log n)$ stages, $O(1)$ glue types, an $O(1)$ scale factor, temperature $\tau = 1$, and full connectivity to assemble (i) any two-dimensional shape using $O(k)$ bins, and (ii) any monotone three-dimensional shape using $O(n)$ bins.

1.2 Related Work

In the past few years, many models have been developed on top of the aTAM. Demaine et al. [3] introduced the two-handed assembly model (2HAM) in which two partial assemblies can bind to each other; from this model they developed the sTAM. Padilla et al. [5] introduced a hierarchical system based on signal passing, i.e., glues are inactive until activated by a signal.

Stages are also considered in the literature: Reif [7] uses a step-wise model for parallel computing, Park et al. [6] assemble DNA lattices with a hierarchical assembly technique and Somei et al. [9] use a step-wise assembly of DNA tiles. However, non of these works consider complexity aspects. In a recent paper, Chen and Doty [2] study the effects of parallelism in

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

hierarchical systems. In contrast to these works, Chalk et al. [1] consider the optimal stage complexity for a fixed number of bins and glue types in the sTAM. They show that for b bins and t tile types $O(\frac{K(S)-t^2-bt}{b^2} + \frac{\log\log b}{\log t})$ stages are sufficient, where $K(S)$ is the Kolmogorov complexity of a shape S. However, their technique uses a large scale factor; we use a scale factor of only five.

2 Definitions

Due to space constraints we omit the formal definition of an sTAM, which can be found in [3].
Polyominoes. A *polyomino* P is a union of unit squares, called *pixels*, joined edge to edge (see Fig. 1). W.l.o.g., we assume that the pixels are centered at points from $\mathbb{Z}^2$. A pixel is a *boundary pixel* if at least one of the eight (axis parallel or diagonal) neighbor positions is not occupied by a pixel in P. A boundary pixel p is a *corner* pixel of P if there are no two neighbor pixels p' and p'' that are collinear with p and that are boundary pixels. The *corner set* of a polyomino is the set of all corner pixels along all boundaries.

We define k as the cardinality of the corner set, N as the number of pixels of P, and n the side length of a bounding square. We consider the following metrics:
Stage Complexity: The number of stages needed to assemble the shape.
Bin Complexity: The maximum number of different bins used at the same time.
Glue Complexity: The number of different glue types.
Scale Factor: A scale factor c replaces each tile of a polyomino by a $c \times c$ supertile.
Full Connectivity: A polyomino is called *fully connected* if and only if there is a matching glue type between each pair of adjacent tiles.

Polycubes and the corresponding metrics are defined analogously.

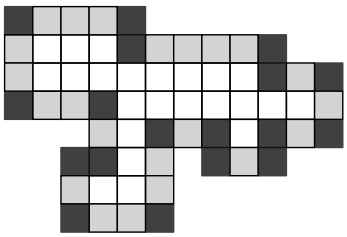


Figure 1 A polyomino with $N = 63$ tiles, its $k = 18$ corner pixels (dark gray), and its ordinary boundary pixels (light gray). A smallest square bounding the polyomino has side length $n = 12$.

3 Two Dimensions

We describe a staged tile self-assembly system that constructs an arbitrary two-dimensional shape within $O(\log n)$ stages using only $O(1)$ glue types and an appropriate scale factor. This is an improvement of a factor of $O(\log n)$ compared to Demaine et al. [4]. However, the construction is similar.

Consider a polyomino P^5, which is the result of scaling a polyomino P by five. We define the *backbone* similar to the backbone in [4]: First, we add all boundary pixels of P^5 to the backbone. Furthermore, we add *tunnels* starting at the leftmost bottommost boundary pixel p belonging to a hole. The tunnel extends two lines to the left; one at p and one two rows above p. We stop extending as soon as we reach another boundary pixel. Having a tunnel we remove

the boundary pixels enclosed by the tunnel (see Fig. 2 left). Note that special cases may occur when connecting a tunnel to the boundary (see Fig. 2 right). This construction gives a single cycle visiting all holes, which is easy to construct.

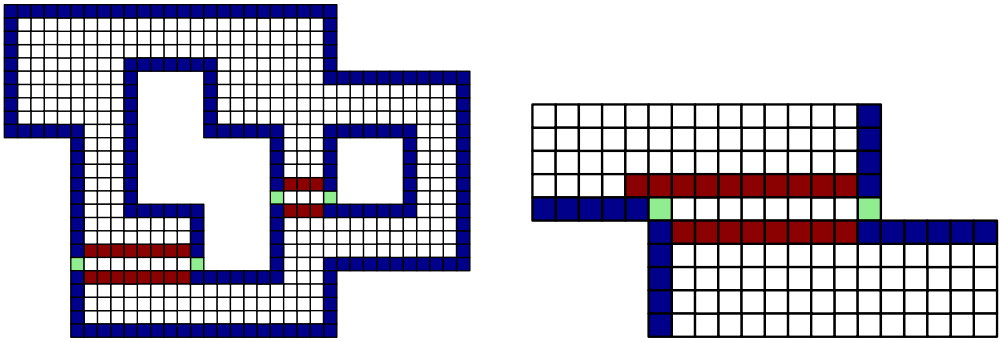


Figure 2 Left: Example of a backbone (blue and red). Blue pixels denote boundary pixels, red pixel represent tunnels. The green pixel get removed from the backbone. Right: A special case that may occur during construction of the backbone: The tunnel (red) reaches a corner. One side of the tunnel connects to the first pixel after the corner.

▶ **Theorem 1.** *The backbone B_P of a polyomino P is constructible within $O(\log n)$ stages using $O(k)$ bins and three glue types at temperature $\tau = 1$.*

Proof. Each tile is connected to exactly two other tiles. Let p_0 be any tile of B_P and p_1, p_2 its two neighbors. Also, let a be the glue type between p_1 and p_0, and b the glue type between p_2 and p_0. We remove p_0 from B_P, obtaining a path-like structure whose ends have glue types a and b. Now, decompose this path by cutting it at a corner tile along an edge such that the number of corner tiles in both substructures are the same. The glue type between the two tiles where we made the cut will be the third glue type c. Again, we have path-like structures left, thus we can repeat the procedure. At some point we have only straight line paths left. These can be assembled with three glue types and $O(1)$ bins within $O(\log n)$ stages at temperature $\tau = 1$ (see [3]). As there are at most $O(k)$ different lines, $O(k)$ bins will be sufficient. ◀

▶ **Theorem 2.** *A polyomino P can be constructed within $O(\log n)$ stages using $O(k)$ bins, $O(1)$ glue types with full connectivity, and a scale factor five at temperature $\tau = 1$.*

Proof. The construction of an arbitrary polyomino proceeds in three steps: (1) Construct the backbone, (2) fill up missing boundary tiles and finally, (3) fill up all other tiles.

While constructing the backbone B_P we can ensure that each side of a tile in B_P pointing to the inner side of the polyomino gets a glue type corresponding to the chart in Fig. 3. For example, if the east side of a boundary tile points to the inner side, the glue type is $\{0, 1\}$, $\{5, 6\}$, $\{10, 11\}$, $\{15, 16\}$ or $\{20, 21\}$ depending on the position within the scaled tile (see Fig. 3). If a boundary tile in B_P is adjacent to a missing boundary tile (green tile in Fig. 2), then we set the glue type as shown in Fig. 4, increasing the number of glue types by seven.

After constructing the backbone with all glues needed, we add the missing boundary tiles in a single stage. In another stage, we can now fill up the whole polyomino with the tiles shown in Fig. 3. Because any side facing to a hole or to the outer face has no glue, no tile can be placed to a position not belonging to the polyomino.

We conclude: For phase (2) and (3) we need two stages and $O(1)$ glue types. Thus, in total we have $O(\log n)$ stages, $O(1)$ glue types, scale factor five, $O(k)$ bins, and full connectivity at temperature $\tau = 1$. ◀

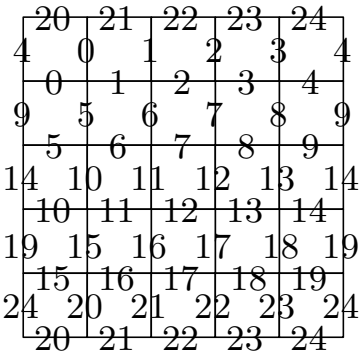


Figure 3 Glue types for tiles in a scaled tile; these are 25 different types.

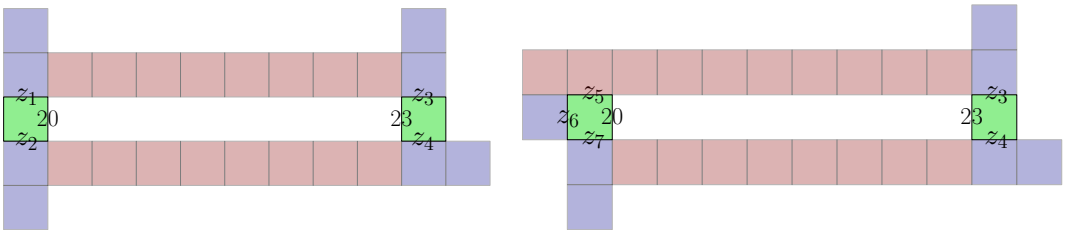


Figure 4 Glue types for tiles on the boundary missing in B_P.

4 Monotone, Three-Dimensional Shapes

Most assembly systems use the fact that the third dimension can be used to combine two partial assemblies. However, this cannot be easily done while constructing three-dimensional shapes, unless we can make use of a fourth dimension. Therefore, we need staged assembly systems that only combine partial three-dimensional assemblies for which we have an obstacle-free path. Previous work [8] considers assembly of polyominoes with straight free paths. In this paper we only consider a special case of three-dimensional shapes: z-monotone polycubes.

▶ **Definition 3.** A polycube P is z-monotone if the intersection of any x-y-plane with P results in a two-dimensional connected polyomino. We call such an intersection a *layer* of P. See Fig. 5 for an example.

The idea to construct three-dimensional monotone shapes is to construct *slices* that are scaled by a factor of five, having *plugs* and *sockets* (similar to tabs and pockets for two-dimensional shapes in [3]). Consider a polycube P scaled by a factor of five resulting in P^5. We enumerate each layer of P^5 bottom to top modulo five. For each block of five layers,

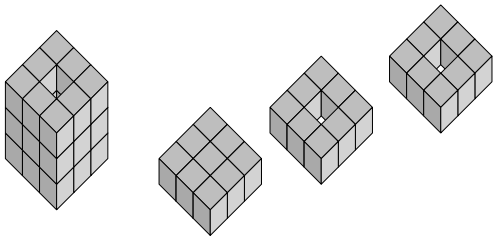


Figure 5 Left: A three-dimensional z-monotone polycube. Right: Three layers of P.

we create two polycubes as follows: P_1 gets layers 0 and 1 completely, and additionally the boundary tiles of layer 2, which form a socket (brown in Fig. 6). If there is a layer -1 below layer 0, then we form a plug with the tiles from layer -1. Let L_{-1} be the tiles in layer -1 having a neighbor in layer 0. We add all tiles of L_{-1} to P_1 except its boundary tiles (turquoise in Fig. 6).

The second polycube, P_2, gets layer 3 and 4 completely, and additionally all tiles of layer 2 except its boundary tiles, which form a plug (turquoise in Fig. 6). If there is a layer above layer 4, namely layer 5, then we form a socket. Let L_4 be the set of tiles of layer 4 having a neighbor in layer 5. Then we remove all tiles except the boundary pixels of L_4 from P_2 (brown in Fig. 6).

Performing this procedure for each block of five layers results in a decomposition into slices (an example is given in Fig. 6). Before we start describing how to assemble the slices, we explain how the slices can be used to get the final shape (note that this construction is similar to a rectangle decomposition in 2D [4]). We cut between two slices that split the polycube in half. We give both, the plug and the socket, a set of glue types, say G_1 (see Fig. 7 left). At the next cut we use the set of glue types G_2. Let G_3 be the third set of glues. At each recursion step, any subpolycube $\tilde{P}$ uses two of the three sets. We cut between two slices in $\tilde{P}$ and use the third set of glues for the new plug and the socket. This gives us a decomposition tree with logarithmic height and linear width, thus we need $O(\log n)$ stages and $O(n)$ bins.

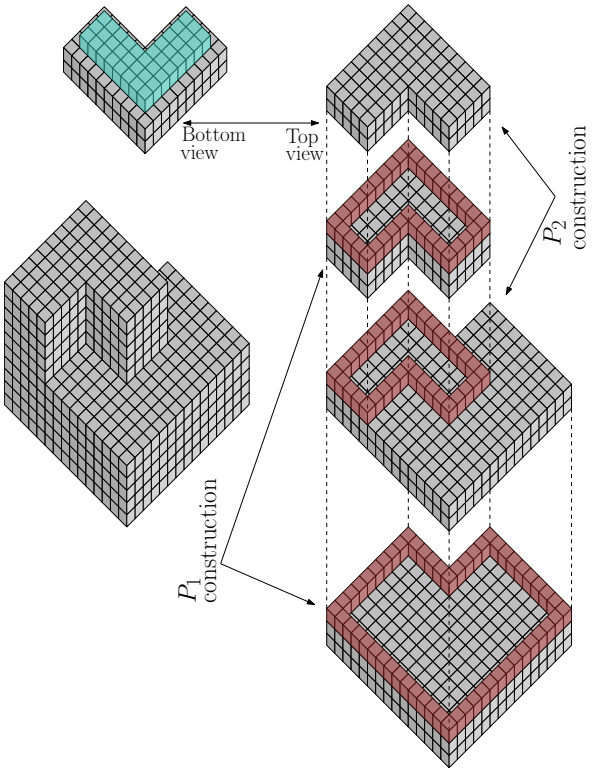


Figure 6 Decomposition of a polyomino (left) into slices (right). Slices have sockets (brown tiles) on the top and plugs on the bottom side (turquoise tiles).

▶ **Lemma 4.** *A slice can be constructed within $O(\log n)$ stages using $O(1)$ glue types, $O(k)$ bins with full connectivity at temperature $\tau = 1$.*

Proof. For each slice we have at least one layer with all tiles, one socket on the top and a plug on the bottom. Consider the tiles T_p we removed to obtain the plug, i.e., the tile of the socket of

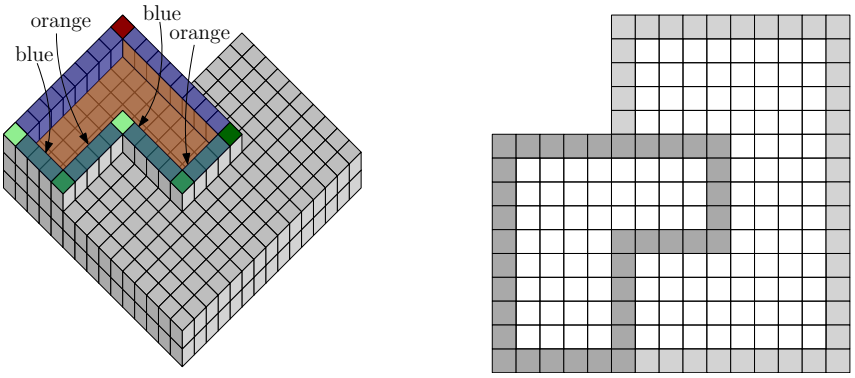


Figure 7 A slice with $O(1)$ glue types for the socket and its 2D projection.

the slice below the current slice. Furthermore, let T_s be the set of tiles of the socket. If we look at a 2D projection of T_p, T_s and a complete layer P_L, i.e., we ignore the z-direction, then at least one of T_p or T_s is the same as the boundary tiles T_L of P_L.

W.l.o.g., let T_s be unequal to T_L. Due to space constraints we only consider the case where P_L has no holes and T_s and T_L have common tiles (in other cases we can make use of tunnels like in Section 3). Let $\tilde{P}$ be the polyomino obtained by projecting T_s, T_p and T_L onto the x-y-plane (see Fig. 7 right). The tiles of $\tilde{P}$ can be divided into three types of tiles: (i) tiles only belonging to T_L, (ii) tiles only belonging to T_s, and (iii) tiles belonging to both, T_L and T_s. We can decompose $\tilde{P}$ by cutting off type (i) and type (ii) tiles with $O(\log n)$ cuts first, such that only lines with tiles of type (i), (ii) or (iii) remain. These lines can be built within $O(\log n)$ stages with correct glue types on their sides.

We can now fill up the slice with tiles that have glues on the top and bottom encoding (1) the glue type of the socket above, (2) the glue type of the plug below the current pixel, and (3) the distance to the top/bottom of the slice. Because there are only a constant number of such combinations possible, also a constant number of glue types will be sufficient. It remains to differentiate between tiles that have the socket above and those that do not have the socket above. We use 25 glue types to fill up the area, that is enclosed by type (ii) and (iii) tiles, as shown in Section 3. Another 25 glue types can be used to fill up the remaining area. In total these are $O(1)$ glue types. ◀

▶ **Theorem 5.** *A z-monotone polycube can be constructed within $O(\log n)$ stages using $O(1)$ glue types, $O(n)$ bins with full connectivity, and a scale factor five at temperature $\tau = 1$.*

5 Conclusion and Future Work

In this paper we started the first investigation of three-dimensional shapes within the staged self-assembly model. We showed that monotone three-dimensional shapes can be assembled within $O(\log n)$ stages, $O(1)$ glue types, $O(n)$ bins using full connectivity and a scale factor of 5. Future work could consider to look at upper and lower bounds for the stage complexity. Is it still possible to assemble an arbitrary polycube within $O(\log n)$ stages and only $O(1)$ glue types?

References

1 C. Chalk, E. Martinez, R. Schweller, L. Vega, A. Winslow, and T. Wylie. Optimal staged self-assembly of general shapes. *Algorithmica*, pages 1–27, 2016.

2 H.-L. Chen and D. Doty. Parallelism and time in hierarchical self-assembly. *SIAM Journal on Computing*, 46(2):661–709, 2017.

3 E. D. Demaine, M. L. Demaine, S. P. Fekete, M. Ishaque, E. Rafalin, R. T. Schweller, and D. L. Souvaine. Staged self-assembly: nanomanufacture of arbitrary shapes with O(1) glues. *Natural Computing*, 7(3):347–370, 2008.

4 E. D. Demaine, S. P. Fekete, C. Scheffer, and A. Schmidt. New geometric algorithms for fully connected staged self-assembly. *Theoretical Computer Science*, 671:4–18, 2017.

5 J. E. Padilla, W. Liu, and N. C. Seeman. Hierarchical self assembly of patterns from the robinson tilings: DNA tile design in an enhanced tile assembly model. *Natural Computing*, 11(2):323–338, 2012.

6 S. H. Park, C. Pistol, S. J. Ahn, J. H. Reif, A. R. Lebeck, C. Dwyer, and T. H. LaBean. Finite-size, fully addressable DNA tile lattices formed by hierarchical assembly procedures. *Angewandte Chemie*, 118(5):749–753, 2006.

7 J. H. Reif. Local parallel biomolecular computation. In *DNA-Based Computers*, volume 3, pages 217–254, 1999.

8 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.

9 K. Somei, S. Kaneda, T. Fujii, and S. Murata. A microfluidic device for DNA tile self-assembly. In *DNA Computing (DNA 11)*, pages 325–335. 2006.

10 H. Wang. Proving theorems by pattern recognition—II. *Bell system technical journal*, 40(1):1–41, 1961.

11 E. Winfree. *Algorithmic self-assembly of DNA*. PhD thesis, California Institute of Technology, 1998.

$\mathcal{O}$-Hull Formation for Programmable Matter

Joshua J. Daymude[1], Robert Gmyr[2], Kristian Hinnenthal[3], Irina Kostitsyna[4], Christian Scheideler[3], and Andréa W. Richa[1]

1 **Computer Science, CIDSE, Arizona State University, Tempe, AZ, USA**
{jdaymude, aricha}@asu.edu
2 **Department of Computer Science, University of Houston, Houston, TX, USA**
rgmyr@uh.edu
3 **Department of Computer Science, Paderborn University, Paderborn, Germany**
{krijan, scheidel}@mail.upb.de
4 **Department of Mathematics and Computer Science, TU Eindhoven, the Netherlands**
i.kostitsyna@tue.nl

1 Introduction

Research in *self-organizing programmable matter* is becoming increasingly popular in many fields with potential for broad applications, for example, in nanomedicine. Imagine tiny particles locating and repairing small wounds in the human body, or capturing harmful cells and transporting them out of the body. In this context various problems such as *shape formation* [6, 15, 10, 12], *coating* [7, 1], and *shape recognition* [9] have recently been investigated under various theoretical models. Somewhat in between these lies the problem of *shape sealing*, where the goal is to isolate an object by enclosing it with a shell of particles.

In this paper we study the problem of sealing a 2D object under the *amoebot model* [5, 4], which models programmable matter as a collection of nanoscale agents (called *particles*) with limited computational capabilities that move on a grid and can locally exchange information in order to collectively achieve a given goal.

The Amoebot Model. In the amoebot model the underlying geometry is an infinite triangular lattice $G_\triangle = (V, E)$. Each particle occupies either a single node in V (*contracted* particle) or a pair of adjacent nodes in V (*expanded* particle). Particles move via a series of *expansions* and *contractions*: a contracted particle can expand into an unoccupied adjacent node, and contract into one of its nodes (see Fig. 1). Neighboring particles can coordinate their movements in a *handover*, which can occur when: a contracted particle P "pushes" an expanded neighbor Q by expanding into a node occupied by Q, forcing it to contract; or an expanded particle Q "pulls" a contracted neighbor P by contracting, forcing P to expand into the node it is vacating. Handovers help maintain the connectivity of the particle system.

The particles are assumed to be anonymous, with no global coordinate system or compass. The only assumption is that the particles have a common *chirality*, which allows them to number the incident edges in clockwise order.

We assume the standard asynchronous model of distributed computing (see, e.g., [11]). A classical result under this model states that for any concurrent asynchronous execution of atomic actions, there exists a sequential ordering of actions producing the same end result, provided conflicts that arise in the concurrent execution are resolved. In the amoebot model, an atomic action corresponds to a single particle activation in which a particle can perform some computation involving its memory and the memories of its neighbors and at most one expansion or contraction. Conflicts involving concurrent memory writes or simultaneous particle expansions into the same unoccupied node are resolved arbitrarily such that at most one particle is writing into a given memory location or expanding into a given node at a time.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

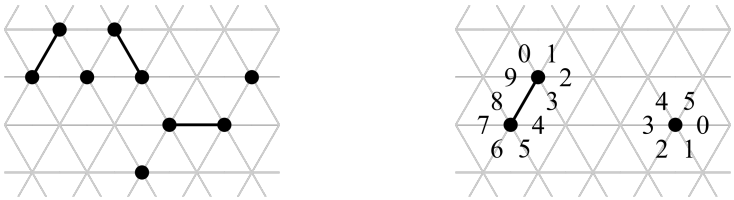


Figure 1 Left: Expanded and contracted particles. Right: particles with edge numbering.

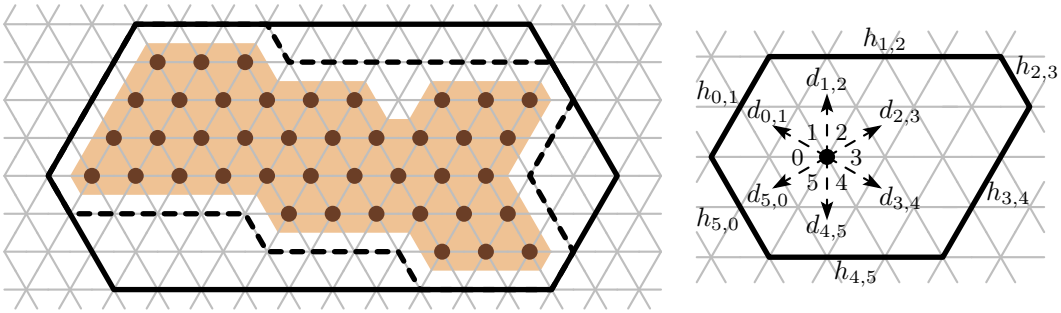


Figure 2 Left: An example of an object S (highlighted in orange), enclosed by its strong $\mathcal{O}$-hull (solid line) and its $\mathcal{O}$-hull (dashed line). Right: A particle's local labeling of the six half-planes composing the strong $\mathcal{O}$-hull; $\mathcal{D} = \{2, 2, 4, 3, 2, 2\}$.

While in reality many particles may be active concurrently, when analyzing our algorithms it suffices to consider a sequence of activations where only one particle is active at a time. We assume the activation sequence is *fair*: any particle P will be activated at some future time. An *asynchronous round* is complete once all particles have been activated at least once.

Problem Description. Shape sealing in two dimensions reduces to enclosing an object in a cycle. To optimize the number of particles needed to seal an object, we study the problem of particles forming a convex hull. The amoebot model limits the movement of the particles to three directions, thus we build a *restricted-orientation hull*, or an *$\mathcal{O}$-hull*, of a given object.

The notions of $\mathcal{O}$-convexity and an $\mathcal{O}$-hull were introduced by Rawlins [14] (see also [8]). Given a set of fixed orientations $\mathcal{O}$, a set is *$\mathcal{O}$-convex* if its intersection with any line with one of the orientations in $\mathcal{O}$ is connected. An *$\mathcal{O}$-hull* of a given set S is defined as an intersection of all $\mathcal{O}$-convex sets containing S. Furthermore, a *strong $\mathcal{O}$-hull* of S is an intersection of all half-planes bounded by lines with orientations in $\mathcal{O}$ and containing S. In our case $\mathcal{O}$ consists of three orientations of the axes of the triangular grid $G_\triangle$.

Let S be a simply-connected subgraph of $G_\triangle$, and $\mathcal{P}$ be a connected system of initially contracted amoebot particles on $G_\triangle$ (non-overlapping with S). The shape sealing problem is to reconfigure $\mathcal{P}$ within $G_\triangle \setminus S$ so that every node of the $\mathcal{O}$-hull of S is occupied by a contracted particle. Note that as the particles are not allowed to occupy the nodes of S, the hull that will be constructed by $\mathcal{P}$ will in fact be the offset-by-one $\mathcal{O}$-hull of S (see Fig. 2 (left)). Nevertheless, to simplify the exposition, we will refer to it using the same term $\mathcal{O}$-hull. We further assume that $\mathcal{P}$ has enough particles to form an $\mathcal{O}$-hull, that it contains a unique leader particle[1] ℓ initially adjacent to S, and that S does not contain any *tunnels* of width 1

[1] Such a particle can be determined in $O(|\mathcal{P}|)$ asynchronous rounds with high probability using a slightly modified version of the leader election algorithm of [3].

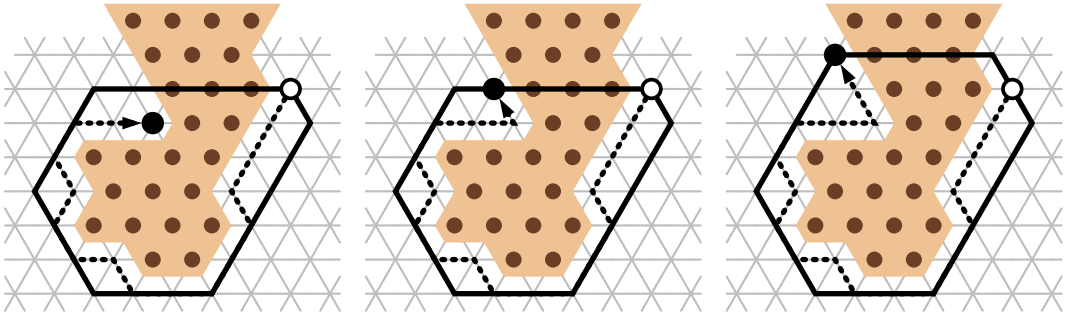


Figure 3 A particle (black dot) estimates strong $\mathcal{O}$-hull (black) after having traversed the dotted path from its starting point (black circle). Left: $d_h \geq 1$ for all $h \in \mathcal{H}$, the next move does not push any half-plane. Middle: $d_{1,2} = 0$. Right: Half-plane $d_{1,2}$ has been pushed.

(that is, $G_\triangle \setminus S$ is 2-connected).

We present a fully distributed, local algorithm for the shape sealing problem that runs in $O(B+H)$ asynchronous rounds, where B is the perimeter of S, and H is the perimeter of the $\mathcal{O}$-hull of S. We present the algorithm in three parts: we first describe how a single particle with unbounded memory can find the strong $\mathcal{O}$-hull using a simple geometric observation. The main difficulty of our result lies in emulating the single-particle algorithm with a system of bounded-memory particles. The final part, converting the strong $\mathcal{O}$-hull into the $\mathcal{O}$-hull, is rather straightforward: all particles at the convex vertices that are not adjacent to S can move inside thus "deflating" the strong $\mathcal{O}$-hull towards the $\mathcal{O}$-hull; after additional $O(H)$ asynchronous rounds the $\mathcal{O}$-hull is achieved. In the rest of this extended abstract we present the first two parts of the algorithm. Due to space constraints, we omit the proofs of the theorems, which can be found in the full version of this paper [2].

2 Single Particle Algorithm

Consider a single particle P with unbounded memory. Let P be initially placed somewhere adjacent to S. The main idea of this algorithm is to let P traverse the boundary of S clockwise, while internally maintaining a representation of the strong $\mathcal{O}$-hull. The strong $\mathcal{O}$-hull can be represented with six half-planes $\mathcal{H} = \{h_{0,1}, h_{1,2}, \ldots, h_{5,0}\}$, which P can label clockwise (in Fig. 2 (right)). Particle P computes the locations of these half-planes by maintaining six counters $\mathcal{D} = \{d_h : h \in \mathcal{H}\}$, where d_h holds the distance from P to the line supporting h. If one of these counters is 0, P is on the current estimate of the strong $\mathcal{O}$-hull.

Counters initially are set to 0. As P moves, it always stays parallel to two half-planes; their counters do not get updated. For two other half-planes P gets further away; their counters get incremented. For each of the remaining two half-planes, P either moves closer to it (if its counter was > 0) or steps outside of the half-plane (if the counter was $= 0$). In the former case the counter gets decremented, in the latter case the counter does not get updated which corresponds to a half-plane getting "pushed". Refer to Fig. 3 for an example.

Finally, P needs to detect when it has computed the strong $\mathcal{O}$-hull. To do so, it stores six terminating bits $\{b_h : h \in \mathcal{H}\}$, where $b_h = 1$ if P has visited the line supporting half-plane h since it last pushed any half-plane, and $b_h = 0$ otherwise. Whenever P moves without pushing a half-plane, for each h with $d_h = 0$ it sets b_h to 1. Otherwise, when P pushes a half-plane, it sets b_h to 0 for all h. If after a move all six terminating bits are 1, P contracts and terminates.

▶ **Theorem 2.1.** *The single-particle algorithm terminates after $O(B)$ asynchronous rounds with particle P holding the correct representation of the strong $\mathcal{O}$-hull in the six counters $\mathcal{D}$.*

3 The Strong $\mathcal{O}$-hull Algorithm

Next we show how a system of n particles each with only constant-size memory can emulate the single-particle algorithm. The leader particle ℓ of $\mathcal{P}$ is primarily responsible for emulating the particle with unbounded memory in the single-particle algorithm. To do so, it utilizes the other particles in the system as distributed memory. More precisely, as ℓ moves, it will create a chain of particles behind it that will be used to store the distances d_h from ℓ to the lines supporting half-planes h as binary counters. Once these measurements are complete, ℓ uses them to lead the other particles in forming the $\mathcal{O}$-hull.

A Binary Counter of Particles. We build upon an increment-only binary counter under the amoebot model [13]. Suppose that the participating particles are organized as a simple chain with the leader at its front. Each particle P has a bit value $P.bit$, that can be empty if P is not part of the counter. A *final token* f is held by a particle marking the end of the counter. Then the counter value is represented by the bits of the particles from ℓ (holding the least significant bit) up to the particle holding the token f.

The leader ℓ initiates counter operations, and the rest of the particles carry these operations out. To increment (decrement) the counter, the leader ℓ generates an increment token c^+ (decrement token c^-). The tokens are consumed or passed along the chain (as a carry bit) while updating the bits of the particles accordingly until they are consumed. To test whether the counter is 0, the leader checks the status of its follower counter particle P_1. If P_1 is holding a decrement token c^- and $P_1.bit = 1$, ℓ cannot conclusively test whether the counter's value is 0. Otherwise, the counter value is 0 if and only if $\ell.bit = 0$, P_1 is holding the final token f, and P_1 is not holding an increment token c^+.

The proof of the following theorem is rather involved, we omit it due to space constraints.

▶ **Theorem 3.1.** *Given any fair asynchronous activation sequence of the particles, and any nonnegative sequence of m operations, the distributed binary counter correctly processes all operations in $O(m)$ asynchronous rounds.*

3.1 Estimating the Strong $\mathcal{O}$-Hull

We now combine the movement rules of the single-particle algorithm with our distributed, multi-particle binary counter to enable the leader to compute the strong $\mathcal{O}$-hull of S.

First, using the *spanning tree primitive* (see, e.g., [7]), a spanning tree is constructed on the particle system rooted at the leader ℓ. Each activated particle P, if it has a neighbor Q already in the tree, becomes a *follower* and sets $P.parent$ to Q. Note that the leader can immediately begin estimating the strong $\mathcal{O}$-hull without waiting for the entire tree to form.

The first few followers of ℓ form a counter chain and store six counters d_h in a distributed fashion. Imitating the single-particle algorithm, ℓ performs a clockwise traversal of the boundary of S using the right-hand rule, updating its counters along the way. It terminates once it has moved in all six directions without pushing a half-plane, which it detects using its six terminating bits b_h. In the multi-particle setting, we need to carefully consider both how ℓ interacts with its followers as it moves and how it updates its counters.

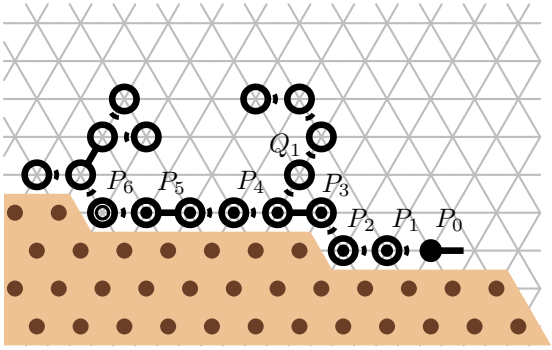


Figure 4 The leader P_0 and its followers. Followers with dots are on the counter, and P_6 holds the final token. Particle Q_1 cannot handover with P_3, while all other potential handovers are safe.

Rules for Distributed Counters. The increment and decrement tokens are handled as described above. However, as at the beginning the particles form a tree, and not a simple path, we enforce that the counters are only extended along followers on the object's boundary.

As there may be role-swaps of the leader ℓ (described below), to maintain connectivity of the counters we modify them to store two bits per particle. Then if a role-swap occurs, the counter bits will need to be shifted towards the leader to keep all the bits of the particles closest to ℓ "full". This can be easily achieved by passing the bit value when a counter particle P detects that there is a bit value missing in the memory of its parent $P.parent$.

Rules for Leader Computation and Movement. First suppose ℓ is contracted. If all its terminating bits $b_h = 1$, then ℓ has computed the strong $\mathcal{O}$-hull. Else, if the zero-test operation is unavailable on any of the counters, ℓ skips its turn; otherwise, ℓ will attempt to expand into the node v along the boundary of S. If v is unoccupied, or is occupied by an expanded particle, ℓ calculates the updated distances $\mathcal{D}$, generates the corresponding increment/decrement tokens and expands into v. If v is occupied by a contracted particle P, ℓ will have to initiate a role-swap with P, such that P becomes the new leader and ℓ becomes its follower (the second particle in the counter). This is allowed only if ℓ has its both bit values full, or if it is holding a final token f_h. In the former case ℓ passes the value only of one least significant bit to P (this is where the two bits are used to maintain the connectivity), and in the latter case ℓ passes a bit (if it exists) and the final token to P. It also updates its terminating bits b_h for all $h \in \mathcal{H}$.

Finally, if ℓ is expanded, let P be its follower child emulating bits of the counters. Then if P is contracted, ℓ pulls P in a handover.

Rules for Follower Movement. For any follower P, if it is expanded and has no children in the spanning tree nor any non-tree neighbor, then it simply contracts. If P is contracted and is following the tail of its expanded parent $Q = P.parent$, then P pushes Q in a handover. Similarly, if P is expanded and has a contracted child Q, P pulls Q in a handover. However, we do not allow handovers that may disconnect the counters (see Fig. 4).

Once the counters contain an accurate representation of the strong $\mathcal{O}$-hull, the leader ℓ can simply lead the particle system in tracing it out by traversing the strong $\mathcal{O}$-hull in clockwise order. While moving along the strong $\mathcal{O}$-hull, ℓ uses its distributed counters to detect when it reaches a vertex of the strong $\mathcal{O}$-hull, at which point it turns 60° to follow the next half-plane, and so on. The movement rules for the leader and the followers in this

phase are very similar to those of the previous phase. With some careful analysis we can show the following theorem.

▶ **Theorem 3.2.** *The presented algorithm solves the strong $\mathcal{O}$-hull formation problem for an object S in $O(B + H)$ asynchronous rounds in the worst case.*

Acknowledgments. Joshua J. Daymude and Andrá W. Richa are supported in part by NSF Awards CCF-1637393 and CCF-1733680.

References

1 J. J. Daymude, Z. Derakhshandeh, R. Gmyr, A. Porter, A. W. Richa, C. Scheideler, and T. Strothmann. On the runtime of universal coating for programmable matter. *Natural Computing*, 17(1):81–96, 2018.
2 J. J. Daymude, R. Gmyr, K. Hinnenthal, I. Kostitsyna, C. Scheideler, and A. W. Richa. Convex hull formation for programmable matter. Available on arXiv, 2018.
3 J. J. Daymude, R. Gmyr, A. W. Richa, C. Scheideler, and T. Strothmann. Improved leader election for self-organizing programmable matter. In *Algorithms for Sensor Systems (ALGOSENSORS)*, pages 127–140, 2017.
4 J. J. Daymude, A. W. Richa, and C. Scheideler. The amoebot model. Available online at `https://sops.engineering.asu.edu/sops/amoebot`, 2017.
5 Z. Derakhshandeh, S. Dolev, R. Gmyr, A. W. Richa, C. Scheideler, and T. Strothmann. Brief announcement: Amoebot - a new model for programmable matter. In *Proc. 26th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 220–222, 2014.
6 Z. Derakhshandeh, R. Gmyr, A. W. Richa, C. Scheideler, and T. Strothmann. Universal shape formation for programmable matter. In *Proc. 28th Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 289–299, 2016.
7 Z. Derakhshandeh, R. Gmyr, A. W. Richa, C. Scheideler, and T. Strothmann. Universal coating for programmable matter. *Theoretical Computer Science*, 671:56–68, 2017.
8 E. Fink and D. Wood. *Restricted-Orientation Convexity*. Monographs in Theoretical Computer Science. An EATCS Series. Springer-Verlag Berlin Heidelberg, 2004.
9 R. Gmyr, K. Hinnenthal, I. Kostitsyna, F. Kuhn, D. Rudolph, and C. Scheideler. Shape Recognition by a Finite Automaton Robot. In *Proc. 43rd International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 117 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 52:1–52:15, 2018.
10 F. Hurtado, E. Molina, S. Ramaswami, and V. Sacristán. Distributed reconfiguration of 2D lattice-based modular robotic systems. *Autonomous Robots*, 38(4):383–413, 2015.
11 N. Lynch. *Distributed Algorithms*. Morgan Kauffman, 1996.
12 M. J. Patitz. An introduction to tile-based self-assembly and a survey of recent results. *Natural Computing*, 13(2):195–224, 2014.
13 A. Porter and A. W. Richa. Collaborative computation in self-organizing particle systems. In *Proc. 17th International Conference on Unconventional Computing and Natural Computation (UCNC)*, 2018.
14 G. J. E. Rawlins. *Explorations in Restricted Orientation Geometry*. PhD thesis, University of Waterloo, 1987. AAI0561642.
15 D. Woods, H.-L. Chen, S. Goodfriend, N. Dabby, E. Winfree, and P. Yin. Active self-assembly of algorithmic shapes and patterns in polylogarithmic time. In *Proc. 4th Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 353–354, 2013.

Characterization and Computation of the Feasible Space of an Articulated Probe

Ovidiu Daescu and Ka Yaw Teo

Department of Computer Science, University of Texas at Dallas, Richardson, TX, USA
{ovidiu.daescu, ka.teo}@utdallas.edu

Abstract

We present an efficient algorithm for computing the feasible solution space for a trajectory planning problem involving an articulated two-link probe constrained to a fixed sequence of motions – a straight line insertion, possibly followed by a rotation of the end link. Given n line segment obstacles in the workspace, we show that the feasible trajectory space of the articulated probe can be characterized by an arrangement of simple curves of complexity $O(k)$, which can be constructed in $O(n \log n + k)$ time using $O(n+k)$ space, where $k = O(n^2)$ is the number of vertices of the arrangement. Additionally, our solution approach produces a new data structure for solving a special case of the circular sector intersection query problem.

1 Introduction

In minimally invasive surgeries, a rigid needle-like robotic arm is typically inserted through a small incision to reach its given target, after which it may perform operations such as tissue resection and biopsy. Some newly developed variants allow for a joint to be close to the acting end (tip) of the arm; after inserting the arm in a straight path, the surgeon may rotate the tip around the joint to reach the target. This enhances the ability to reach deep targets but greatly increases the complexity of finding acceptable insertion/rotation pairs.

Unlike polygonal linkages that can rotate freely at the joints while moving between a start and target configuration [2, 8, 9], a simple articulated probe is constrained to a fixed sequence of moves – a straight line insertion, possibly followed by a rotation of the short link. This type of motion has not received attention until very recently [3, 4].

As originally proposed in [4], an articulated probe is modeled in $\Re^2$ as two line segments, ab and bc, joined at point b. The length of ab can be arbitrarily large (infinitely long), while bc, corresponding to the tip of the probe, has a fixed length r. A two-dimensional workspace (see Figure 1) is given by the region bounded by a large circle S of radius R centered at t, enclosing a set P of n disjoint line segment obstacles and a target point t in the free space.

At the start, the probe is outside S and assumes an *unarticulated* configuration, in which ab and bc are collinear, with $b \in ac$. A *feasible probe trajectory* consists of an initial insertion of straight line segment abc, possibly followed by a rotation of bc at b up to $\pi/2$ radians in either direction, such that point c ends at t, while avoiding obstacles in the process. If a rotation is performed, then we have an *articulated final* configuration of the probe.

The objective of this paper is to characterize and compute the feasible trajectory space (i.e., set of all feasible trajectories) of the articulated probe.

Previous work. The articulated probe problem in two dimensions was formally introduced in [4], where an algorithm was presented for finding so-called *extremal* feasible probe trajectories in $O(n^2 \log n)$ time using $O(n \log n)$ space. In an extremal probe trajectory, the probe is tangent to one or two obstacle endpoints. Later, it was shown in [3] that, for any constant $\delta > 0$, a feasible probe trajectory with a clearance δ from the obstacles can be determined in $O(n^2 \log n)$ time using $O(n^2)$ space.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

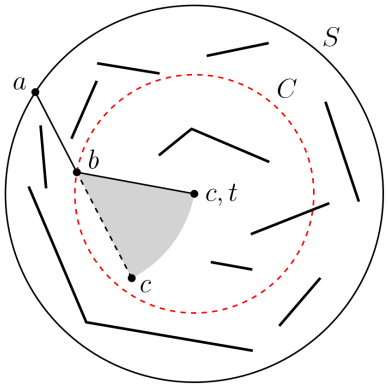


Figure 1 Trajectory planning for an articulated probe. In order to reach point t, a straight insertion of line segment abc may be followed by a rotation of line segment bc from its intermediate position (black dashed line) to the final position (black solid line).

Results and contributions. We describe a geometric combinatorial approach for characterizing and computing the feasible trajectory space of the articulated probe. The feasible configuration space has worst-case complexity of $O(n^2)$ and can be described by an arrangement of simple curves. By using the topological sweep method [1], the arrangement can be constructed in $O(n \log n + k)$ time using $O(n + k)$ working storage, where k is the number of vertices of the arrangement. Our approach also results in a simplified data structure of similar space/time complexity compared to that in [4] for solving a special instance of the *circular sector intersection query* problem (i.e., for a query circular sector with a fixed radius r and a fixed arc endpoint t).

2 Solution approach

We characterize 1) the final configuration space, 2) the *forbidden* final configuration space, and 3) the *infeasible* final configuration space, as detailed in this section.

2.1 Final configuration space

In a final configuration of the articulated probe, point a can be assumed to be located on S, and point b lies on a circle C of radius r centered at t (see Figure 1). Let θ_S and θ_C be the angles of ta and tb relative to the x-axis, respectively, where $\theta_S, \theta_C \in [0, 2\pi)$. Since bc may rotate as far as $\pi/2$ radians in either direction, for any given θ_S, we have $\theta_C \in [\theta_S - \cos^{-1} r/R, \theta_S + \cos^{-1} r/R]$. We call this the *unforbidden* range of θ_C. A final configuration of the articulated probe can be specified by a two-tuple (θ_S, θ_C), depending on the locations of points a and b on circles S and C, respectively (see Figure 2). The final configuration space Σ_{fin} of the articulated probe can be computed in $O(1)$ time.

2.2 Forbidden final configuration space

A final configuration is called *forbidden* if the final configuration (represented by ab and bt) intersects with one or more of the obstacle line segments. The following two cases arise.

Case 1. Obstacle line segment s outside C. The corresponding forbidden final configuration space can be characterized as follows. Let angles θ_i, where $i = 1, \ldots, 6$, be defined

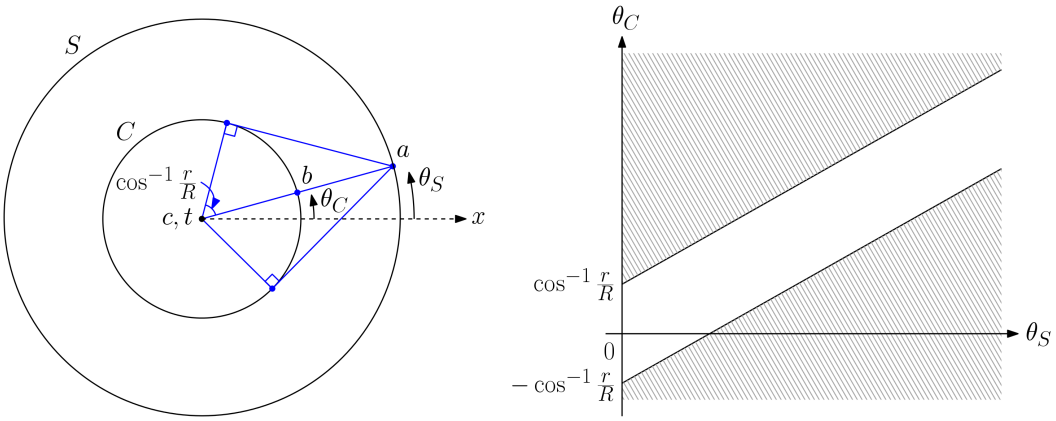


Figure 2 Final configurations of the probe. The unshaded region of the (θ_S, θ_C)-plot represents the unforbidden final configuration space when the workspace contains no obstacles.

in the manner depicted in Figure 3. Briefly, each θ_i corresponds to an angle θ_S at which point a tangent line between C and s, or from t to s, intersects S. As θ_S increases from θ_1 to θ_3, the upper bound of the unforbidden range of θ_C decreases as a continuous function of θ_S. Similarly, when θ_S varies from θ_4 to θ_6, the lower bound of the unforbidden range of θ_C decreases as a continuous function of θ_S. For $\theta_3 \leq \theta_S \leq \theta_4$, there exists no unforbidden final configuration at any θ_C. For conciseness, the upper (resp. lower) bound of the unforbidden range of θ_C is simply referred to as the upper (resp. lower) bound of θ_C hereafter.

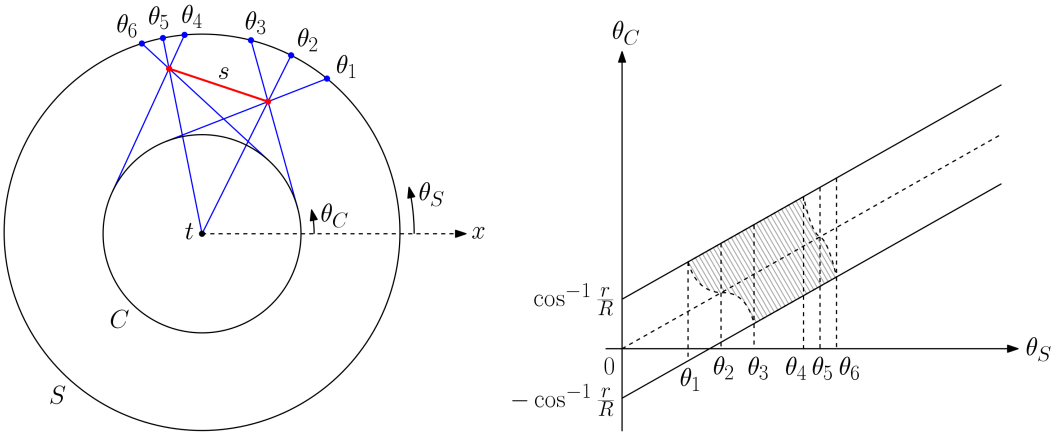


Figure 3 Forbidden final configurations due to an obstacle line segment s outside C.

Case 2. Obstacle line segment s inside C. We compute the forbidden final configuration space for s in a similar way. Note that, as shown in Figure 4, angles θ_i (where $i = 1, \ldots, 6$) are defined differently from the previous case. For $\theta_1 \leq \theta_S \leq \theta_4$, the upper bound of θ_C is equivalent to θ_2. For $\theta_3 \leq \theta_S \leq \theta_6$, the lower bound of θ_C equals to θ_5.

We can find the forbidden final configuration space for an obstacle line segment (i.e., final configuration obstacle) in $O(1)$ time. Thus, for n obstacle line segments, it takes a total of $O(n)$ time to compute the corresponding set of configurations. The union of these configurations forms the forbidden final configuration space $\Sigma_{fin,forb}$ of the articulated probe.

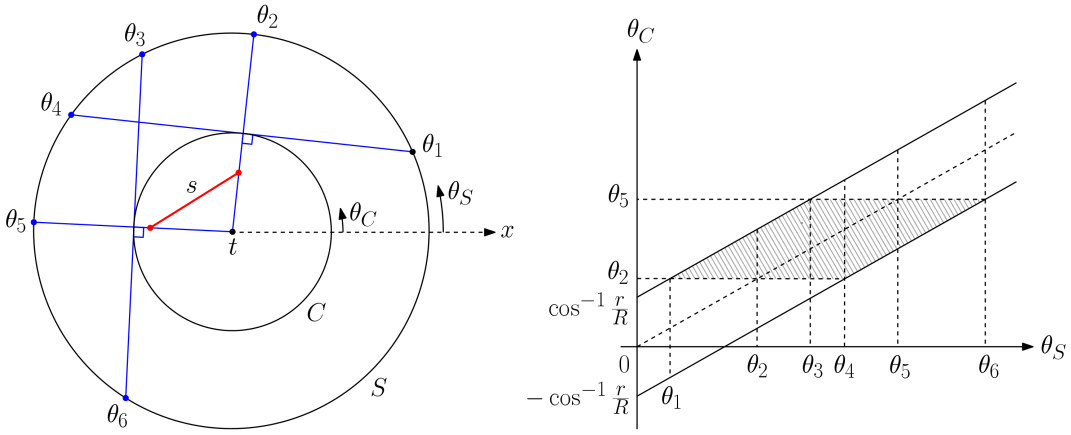


Figure 4 Forbidden final configurations due to an obstacle line segment s inside C.

The free final configuration space $\Sigma_{fin,free}$ of the articulated probe is the complement of $\Sigma_{fin,forb}$; that is, $\Sigma_{fin,free} = \Sigma_{fin} \setminus \Sigma_{fin,forb}$.

2.3 Infeasible final configuration space

The feasible trajectory space of the articulated probe can be characterized as a subset of $\Sigma_{fin,free}$. A final configuration is called *infeasible* if the circular sector associated with the final configuration (i.e., the area swept by segment bc to reach target point t) intersects with any obstacle line segment. We denote the infeasible final configuration space as $\Sigma_{fin,inf}$.

Let C' be a circle centered at t and of radius $\sqrt{2}r$. A circular sector associated with a final configuration can only intersect with an obstacle line segment lying inside C'. In contrast to characterizing the lower and upper bounds of θ_C as θ_S varies from 0 to 2π as in the prior section, we herein perform the characterization in reverse. For conciseness, we only present arguments for the negative half of the θ_S-range, which is $[\theta_C - \cos^{-1} r/R, \theta_C]$, and the similar arguments apply to the other half due to symmetry. As before, two cases arise.

Case 1. Obstacle line segment s inside C. As shown in Figure 5, we are only concerned with computing the lower bound of θ_S for $\theta_C \in [\phi_1, \phi_2]$, given that the entire negative half of the θ_S-range (i.e., $[\theta_C - \cos^{-1} r/R, \theta_C]$) is feasible for $\theta_C \in [0, \phi_1] \cup [\phi_3, 2\pi)$, and is infeasible for $\theta_C \in [\phi_2, \phi_3]$ due to intersection of bt with s.

For brevity, the quarter-circular sector associated with a point b (i.e., the maximum possible area swept by segment bc to reach point t), where the angle of tb (relative to the x-axis) is θ_C, is henceforth referred to as the *quarter-circular sector associated with* θ_C.

ϕ_1, ϕ_2 and ϕ_3 can be described in brief as follows (see Figure 5a). ϕ_1 is the smallest angle θ_C at which the circular arc (of the quarter-circular sector associated with θ_C) intersects with s (at one of its endpoints or interior points). ϕ_2 is the smallest angle θ_C at which bt (of the quarter-circular sector associated with θ_C) intersects with s (at one of its endpoints). ϕ_3 is the largest angle θ_C at which bt (of the quarter-circular sector associated with θ_C) intersects with s (at one of its endpoints). In other words, as θ_C varies from 0 to 2π, ϕ_1 and ϕ_3 are the angles θ_C at which the quarter-circular sector associated with θ_C first and last intersects with s, respectively.

For $\theta_C \in [\phi_1, \phi_2]$, the lower bound of θ_S can be represented by a piecewise continuous curve, which consists of at most two pieces, corresponding to two intervals $[\phi_1, \alpha]$ and $[\alpha, \phi_2]$,

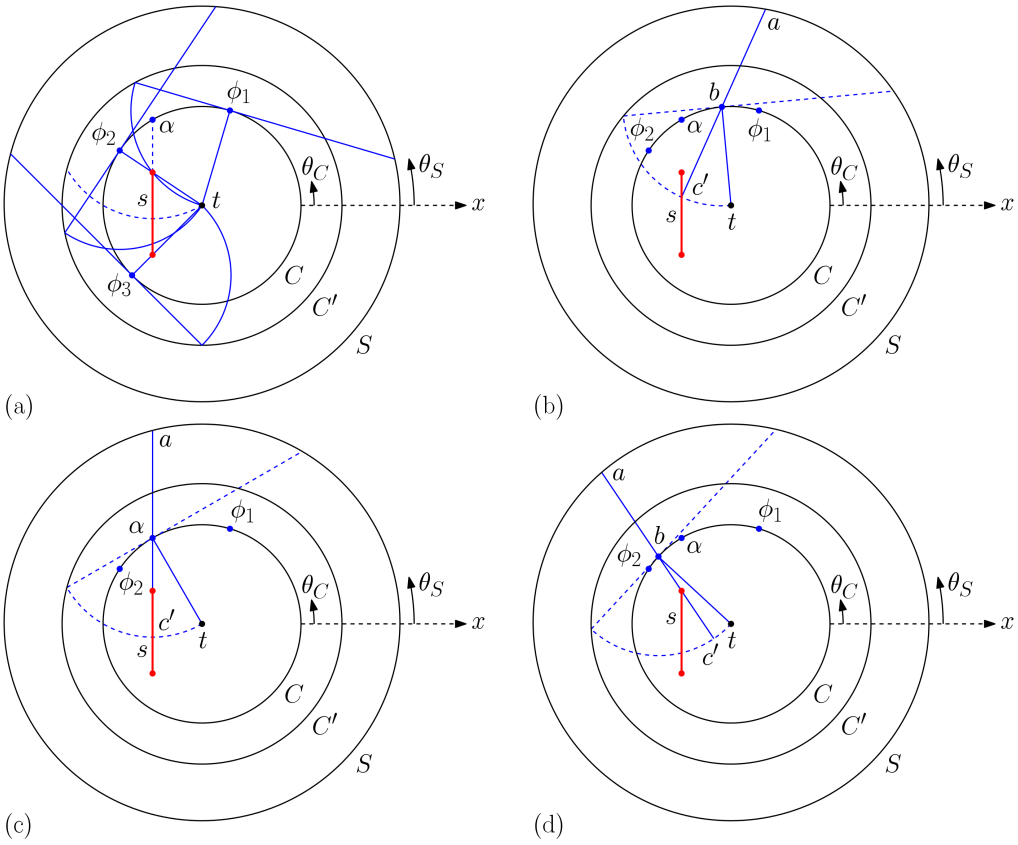


Figure 5 Infeasible final configurations due to an obstacle line segment s inside C. Illustration of θ_S-lower bound for (a) $\theta_C \in [\phi_1, \phi_2]$, (b) $\phi_1 < \theta_C < \alpha$, (c) $\theta_C = \alpha$, and (d) $\alpha < \theta_C < \phi_2$.

where α is the angle θ_C of the intersection point between C and the supporting line of s. Note that, if $\phi_1 \leq \alpha$, then the lower-bound curve of θ_S has two pieces; otherwise, the curve is composed of one single piece.

For $\theta_C \in [\phi_1, \alpha]$, as depicted in Figure 5b, the lower bound of θ_S is indicated by point a of straight line segment abc' (i.e., intermediate configuration), where c' is the intersection point between the circular arc centered at b and obstacle line segment s. If no intersection occurs between the circular arc and obstacle line segment s, then the lower bound of θ_S is given by point a of straight line segment abc', where bc' intersects with the endpoint of obstacle line segment s farthest from point b.

For $\theta_C \in [\alpha, \phi_2]$, the lower bound of θ_S is indicated by point a of straight line segment abc', where bc' intersects with the endpoint of obstacle line segment s closest to point b (see Figure 5d). Observe that the lower bound of θ_S is equivalent to θ_C when $\theta_C = \phi_2$. A sketch of the corresponding infeasible final configuration space is shown in Figure 6.

Case 2. Obstacle line segment s outside C and inside C'. As depicted in Figure 7, we only need to worry about computing the lower bound of θ_S for $\theta_C \in [\phi_1, \phi_2]$, given that the entire negative half of the θ_S-range (i.e., $[\theta_C - \cos^{-1} r/R, \theta_C]$) is feasible for $\theta_C \in [0, \phi_1] \cup [\phi_2, 2\pi)$. The analysis is similar to Case 1 and thus omitted. A sketch of the corresponding infeasible final configuration space is shown in Figure 8.

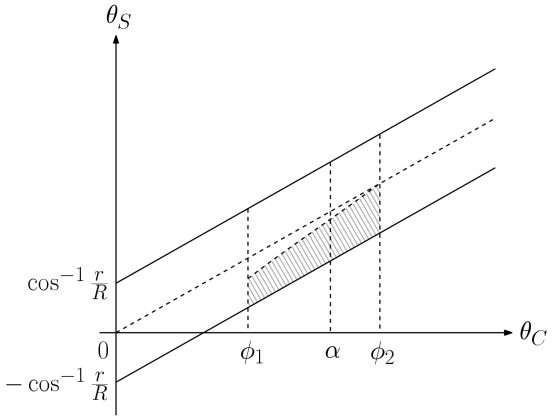


Figure 6 Infeasible final configuration space due to an obstacle line segment s inside C.

2.4 Complexity and construction of the feasible trajectory space

The feasible trajectory space of the articulated probe is represented by $\Sigma_{fin} \setminus (\Sigma_{fin,forb} \cup \Sigma_{fin,inf})$. A set of lower- and upper-bound curves – σ_{fin}, $\sigma_{fin,forb}$, and $\sigma_{fin,inf}$ – was obtained from characterizing the final, forbidden final, and infeasible final configuration spaces, respectively. Each of these curves is a function of θ_S – that is, $\theta_C(\theta_S)$.

As illustrated in Figure 2, σ_{fin} contains two linearly increasing curves, $\theta_C = \theta_S - \cos^{-1} r/R$ and $\theta_C = \theta_S + \cos^{-1} r/R$, which are totally defined over $\theta_S \in [0, 2\pi)$. Each curve in $\sigma_{fin,forb}$ is partially defined, continuous, and monotone in θ_S. Specifically, as shown in Figure 3 & 4, the curves in Case 1 are monotonically decreasing with respect to θ_S, and the curves in Case 2 are of zero slopes (i.e., of some constant θ_C). Furthermore, any two curves in Case 1 can only intersect at most once. Likewise, each curve in $\sigma_{fin,inf}$ is bounded and monotonically increasing with respect to θ_S (see Figure 6 & 8). Any curve in $\sigma_{fin,inf}$ can only intersect with another at most once.

From the observations above, it can be easily deduced that the number of intersections between any two curves in $\sigma = \sigma_{fin} \cup \sigma_{fin,forb} \cup \sigma_{fin,inf}$ is at most one. In other words, the curves of σ are essentially lines, line segments, or pseudo-line segments. For a set σ of $O(n)$ x-monotone Jordan arcs, bounded or unbounded, with at most c intersections per pair of arcs (for some fixed constant c), the maximum combinatorial complexity of the arrangement $A(\sigma)$ is $O(n^2)$ [6].

An incremental construction approach, as detailed in [5], can be used to construct arrangement $A(\sigma)$ in $O(n^2\alpha(n))$ time using $O(n^2)$ space, where $\alpha(n)$ is the inverse Ackermann function. By using topological sweep [1] in computing the intersections for a collection of well-behaved curves (e.g., Jordan curves described above), the time and space complexities can be improved to $O(n \log n + k)$ and $O(n + k)$, respectively.

▶ **Theorem 2.1.** *The feasible trajectory space of the articulated probe can be represented as a simple arrangement of maximum combinatorial complexity $k = O(n^2)$ and can be constructed in $O(n \log n + k)$ time using $O(n + k)$ space.*

Remark. The analytical approach above, with a slight change of parameterization and some additional data structure, can be used to solve a special case of the circular sector intersection query problem, and the result is summarized in Theorem 2.2.

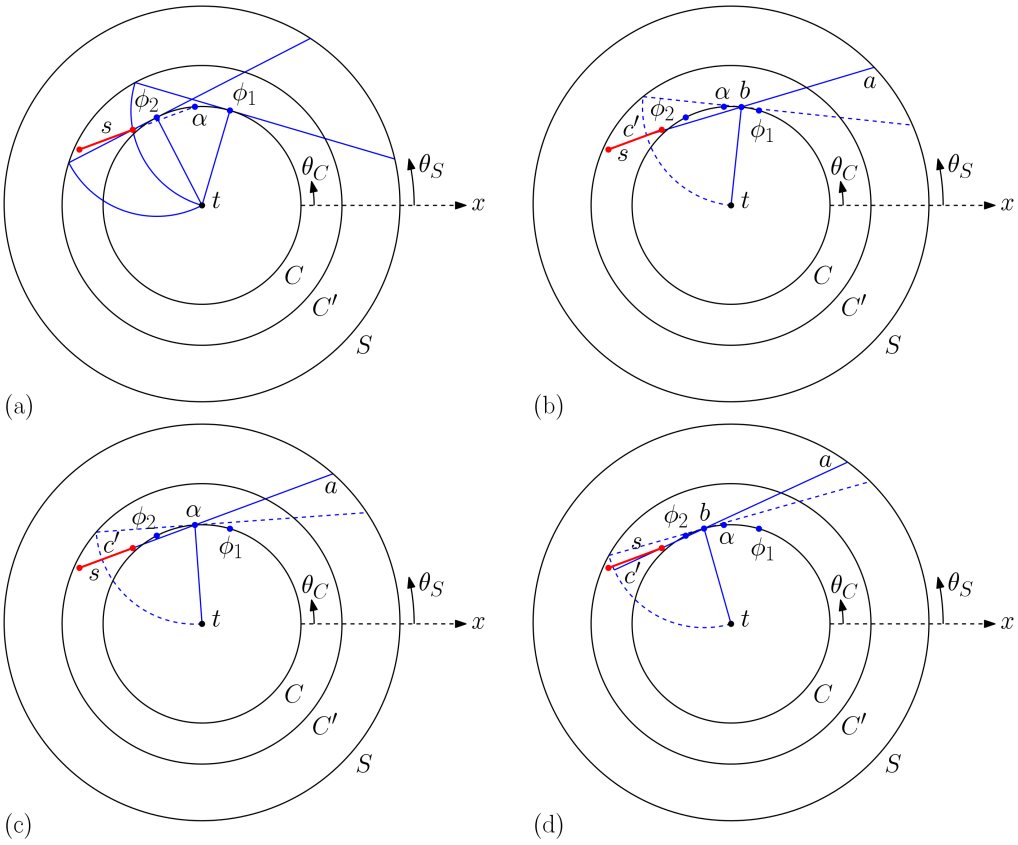


Figure 7 Infeasible final configurations due to an obstacle line segment s outside C and inside C'. Illustration of θ_S-lower bound for (a) $\theta_C \in [\phi_1, \phi_2]$, (b) $\phi_1 < \theta_C < \alpha$, (c) $\theta_C = \alpha$, and (d) $\alpha < \theta_C < \phi_2$.

▶ **Theorem 2.2.** *A set P of n line segments in $\Re^2$ can be preprocessed in $O(n \log n)$ time into a data structure of size $O(n\alpha(n))$ so that, for a query circular sector σ with a fixed radius r and a fixed arc endpoint t, one can determine if σ intersects P in $O(\log n)$ time.*

References

1. Ivan J Balaban. An optimal algorithm for finding segments intersections. In *Proceedings of the eleventh annual symposium on Computational geometry*, pages 211–219, 1995.
2. Robert Connelly and Erik D Demaine. Geometry and topology of polygonal linkages. *Handbook of Discrete and Computational Geometry*, pages 233–256, 2017.
3. Ovidiu Daescu, Kyle Fox, and Ka Yaw Teo. Computing trajectory with clearance for an articulated probe. In *28th Annual Fall Workshop on Computational Geometry*, 2018.
4. Ovidiu Daescu, Kyle Fox, and Ka Yaw Teo. Trajectory planning for an articulated probe. In *30th Annual Canadian Conference on Computational Geometry*, pages 296–303, 2018.
5. Herbert Edelsbrunner, Leonidas Guibas, János Pach, Richard Pollack, Raimund Seidel, and Micha Sharir. Arrangements of curves in the plane – topology, combinatorics, and algorithms. *Theoretical Computer Science*, 92(2):319–336, 1992.
6. Dan Halperin and Micha Sharir. Arrangements. *Handbook of Discrete and Computational Geometry*, pages 723–762, 2017.

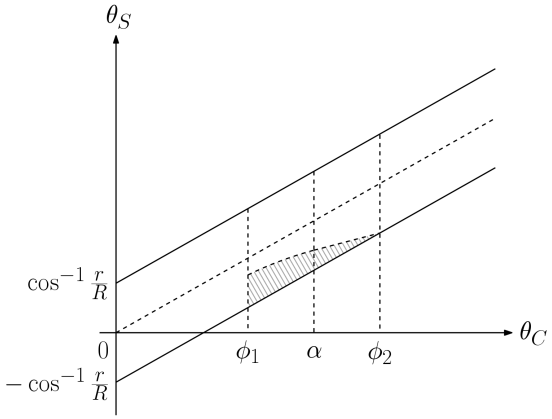


Figure 8 Infeasible space due to a line segment s outside C and inside C'.

7 John Hershberger. Finding the upper envelope of n line segments in $O(n \log n)$ time. *Information Processing Letters*, 33(4):169–174, 1989.

8 John Hopcroft, Deborah Joseph, and Sue Whitesides. Movement problems for 2-dimensional linkages. *SIAM Journal on Computing*, 13(3):610–629, 1984.

9 Steven M LaValle. *Planning algorithms*. Cambridge University Press, 2006.

10 Micha Sharir and Pankaj K Agarwal. *Davenport-Schinzel sequences and their geometric applications*. Cambridge University Press, 1995.

Improved Time-Space Bounds for Grid Graph Reachability

Rahul Jain[1] **and Raghunath Tewari**[2]

1 **Indian Institute of Technology Kanpur**
`jain@cse.iitk.ac.in`
2 **Indian Institute of Technology Kanpur**
`rtewari@cse.iitk.ac.in`

Abstract

The reachability problem is to determine if there exists a path from one vertex to the other in a graph. Grid graphs are the class of graphs where vertices are present on the lattice points of a two-dimensional grid, and an edge can occur between a vertex and its immediate horizontal or vertical neighbor only.

Asano et al. presented the first simultaneous time space bound for reachability in grid graphs with n vertices by presenting an algorithm that solves the problem in polynomial time and $O(n^{1/2+\epsilon})$ space [2]. In 2018, the space bound was improved to $\tilde{O}(n^{1/3})$ by Ashida and Nakagawa [4]. In this paper, we further improve the space bound and present a polynomial time algorithm that uses $O(n^{1/4+\epsilon})$ space to solve reachability in a grid graph.

1 Introduction

Given a graph G and two vertices s and t in it, the *reachability problem* seeks to answer whether there exists a path from s to t in G. It is of fundamental importance in the field of computer science. Not only is it used as a subroutine in many graph algorithms, but its study also gives insights into space bounded computations. Reachability in a directed graph is complete for the class of problems solvable by a nondeterministic Turing machine in logspace. A deterministic logspace algorithm for it would show NL to be equal to L, thus solving an open question in the area of computational complexity. In an undirected graph, reachability was shown to be in L by Reingold [11].

Standard graph traversal algorithms solve reachability in directed graphs using linear time and space. We also know of Savitch's algorithm which can solve reachability in $O(\log^2 n)$ space but the algorithm requires $2^{\Omega(\log^2 n)}$ time [12]. So, on one end we have algorithms that require a small amount of time and a large amount of space, while on the other end, we have Savitch's algorithm which requires a small amount of space but a large amount of time. A natural question we can ask is whether there exists an algorithm that uses time and space both in small amounts. This question was formally asked by Wigderson in his survey of reachability problems, if there exists an algorithm for graph reachability which maintains the polynomial time bound while running in $O(n^\epsilon)$ space, for some $\epsilon < 1$ [13].

Barnes, Buss, Ruzzo, and Schieber gave the first sublinear space polynomial time algorithm for reachability in directed graph [5]. The space complexity of their algorithm is $n/2^{\Theta(\sqrt{\log n})}$. We know of polynomial time algorithms with better space complexity for various subclasses of directed graphs. These include planar graphs [9][3], genus g graphs, H-minor-free graphs, $K_{3,3}$-minor-free graphs, K_5-minor free graphs [7], Layered planar graphs [8] and Unique path graphs [10].

Our concern here is with grid graphs. Grid graphs are a subclass of planar graphs. Reachability in planar graphs belongs to a subclass of NL called unambiguous logspace UL [6]. Reachability in planar graphs can be reduced to reachability in grid graphs in logspace

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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]. Asano and Doerr presented a polynomial time algorithm that uses $O(n^{1/2+\epsilon})$ space for solving reachability in grid graphs [2]. Ashida and Nakagawa presented an algorithm with improved space complexity of $\tilde{O}(n^{1/3})$ [4]. Ashida and Nakagawa's algorithm proceeded by first dividing the input grid graph into subgrids. It then used a gadget to transform each subgrid into a planar graph, making the whole of the resultant graph planar. Finally, it used the planar reachability algorithm of Imai et al. [9] as a subroutine to get the desired space bound.

In this paper, we present an algorithm with a space complexity of $O(n^{1/4+\epsilon})$, thereby improving the bound of Ashida and Nakagawa.

▶ **Theorem 1.1.** *For every $\epsilon > 0$, there exists a polynomial time algorithm that decides reachability in grid graphs using $O(n^{1/4+\epsilon})$ space.*

Our algorithm works by first dividing the grid into subgrids. It then recursively solve each grid to get an *auxiliary graph*. It then solves this auxiliary graph by using a space efficient subroutine that we develop for it.

2 Preliminaries

We denote the vertex set of a graph G by $V(G)$ and its edge set by $E(G)$. For a subset U of $V(G)$, we denote the subgraph of G induced by the vertices of U as $G[U]$. For a graph G, we denote the set of all its connected components by $\mathsf{cc}(G)$. For an edge $e = (u, v)$, we let $\mathsf{tail}(e)$ be u and $\mathsf{head}(e)$ be v.

In a *drawing* of a graph on a plane, each vertex is mapped to a point of the plane, and each edge is mapped to a simple arc whose endpoints coincide with the mappings of the end vertices of the edge. Also, the interior of an arc for an edge does not contain any other vertex points.

We call a graph G an $N \times N$ grid graph if its vertices can be drawn on coordinates (i, j) where $0 \leq i, j \leq N$ and for all edges of G, their end vertices are at unit distance.

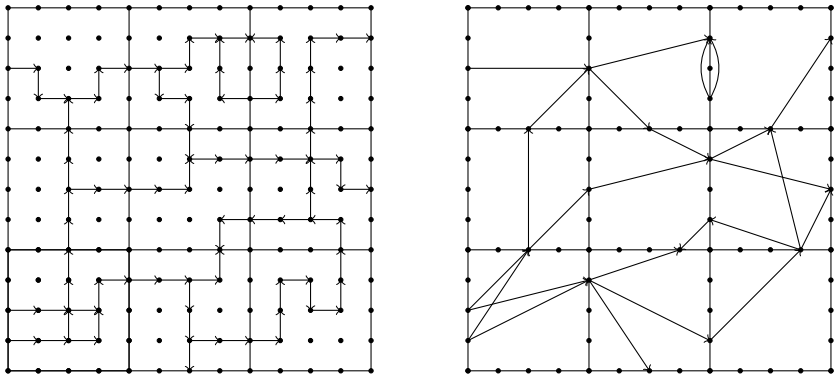
3 Main Result

For simplicity of discussion, we begin with an $N \times N$ grid graph and present a polynomial time algorithm with a space complexity $O(N^{1/2+\epsilon})$. When measured in terms of the number of vertices n of the graph, this space complexity translates to $O(n^{1/4+\epsilon})$.

3.1 Auxiliary Graph

For a parameter $\alpha < 1$, we first define the *α-auxiliary graph* G^α of a grid graph G. We divide our grid graph G into $N^{2\alpha}$ subgrids such that each subgrid is a $N^{1-\alpha} \times N^{1-\alpha}$ grid as shown in Figure 1*a*. Let $G^\alpha_{i,j}$ be the graph obtained by solving reachability in the subgrid in the i-th row and j-th column. We obtain G^α by replacing each subgrid by its corresponding solved blocks. Since each of the subgrids contains $4N^{1-\alpha}$ vertices on its boundary, the total number of vertices in G^α is at most $4N^{1+\alpha}$. An example of G^α is shown in Figure 1*b*. For the rest of this article, for any i and j, we will call the graph $G^\alpha_{i,j}$ a *block* of G^α. Note that G^α might have parallel edges. However, each such edge will belong to a different block of G^α.

Our algorithm for reachability constructs G^α by solving the $N^{1-\alpha} \times N^{1-\alpha}$ grids recursively. It then uses a polynomial time subroutine which solves G^α. Note that we do not store the graph G^α explicitly, since that would require too much space. Instead, we solve a subgrid recursively everytime the subroutine queries for an edge in that subgrid of G^α.



(a) Graph divided into subgrids **(b)** Auxiliary Graph

Figure 1 An example of a grid graph G and the corresponding auxiliary graph G^α

Our strategy is to show that for every small positive constant β, there exists a polynomial time algorithm which solves reachability in G^α using $\tilde{O}(\tilde{n}^{1/2+\beta/2})$ space where $\tilde{n}$ is the number of vertices in G^α. As discussed earlier, $\tilde{n}$ can be at most $4N^{1+\alpha}$. Hence, the main algorithm would require $\tilde{O}(N^{1/2+\beta/2+\alpha/2+\alpha\beta/2}) = O(N^{1/2+\epsilon})$ space.

3.2 Properties of the Auxiliary Graph

The auxiliary graph that we construct might not be planar; there can be edges that cross each other. However, since we construct it from a grid graph, we can make the following essential observations about the auxiliary graph G^α.

▶ **Observation 3.1.** *If two edges e and f of an auxiliary graph G^α cross each other, then the graph G also has the edges $(\mathsf{tail}(e), \mathsf{head}(f))$ and $(\mathsf{tail}(f), \mathsf{head}(e))$.*

▶ **Observation 3.2.** *If two edges e_1 and e_2 crosses a certain edge f, and e_1 is closer to $\mathsf{tail}(f)$ than e_2, then the edge $(\mathsf{tail}(e_1), \mathsf{head}(e_2))$ is also present in the graph G^α.*

3.3 Constructing a pseudoseparator

Imai et al. used a separator construction to solve the reachability problem in planar graphs [9]. A separator is a *small* set of vertices whose removal disconnects the graph into *smaller components*. An essential property of a separator is that, for any two vertices, a path between the vertices must contain a separator vertex if the vertices lie in two different components with respect to the separator.

Unfortunately the graph G^α might not have a small separator. However, G^α has a different kind of separator, which we call a pseudoseparator (see Definition 3.3). The pseudoseparator allows us to decide reachability in G^α, in an efficient manner and obtain the claimed bounds.

▶ **Definition 3.3.** Let G be a grid graph and H be a vertex-induced subgraph of G^α with h vertices. Let C be a subgraph of H and $H^{(C)}$ be the subgraph of H formed by removing all the vertices of C and all the edges which crosses an edge of C. Let $f : \mathbb{N} \to \mathbb{N}$ be a function. We call C an $f(h)$-pseudoseparator if the size of every connected component in $cc(H^{(C)})$ is at most $f(h)$. The size of C is the total number of vertices and edges of C summed together.

For a vertex-induced subgraph H of G^α, an $f(h)$-pseudoseparator is a subgraph of H that has the property that, if we remove the vertices as well as all the edges which cross

one of the edges of pseudoseparator, the graph gets disconnected into small pieces. Now, any path which connects two vertices in different components, must either contain a vertex of pseudoseparator or must contain an edge that crosses an edge of pseudoseparator (see Observation 3.4). We divide the graph using this pseudoseparator and show an algorithm which recursively solves each subgraph and then combines their solution efficiently using the above observations.

▶ **Observation 3.4.** *Let G be a grid graph and let H be a vertex-induced subgraph of G^α. Let C be a subgraph of H. Then the following holds:*

1. *For any two distinct U_1 and U_2 of $\mathsf{cc}(H^{(C)})$, $U_1 \cap U_2 = \emptyset$.*
2. *$V(C) \cup (\bigcup_{U \in \mathsf{cc}(H^{(C)})} U) = V(H)$*
3. *For every edge e in H, if there exist distinct sets U_1 and U_2 in $\mathsf{cc}(H^{(C)})$ such that one of the endpoints of e is in U and the other is in U_2, then there exists an edge f in C such that e crosses f.*

We construct a pseudoseparator using the next lemma.

▶ **Lemma 3.5.** *Let G be a grid graph, and let H be a vertex-induced subgraph of G^α with h vertices. For any constant $\beta > 0$, there exists an algorithm which takes H as input and outputs an $(h^{1-\beta})$-pseudoseparator of H of size $O(h^{1/2+\beta/2})$ in $\tilde{O}(h^{1/2+\beta/2})$ space and polynomial time.*

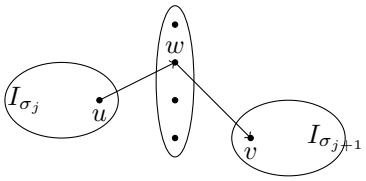
We briefly comment on how to construct a pseudoseparator of a vertex-induced subgraph H of G^α. First, we pick, in logspace, a maximal subset of edges from H so that no two edges *cross*. Then, we triangulate the resulting graph and use Imai et al.'s algorithm to find its separator. Call the triangulated graph $\widehat{H}$ and the set of separator vertices S. The vertex set of pseudoseparator of H contains all the vertices of S and four additional vertices for each edge of $\widehat{H}[S]$ that is not present in H. The edge set of pseudoseparator of H contains all edges of H which are also in $\widehat{H}[S]$ and four additional edges for each edge of $\widehat{H}[S]$ that is not present in H.

3.4 Sketch of an Algorithm to Solve Reachability in the Auxiliary Graph

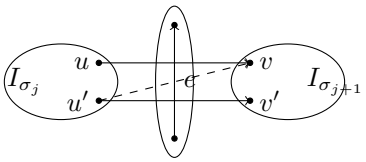
Given a vertex-induced subgraph H of G^α, we first construct its pseudoseparator using Lemma 3.5. Call this pseudoseparator C. We ensure that s and t are part of the pseudoseparator. Let $I_1, I_2, \ldots, I_l$ be the components received after dividing the graph using pseudoseparator. The subroutine performs a loop with $|H|$ iterations and updates a set of marked vertices. Initially, it marks the vertex s. After an iteration, it marks a vertex of C if there is a path from a marked vertex to it such that the internal vertices of that path all belong to only one of the components I_i. Also, for each edge e of C, the vertex v closest to $\mathsf{tail}(e)$ which satisfies the following two conditions is marked: (i) There exists an edge f which cross e and $\mathsf{tail}(f) = v$, and (ii) there is a path from a marked vertex to v such that the internal vertices of the path all belong to only one of the components I_i.

Let P be the shortest path from s to t in H. Suppose P passes through the components $I_{\sigma_1}, I_{\sigma_2}, \ldots, I_{\sigma_L}$ in this order. The length of this sequence can be at most $|H|$. As the path leaves the component I_{σ_j} and goes into $I_{\sigma_{j+1}}$, it can do this in the following two ways:

1. The path exits I_{σ_j} through a vertex of pseudoseparator as shown in Figure 2a. In this case, our algorithm would mark the vertex w.
2. The path exits I_{σ_j} through an edge (u, v) whose other endpoint is in $I_{\sigma_{j+1}}$. By Observation 3.4, this edge will cross an edge e of the pseudoseparator. In this case, the algorithm



(a) The s-t path contains a vertex of the separator



(b) The s-t path crosses an edge of the separator

Figure 2 Types of crossing of an s-t path with the separator.

would mark the vertex u' which is closer than u to $\mathsf{tail}(e)$ and an edge (u', v') crosses e. By Observation 3.2, the edge (u', v) would be present in the graph.

Thus after j iteration, the subroutine would traverse the fragment of the path in the component I_{σ_j} and either mark its endpoint or a vertex which is closer to the edge e of C which the path crosses. Finally, t will be marked after L iterations if and only if there is a path from s to t in H.

3.5 Complexity of the Algorithm

Our subroutine solves reachability in a subgraph H (having size h) of G^α. We do not explicitly store a component of $\mathsf{cc}(H^{(C)})$, since it might be too large. Instead, we identify a component with the lowest indexed vertex present in it and use Reingold's algorithm on $H^{(C)}$ to determine if a vertex is present in that component. We require $\tilde{O}(h^{1/2+\beta/2})$ space to calculate pseudoseparator by Lemma 3.5. We can potentially mark all vertices of pseudoseparator and for each edge of pseudoseparator we mark at most one additional vertex. Since the size of pseudoseparator is at most $O(h^{1/2+\beta/2})$, we require $\tilde{O}(h^{1/2+\beta/2})$ space. The algorithm recurses on a graph with $h^{1-\beta}$ vertices. Hence the depth of the recursion is $1/\beta$, which is a constant. The total space complexity is thus $\tilde{O}(\tilde{n}^{1/2+\beta/2})$.

Since the graph H is given implicitly in our algorithm, there is an additional polynomial overhead involved in obtaining its vertices and edges. However, the total time complexity remains a polynomial in the number of vertices since the recursion depth is constant.

References

1 Eric Allender, David A Mix Barrington, Tanmoy Chakraborty, Samir Datta, and Sambuddha Roy. Planar and grid graph reachability problems. *Theory of Computing Systems*, 45(4):675–723, 2009.

2 Tetsuo Asano and Benjamin Doerr. Memory-constrained algorithms for shortest path problem. In *Proceedings of the 23rd Annual Canadian Conference on Computational Geometry (CCCG 2011)*, 2011.

3 Tetsuo Asano, David Kirkpatrick, Kotaro Nakagawa, and Osamu Watanabe. $\tilde{O}(\sqrt{n})$-space and polynomial-time algorithm for planar directed graph reachability. In *Proceedings of the 39th International Symposium on Mathematical Foundations of Computer Science (MFCS 2014)*, pages 45–56, 2014.

4 Ryo Ashida and Kotaro Nakagawa. $\tilde{O}(n^{1/3})$-space algorithm for the grid graph reachability problem. In *Proceedings of the 34th International Symposium on Computational Geometry (SoCG 2018)*, pages 5:1–5:13, 2018.

5 Greg Barnes, Jonathan F. Buss, Walter L. Ruzzo, and Baruch Schieber. A sublinear space, polynomial time algorithm for directed s-t connectivity. *SIAM Journal on Computing*, 27(5):1273–1282, 1998.

6 Chris Bourke, Raghunath Tewari, and NV Vinodchandran. Directed planar reachability is in unambiguous log-space. *ACM Transactions on Computation Theory (TOCT)*, 1(1):4, 2009.

7 Diptarka Chakraborty, Aduri Pavan, Raghunath Tewari, N. V. Vinodchandran, and Lin F. Yang. New time-space upperbounds for directed reachability in high-genus and h-minor-free graphs. In *Proceedings of the 34th Annual Conference on Foundation of Software Technology and Theoretical Computer Science (FSTTCS 2014)*, pages 585–595, 2014.

8 Diptarka Chakraborty and Raghunath Tewari. An $O(n^{\epsilon})$ space and polynomial time algorithm for reachability in directed layered planar graphs. *ACM Transactions on Computation Theory (TOCT)*, 9(4):19:1–19:11, 2017.

9 Tatsuya Imai, Kotaro Nakagawa, Aduri Pavan, N. V. Vinodchandran, and Osamu Watanabe. An $O(n^{\frac{1}{2}+\epsilon})$-space and polynomial-time algorithm for directed planar reachability. In *Proceedings of the 28th Conference on Computational Complexity (CCC 2013)*, pages 277–286, 2013.

10 Sampath Kannan, Sanjeev Khanna, and Sudeepa Roy. STCON in Directed Unique-Path Graphs. In *Proceedings of the 28th Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2008)*, volume 2, pages 256–267, Dagstuhl, Germany, 2008. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

11 Omer Reingold. Undirected connectivity in log-space. *Journal of the ACM (JACM)*, 55(4):17, 2008.

12 Walter J Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970.

13 Avi Wigderson. The complexity of graph connectivity. In *Proceedings of the 17th International Symposium on Mathematical Foundations of Computer Science (MFCS 1992)*, pages 112–132. Springer, 1992.

Smoothed Analysis of the Art Gallery Problem

Michael Gene Dobbins[1], Andreas Holmsen[2], and Tillmann Miltzow[3]

1 **Department of Mathematical Sciences, Binghamton University**
2 **Department of Mathematical Sciences, KAIST**
3 **Department of Information and Computing Sciences, Utrecht University**

Abstract

In the Art Gallery Problem we are given a polygon P on n vertices and a number k. We want to find a guard set G of size k, such that each point in P is *seen* by a guard in G. Formally, a guard g sees a point $p \in P$ if the line segment pg is fully contained inside P.

We analyze the Art Gallery Problem under the lens of Smoothed Analysis. The significance of our results is that algebraic methods are not needed to solve the Art Gallery Problem in *typical* instances. This is the first time an $\exists\mathbb{R}$-complete problem was analyzed by Smoothed Analysis. Details can be found in the full-version [14].

A short video explaining the result is available at `youtu.be/Axs7k-qL2zY`.

1 Introduction

In the Art Gallery Problem we are given a polygon P and a number k. We want to find a guard set G of size k, such that each point in P is *seen* by a guard in G. Formally, a guard g sees a point $p \in P$ if the line segment pg is fully contained inside P. We usually denote the vertices of P by $v_1, \ldots, v_n$, and the number of vertices by n.

One of the most fundamental questions on the Art Gallery Problem is whether it is contained in the complexity-class NP. A first doubt of NP-membership was raised in 2017, when Abrahamsen, Adamaszek and Miltzow showed that there exist polygons with vertices given by integer coordinates, that can be guarded by three guards, in which case some guards must necessarily have irrational coordinates [1]. (It is an open problem whether irrational guards may be required for polygons which can be guarded by two guards.) Shortly after, the same authors could show that the Art Gallery Problem is complete for the complexity class $\exists\mathbb{R}$ [2].

The class $\exists\mathbb{R}$ is the class of all decision problems that are many-one reducible in polynomial time to deciding whether a given polynomial $Q \in \mathbb{Z}[x_1, \ldots, x_n]$ has a real root, i.e. a solution $x \in \mathbb{R}^n$ such that $Q(x) = 0$. From the field of real algebraic geometry [4], we know that

$$\text{NP} \subseteq \exists\mathbb{R} \subseteq \text{PSPACE}.$$

The complexity class $\exists\mathbb{R}$ provides a tool to give much more compelling arguments that a problem may not lie in NP than merely observing that the naive way of placing the problem into NP does not work. Indeed various problems have been shown to be $\exists\mathbb{R}$-complete [8, 9, 13, 16, 17, 20–23] and thus either non of them lie in NP or all of them do.

While those theoretical results on the Art Gallery Problem are quite negative, the history and practical experiences tell a more positive story. First of all, it took more than four decades before an example could be found that requires irrational guards [1]. Regarding the practical study of the Art Gallery Problem, we want to point out that several researchers have implemented heuristics, that were capable of finding optimal solutions for a large class of simulated instances [3, 5–7, 10–12, 15, 19]. Even up to 5000 vertices.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

$\exists\mathbb{R}$-completeness is our main motivation, to see if there is a *simple* algorithm that solves the Art Gallery Problem. As we don't expect that such an algorithm is correct in the worst case, we turn our attention to different ways to analyze algorithms.

Smoothed Analysis

Some algorithms perform much better than predicted by their worst case analysis. The most famous example seems to be the Simplex-Algorithm. It is an algorithm that solves linear programming efficiently in practice, although it is known that there are instances for seemingly all variants of the algorithm that take an exponential amount of time (see for instance [18]). There are several possible ways to explain this behavior. For example, it could be that all practical instances have some structural properties, which we have not yet discovered. We could imagine that a more clever analysis of the Simplex-Algorithm would yield that it runs in polynomial time, assuming the property is presented. To the best of our knowledge such a property has not yet been identified. Another approach would be to argue that worst case examples are just very "rare in practice". The problem with this approach is that it is difficult to formalize. Smoothed Analysis is a nice combination of the average case and the worst case analysis and generally referred to as Smoothed Analysis, as it *smoothly* interpolates between the two. It was developed by Spielman and Teng [24], who introduced the field in their celebrated seminal paper "Smoothed Analysis of algorithms: Why the simplex algorithm usually takes polynomial time". Both authors received the Gödel Prize in 2008, and the paper was one of the winners of the Fulkerson Prize in 2009. In 2010 Spielman received the Nevanlinna Prize for developing Smoothed Analysis.

The rough idea is to take the worst instance and perturb it slightly in a random way. The smoothed expected running time can be defined as follows: Let us fix some δ, which describes the maximum magnitude of perturbation. We denote by $(\Omega_\delta, \mu_\delta)$ a corresponding probability space where each $x \in \Omega_\delta$ defines for each instance I a new 'perturbed' instance I_x. We denote by $T(I_x)$, the time to solve the instance I_x. Now the smoothed expected running time of instance I equals

$$T_\delta(I) = \mathop{\mathbb{E}}_{x\in\Omega_\delta} T(I_x) = \int_{x\in\Omega_\delta} T(x)\mu_\delta(x).$$

If we denote by Γ_n the set of instances of size n, then the smoothed running time equals:

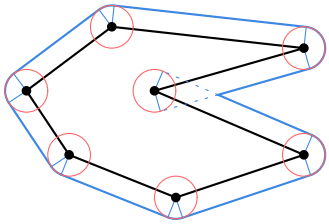
$$T_{\text{smooth}}(n) = \max_{I\in\Gamma_n} \mathop{\mathbb{E}}_{x\in\Omega_\delta} T(I_x).$$

Roughly speaking this can be interpreted as saying, that not only do the majority of instances have to behave nicely, but actually in every neighborhood the majority of instances behave nicely. The expected running time is measured in terms of n and δ. If the expected running time is small in terms of $1/\delta$ then this means that difficult instances are *fragile* with respect to perturbations. This serves as theoretical explanation why such instances may not appear in practice.

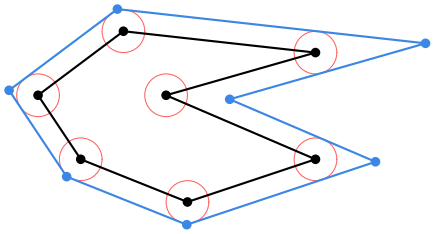
Although the concept of Smoothed Analysis is more complicated than simple worst case analysis, it is a new success story in theoretical computer science. It could be shown that various algorithms actually run in smoothed polynomial time, explaining very well their practical performance.

1.1 Defintions

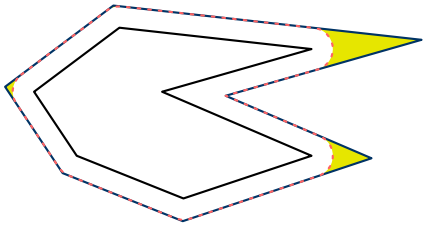
The different models of perturbation are illustrated in Figure 1. A rigorous definition can be found in the full-version of the paper [14].



(a) The Minkowski-sum of a polygon together with a disk. This is also called a Minkowski-Inflation.



(b) The polygon together with an Edge-Inflation. Roughly speaking, every edge of the polygon is "pushed" to the outside by the same amount.



(c) If we continue the edges, of a Minkowski-Inflation, we get an Edge-Inflation.

Figure 1 Overview, over various models, how a polygon can be perturbed. We use the uniform distribution in each case.

1.2 Results

Our main result states that typical instances do not require irrational guards and the expected number of bits per guard is logarithmic in L, δ and n. The result establishes that algebraic methods are *not* needed in typical instances.

▶ **Theorem 1.1** (Bit-complexity). *Let P be a polygon on n vertices, suppose $P \subset [0, L]^2$ for some positive integer L. If $\delta > 0$ is the magnitude of a Minkowski-Inflation or Edge-Inflation, then the expected number of bits per guard to describe an optimal solution equals $O\left(\log\left(\frac{nL}{\delta}\right)\right)$.*

As a simple corollary of the proof, we get that a fine grid of expected width $w = 2^{O(\log(nL/\delta))} = (nL/\delta)^{O(1)}$ will contain an optimal guarding set. This may appear at first sight as a *candidate set* of polynomial size, however recall that the vertices are given in binary and thus L may be exponential in the input size.

It can be argued (see the full-version [14]) that this also leads to expected NP algorithms in a specific sense. A very careful discussion of the different models of computation is needed [14] to make the above statement precise. Our results can also be extended to other types of perturbations [14].

Notation

We write $f(n) \leq_c g(n)$, to indicate $f(n) = O(g(n))$ or equivalently $f(n) \leq cg(n)$, for some large enough constant c. (Note that this is, in turn, equivalent to $f(n) \leq c_1 g(n) + c_2$. To see this note that $g(n) \geq 1$ and choose $c = c_1 + c_2$.)

2 Preliminaries

In this section we establish some general facts that will be needed throughout the paper.

The key idea of the paper are some monotonicity properties of Minkowski-Inflation and Edge-Inflation. Roughly speaking guarding can only get easier after inflations. (We denote by $OPT(P)$ a guarding set of P of optimal size. We denote by $OPT(P, C)$ a guarding set of P of optimal size, when restricted to the set C.)

▶ **Lemma 2.1** (Fixed Minkowski-Inflation). *Let P be a polygon, $t > 0$ and P_t its Minkowski-Inflation by magnitude t. Then $|OPT(P)| \geq |OPT(P_t, w\mathbb{Z}^2)|$, for any $w \leq \sqrt{2}t$.*

Proof. Given $OPT = OPT(P)$, we define a set $G \subseteq w\mathbb{Z}^2$ of guards of size $|G| = |OPT|$, by rounding every point in OPT to its closest grid point in $w\mathbb{Z}^2$. We will show that G guards P_t. See Figure 2 for an illustration.

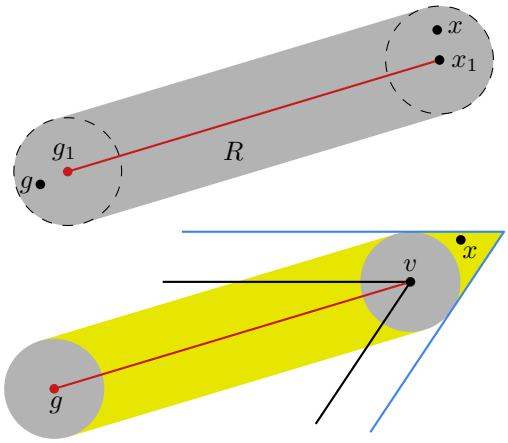


Figure 2 Top: The Region R is convex, and contains a guard $g \in G$ and the point x. Thus x is guarded by g. Bottom: The Region R' is easily seen to be convex.

Let us fix some arbitrary point $x \in P_t$. It is sufficient to show that G guards x. By definition of P_t, there exists an $x_1 \in P$ and an $x_2 \in \text{disk}(t)$ such that $x = x_1 + x_2$. Furthermore let g_1 be a guard of OPT that guards x_1. (disk(t) is a disk of radius t.) Consider the region $R = g_1x_1 \oplus \text{disk}(t)$, i.e., the Minkowski-sum of the segment g_1x_1 with a disk of radius t. As the segment g_1x_1 is contained in P, it holds that R is contained in P_t. Also as both the segment and the disk are convex, so is R. At last notice that R contains a point $g \in G$, as every disk of radius t contains a point of the grid $w\mathbb{Z}^2$ with $w = \sqrt{2}t$. As R is convex, $g \in G$ guards x. ◀

▶ **Lemma 2.2** (Fixed Edge Inflation). *Let P be a polygon with integer coordinates and $t > 0$ and P_t the Edge-Inflation of P by t. Then $|OPT(P)| \geq |OPT(P_t, w\mathbb{Z}^2)|$, for any $w \leq \sqrt{2}t$.*

Proof. We follow closely the proof of Lemma 2.1. See Figure 2 for an illustration.

Given $OPT = OPT(P)$, we define a set $G \subseteq w\mathbb{Z}^2$ of guards of size $|G| = |OPT|$, by rounding every point in OPT to its closest grid point in $w\mathbb{Z}^2$. We will show that G guards the shape P_t. Note that in an edge inflation by t, we get the same shape as by a Minkowski-Inflation by t, except that we have to add some small regions at the convex corners, as illustrated in Figure 1c. We already know that G guards the Minkowski t-inflation of P. So it remains to show that G guards those little extra regions, as discussed above.

Let us fix some arbitrary point $x \in P_t$ inside one of those extra regions. We will show that G guards x. Let v be the vertex according to the region that x sits in. Furthermore let g_1 be a guard of OPT that guards v. Consider the region $R = g_1 v \oplus \text{disk}(0, t)$. We define R' as the region R together with the region that x sits in. Obviously $x \in R'$ and also there exists a point of G in R. It holds by construction that R' is convex. This finishes the proof. ◀

3 Expected Number of Bits

This section is devoted to show the main theorem.

Proof. Let us assume that there are some numbers $0 = t_0 < t_1 < \ldots < t_\ell = \delta$ such that for all i and $s \in [t_{i-1}, t_i)$ holds that $|OPT(P_s)|$ is constant. As $|OPT(P_s)|$ is monotonically decreasing, for increasing s, it holds that $\ell \leq n$. We denote by $\delta_i = t_i - t_{i-1}$.

Note that if the perturbation happens to be $s \in [t_{i-1}, t_i]$ then a grid of width $w = \sqrt{2}(s - t_{i-1})$ contains an optimal solution to guard P_s, see Lemma 2.1 and 2.2. Note that we use the lemmas on the shape $P_{t_{i-1}}$ inflated by $s - t_{i-1}$. Then the number of bits per guard to describe the solution equals $O(\log(L/w))$ per guard. To see this note that we can use $b = \lceil 1/w \rceil$ as denominator of all coordinates and the numerators are upper bounded by $\lceil L/w \rceil$. Thus $O(\log(L/w))$ bits suffice. Let us denote the number of bits after a perturbation by s as $B(s)$. We denote by $\mathbb{E}(B_i)$ the expected number of bits for $s \in [t_{i-1}, t_i)$. The expected number of bits $\mathbb{E}(B_i)$ can be calculated as

$$\mathbb{E}(B_i) = \frac{1}{\delta_i} \int_{s \in [t_{i-1}, t_i)} B(s) \leq_c \frac{1}{\delta_i} \int_{s \in [t_{i-1}, t_i)} \log(L/(s - t_{i-1})) = \frac{1}{\delta_i} \int_{s \in [0, \delta_i]} \log(L/s).$$

Using some computer algebra system and concavity of $\log(1/x)$, we get

$$= \frac{1}{\delta_i} \delta_i \left(1 + \log(L/\delta_i)\right) \leq_c \log(L/\delta_i).$$

We are now ready to compute $\mathbb{E}(B)$.

$$\mathbb{E}(B) = \frac{1}{\delta} \sum_{i=1,\ldots,\ell} \delta_i \mathbb{E}(B_i) \leq_c \frac{1}{\delta} \sum_{i=1,\ldots,\ell} \delta_i \log(L/\delta_i).$$

As the function $x \log(1/x)$ is concave the maximum is attained, if $\delta_1 = \ldots = \delta_\ell = \delta/\ell$. Thus we get

$$\mathbb{E}(B) \leq_c \frac{1}{\delta} \sum_{i=1,\ldots,\ell} \delta/\ell \log(L\ell/\delta) = \log(L\ell/\delta) \leq_c \log(Ln/\delta).$$ ◀

Acknowledgments. We want to thank Stefan Langerman, Jean Cardinal, John Iacono and Mikkel Abrahamsen for helpful discussion on the presentation of the results. We want to thank Joseph O'Rourke for pointing us to an algorithm to check if a given set of guard positions is correct. Part of the research was conducted while visiting KAIST, and we

thank KAIST and BK21 for their support and for providing an excellent working environment. Andreas Holmsen was supported by the Basic Science research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2016R1D1A1B03930998). Tillmann Miltzow acknowledges the generous support from the ERC Consolidator Grant 615640-ForEFront and the Veni EAGER.

References

1 Mikkel Abrahamsen, Anna Adamaszek, and Tillmann Miltzow. Irrational guards are sometimes needed. In *SoCG 2017*, pages 3:1–3:15, 2017. Arxiv 1701.05475.

2 Mikkel Abrahamsen, Anna Adamaszek, and Tillmann Miltzow. The art gallery problem is $\exists\mathbb{R}$-complete. In *STOC 2018*, pages 65–73, 2018. Arxiv 1704.06969. URL: `http://doi.acm.org/10.1145/3188745.3188868`, `doi:10.1145/3188745.3188868`.

3 Yoav Amit, Joseph S.B. Mitchell, and Eli Packer. Locating guards for visibility coverage of polygons. *International Journal of Computational Geometry & Applications*, 20(05):601–630, 2010.

4 Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in real algebraic geometry*. Springer-Verlag Berlin Heidelberg, 2006.

5 Dorit Borrmann, Pedro J. de Rezende, Cid C. de Souza, Sándor P. Fekete, Stephan Friedrichs, Alexander Kröller, Andreas Nüchter, Christiane Schmidt, and Davi C. Tozoni. Point guards and point clouds: Solving general art gallery problems. In *SoCG 2013*, pages 347–348, 2013. URL: `http://doi.acm.org/10.1145/2462356.2462361`, `doi:10.1145/2462356.2462361`.

6 Andrea Bottino and Aldo Laurentini. A nearly optimal sensor placement algorithm for boundary coverage. *Pattern Recognition*, 41(11):3343–3355, 2008.

7 Andrea Bottino and Aldo Laurentini. A nearly optimal algorithm for covering the interior of an art gallery. *Pattern recognition*, 44(5):1048–1056, 2011.

8 Jean Cardinal, Stefan Felsner, Tillmann Miltzow, Casey Tompkins, and Birgit Vogtenhuber. Intersection graphs of rays and grounded segments. *Journal of Graph Algorithms and Applications*, 22:273–295, 2018.

9 Jean Cardinal and Udo Hoffmann. Recognition and complexity of point visibility graphs. *Discrete & Computational Geometry*, 57(1):164–178, 2017.

10 Marcelo C. Couto, Pedro J. de Rezende, and Cid C. de Souza. An exact algorithm for minimizing vertex guards on art galleries. *International Transactions in Operational Research*, 18(4):425–448, 2011.

11 Marcelo C. Couto, Cid C. De Souza, and Pedro J. De Rezende. Experimental evaluation of an exact algorithm for the orthogonal art gallery problem. In *International Workshop on Experimental and Efficient Algorithms*, pages 101–113. Springer, 2008.

12 Pedro J. de Rezende, Cid C. de Souza, Stephan Friedrichs, Michael Hemmer, Alexander Kröller, and Davi C. Tozoni. Engineering art galleries. In Peter Sanders Lasse Kliemann, editor, *Algorithm Engineering – Selected Results and Surveys*, pages 379–417. Springer International Publishing, 2016. URL: `http://dx.doi.org/10.1007/978-3-319-49487-6_12`, `doi:10.1007/978-3-319-49487-6_12`.

13 Michael G. Dobbins, Linda Kleist, Tillmann Miltzow, and Paweł Rzążewski. $\forall\exists\mathbb{R}$-completeness and area-universality. *WG 2018*, 2018. Arxiv 1712.05142.

14 Michael Gene Dobbins, Andreas Holmsen, and Tillmann Miltzow. Smoothed analysis of the art gallery problem. *CoRR*, abs/1811.01177, 2018. URL: `http://arxiv.org/abs/1811.01177`, `arXiv:1811.01177`.

15 S. Friedrichs. Integer solutions for the art gallery problem using linear programming. Masterthesis, 2012.

16 Jugal Garg, Ruta Mehta, Vijay V. Vazirani, and Sadra Yazdanbod. ETR-completeness for decision versions of multi-player (symmetric) Nash equilibria. In *ICALP 2015*, pages 554–566, 2015.

17 Ross J. Kang and Tobias Müller. Sphere and dot product representations of graphs. In *SoCG*, pages 308–314. ACM, 2011.

18 Victor Klee and George J. Minty. How good is the simplex algorithm. Technical report, Washington Univ. Seattle Dept. of Mathematics, 1970.

19 Alexander Kröller, Tobias Baumgartner, Sándor P Fekete, and Christiane Schmidt. Exact solutions and bounds for general art gallery problems. *Journal of Experimental Algorithmics (JEA)*, 17:2–3, 2012.

20 Anna Lubiw, Tillmann Miltzow, and Debajyoti Mondal. The complexity of drawing a graph in a polygonal region. *Arxiv*, 2018. Graph Drawing 2018.

21 Jürgen Richter-Gebert and Günter M. Ziegler. Realization spaces of 4-polytopes are universal. *Bulletin of the American Mathematical Society*, 32(4):403–412, 1995.

22 Marcus Schaefer. Realizability of graphs and linkages. In *Thirty Essays on Geometric Graph Theory*, pages 461–482. Springer, 2013.

23 Yaroslav Shitov. A universality theorem for nonnegative matrix factorizations. *Arxiv 1606.09068*, 2016.

24 Daniel A. 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.

On Plane Subgraphs of Complete Topological Drawings

Alfredo García[*1], Alexander Pilz[†2], and Javier Tejel[‡1]

1 **Departamento de Métodos Estadísticos and IUMA, Universidad de Zaragoza.**
olaverri@unizar.es, jtejel@unizar.es
2 **Institute of Software Technology, Graz University of Technology.**
apilz@ist.tugraz.at

Abstract

We consider plane subgraphs of simple topological drawings of K_n, and in particular maximal ones. Fulek and Ruiz-Vargas showed that between any plane connected subgraph F and a vertex v not in F, there are two edges from v to F not crossing F. We give an $O(n)$ time algorithm to find such edges, and show that the result also holds if F is disconnected. In particular, any plane subgraph can be augmented to a 2-connected one. This leads to our main structural result, showing that maximal plane subgraphs are 2-connected and what we call *essentially 3-edge-connected*.

1 Introduction

In a *topological drawing* (in the plane or on the sphere) of a graph, vertices are represented by points and edges are arcs with its two vertices as endpoints. It is *simple* if two edges intersect at most in a single point, either at a common endpoint or at a crossing in their relative interior. Let D_n be a simple topological drawing of the complete graph K_n on n vertices. Clearly, all straight-line drawings are simple topological drawings, and thus problems on embedding graphs on a set of points usually generalize to finding subgraphs of D_n. Such problems are often concerned with crossing-free (i.e., *plane*) subgraphs. (Herein, we consider graphs in connection with their drawings, and in particular when addressing subgraphs of K_n we also consider the associated sub-drawing of D_n.) Crossing-free edge sets in D_n have attracted considerable attention. Pach, Solymosi, and Tóth [4] showed that any D_n has $\Omega\left(\log^{1/6}(n)\right)$ pairwise disjoint edges. This bound was subsequently improved [5, 1, 8]. The current best bound of $\Omega(n^{1/2-\epsilon})$ is by Ruiz-Vargas [7]. In the course of their work on disjoint edges and empty triangles in D_n, Fulek and Ruiz-Vargas [2] showed the following lemma.

▶ **Lemma 1.1** (Fulek and Ruiz-Vargas [2]). *Between any plane connected subgraph F of D_n and a vertex v not in F, there exist at least two edges from v to F that do not cross F.*

In this work, we show that such edges incident to v can be found in $O(n)$ time. Further, we extend their result to disconnected plane subgraphs. It turns out that any plane subgraph of D_n can be augmented to a 2-connected plane subraph of D_n. Maximal plane subgraphs of D_n have further interesting properties. For example, we show that, when removing two edges, they either stay connected or one of the two components is a single vertex.

[*] Supported by MINECO project MTM2015-63791-R and Gobierno de Aragón under Grant E41-17R.
[†] Supported by a Schrödinger fellowship of the Austrian Science Fund (FWF): J-3847-N35.
[‡] Supported by MINECO project MTM2015-63791-R and Gobierno de Aragón under Grant E41-17R.

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.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 19–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

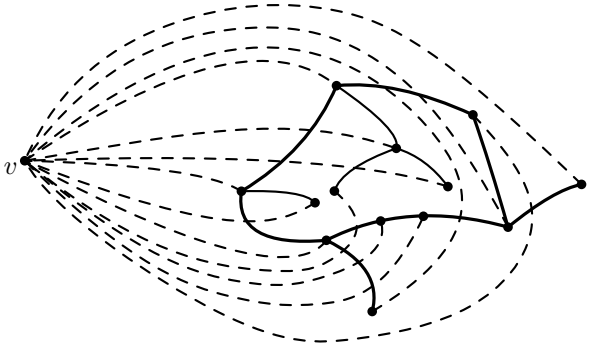


Figure 1 The order of the first intersections of $S(v)$ along the face matches the rotation of v.

2 Adding a single vertex

Lemma 1.1 turns out to be a useful workhorse for identifying plane subgraphs. We provide an efficient algorithm for computing the uncrossed edges. We assume that a topological drawing D_n of K_n with vertex set $V = \{v_1, \ldots, v_n\}$ is given by its rotation system and the inverse rotation system: The *rotation* of a vertex $v_i \in V$ is a permutation of $V \setminus \{v_i\}$ given by the circular order in which the edges to all other vertices emanate from v_i. We denote these edges by $S(v)$, i.e., the *star* with center v. The *inverse rotation system* that, for each vertex v_i and each index $j \neq i$ gives the index of v_j in the (linearized) rotation of v_i. (It can be obtained from the rotation system in $O(n^2)$ time.) Using these two structures, it is well-known that one can determine whether two edges cross, in which direction an edge is crossed, and in which order two non-crossing edges cross a third one in constant time [3].

▶ **Theorem 2.1.** *Given a simple topological drawing of K_n, a connected plane subgraph F, and a vertex v not in F, we can find the edges from v to F not crossing F in $O(n)$ time.*

Proof. W.l.o.g., let the face that contains v be the outer face f. For each edge from v to F, consider its first intersection with F. The order in which these points are encountered in a clockwise walk of the boundary of f matches the rotation of v (restricted to F); see Figure 1. (Edges walked twice can be seen as two "half-edges", essentially treating f as a cycle.) The algorithm starts by finding, for any edge vw_1, the edge of F that intersects vw_1 closest to v along vw_1. Using the rotation system and its inverse, this edge e_1 of face f can be found in linear time. We keep uncrossed edges of $S(v)$ on a stack σ and walk the boundary of f and the rotation of v, in each step making progress in at least one of them or removing an edge from σ. Let vw_i be the next edge in the counterclockwise rotation of v (initially vw_2). If vw_i crosses e_k (initially e_1), we iterate considering the counterclockwise successor vw_{i+1} of vw_i in the rotation of v. If there is no intersection of vw_i and e_k, we iterate with the clockwise successor e_{k+1} of e_k in f instead of e_k, after popping all edges from σ that cross e_{k+1}. We do the same if w_i is the clockwise endpoint of e_k along f; in addition, if vw_i is clockwise between e_k and e_{k+1} in the rotation of w_i, we put vw_i on σ and in the next iteration consider vw_{i+1}. Note that in a generic step σ contains the explored edges of $S(v)$ uncrossed by the explored edges $e_1, \ldots, e_{k-1}$ of f. Eventually, σ contains the uncrossed edges of $S(v)$. ◀

We now discuss an extension of Lemma 1.1 to plane subgraphs. This will also follow independently from Theorem 3.1. Still, our result (proven in the full version) gives further insight on the position of the edges in the rotation of the additional vertex v.

Let F be a plane subgraph of D_n. The edges of star $S(v)$ are called *rays*. Suppose that ray vr first crosses the edge pq of F. W.l.o.g., we suppose that the rays vr, vp, and vq appear in this clockwise order around v. Let x be the crossing of ray vr and edge pq. We define the clockwise range R_{cw} of rays centered at v corresponding to crossing x: if no ray in the range (vp, vq) crosses edge pq between x and p, then R_{cw} is the set of rays in the clockwise range $(vr, vp]$ (i.e., including vp but not vr); otherwise, if some rays in the clockwise range (vp, vq) cross edge pq between x and p, then R_{cw} is the set of rays in clockwise range (vr, vl), where vl is the ray in the range (vp, vq) crossing edge e between x and p in a point y closest to x along pq. See Figure 2, which also indicates the analogous definition of the counterclockwise range R_{ccw}.

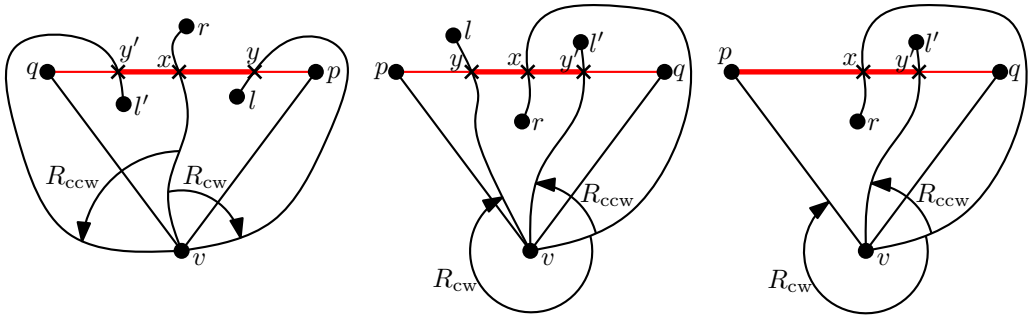


Figure 2 The clockwise and counterclockwise ranges of a first crossing.

▶ **Proposition 2.2.** *Suppose ray vr first crosses edge e of F at point x. Let R_{cw} and R_{ccw} be the ranges of rays of v corresponding to that crossing. Then, each one of those two ranges contains an uncrossed ray. As a consequence, $S(v)$ contains at least two uncrossed rays.*

3 Structure of maximal plane subgraphs

▶ **Theorem 3.1.** *A maximal plane subgraph of any D_n is spanning and 2-connected.*

Proof. The proof is by induction on n. The result is obviously true for $n \leq 3$. For $n > 3$, suppose there exists a maximal plane subgraph $\bar{F}$ that is not 2-connected.

We first argue that, under this assumption, $\bar{F}$ does not have a vertex v of degree less than 3. Suppose the contrary and let F' be the subgraph of $\bar{F}$ after removing v, and let $\bar{F}'$ be a maximal plane subgraph (in the drawing $D_n - \{v\}$ of K_{n-1}) containing F'; by the induction hypothesis, $\bar{F}'$ is 2-connected. We observe that v cannot have degree less than 2, as applying Lemma 1.1 to v and $\bar{F}'$ would give two edges at v not crossing $\bar{F}$, contradicting the maximality of $\bar{F}$. Suppose v has degree 2. As we assume that $\bar{F}$ is not 2-connected, F' cannot be 2-connected. However, $\bar{F}'$ is 2-connected, and hence there exists an edge e' in $\bar{F}' - F'$. By the maximality of $\bar{F}$, e' must cross at least one edge vw of $\bar{F}$ incident to v. But applying Lemma 1.1 to v and $\bar{F}'$ gives at least two edges incident to v not crossing $\bar{F}'$. These two edges and also vw do not cross $\bar{F}$, contradicting the maximality of $\bar{F}$.

Assume that $\bar{F}$ is not connected. Let C_1, C_2 be two connected components of $\bar{F}$. As all vertices have degree at least 3, C_1 cannot be an outerplanar graph, and thus has more than one face. W.l.o.g., the unbounded face contains C_2. Let v_1 be an interior vertex of C_1. Let F' be the graph obtained from $\bar{F}$ by removing v_1, and let f_1 be the face in F' that contains v_1. The face containing C_2 remains unchanged by the removal. By induction, F'

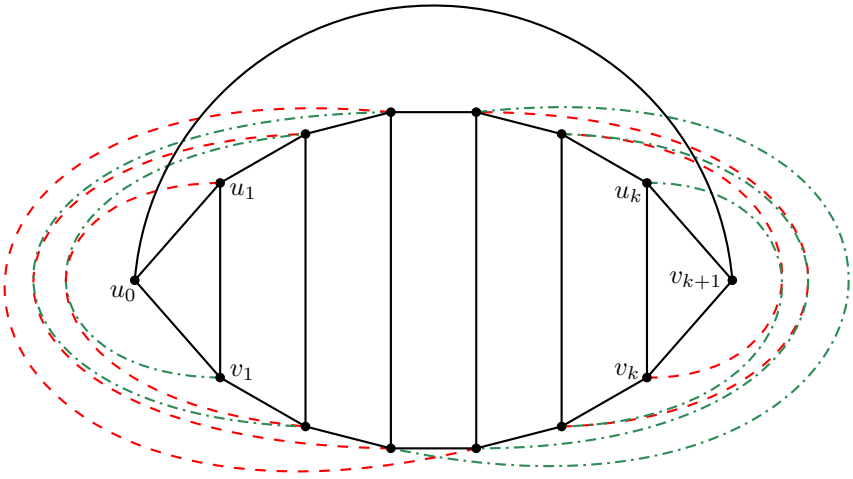


Figure 3 The black edges form a maximal plane subgraph with $\lceil 3n/2 \rceil$ edges. The missing edges should be drawn as straight line segments inside the convex hull of the set of points.

can be completed to a 2-connected plane graph $\bar{F}'$. Due to the maximality of $\bar{F}$, all edges in $\bar{F}' - F'$ must be in f_1. As C_2 is outside f_1, $\bar{F}'$ could not be connected. Thus, $\bar{F}$ is connected.

By a similar reasoning we arrive at our contradiction to $\bar{F}$ not being 2-connected. A *block* is a 2-connected component of a graph, and a *leaf block* is a block with only one cut vertex. As $\bar{F}$ is not 2-connected, it has at least two leaf blocks B_1 and B_2. As all vertices have degree at least 3, B_1 cannot have all its vertices on the same face. Again, w.l.o.g., B_2 is in the outer face of B_1, and there is an interior vertex v_1 of B_1. Removing v_1 from $\bar{F}$, we obtain a plane graph F' that has a face f_1 containing v_1, and F' is contained in a maximal plane graph $\bar{F}'$ that is 2-connected. Again, by maximality of $\bar{F}$, all edges in $\bar{F}' - F'$ must be in f_1. However, this contradicts the fact that $\bar{F}'$ is 2-connected. Hence, $\bar{F}$ must be 2-connected. ◀

Theorem 3.1 gives a means of obtaining more properties of maximal plane subgraphs.

▶ **Lemma 3.2.** *If a maximal plane subgraph $\bar{F}$ of D_n contains a vertex v of degree 2, then the subgraph of $\bar{F}$ obtained after removing v is also maximal in $D_n - \{v\}$.*

▶ **Proposition 3.3.** *Any maximal plane subgraph $\bar{F}$ of D_n with $n \geq 3$ must contain at least $\min(\lceil 3n/2 \rceil, 2n-3)$ edges. This bound is tight.*

A sketch for showing tightness of $\lceil 3n/2 \rceil$ edges is given in Figure 3.

▶ **Lemma 3.4.** *Let $C = (v_1, v_2, \ldots, v_k)$ be a plane cycle of D_n, $k \geq 3$, with faces f_1 and f_2. If there is no diagonal of C entirely in f_1, then all diagonals of C are entirely in f_2.*

Proof. The proof is by induction on k. For $k < 5$ the statement is obvious, so suppose $k \geq 5$ and consider only the subgraph induced by the vertices of C. By Lemma 3.2, there must exist a diagonal placed in f_2 connecting two vertices at distance 2 on C. W.l.o.g., let this diagonal be $v_k v_2$ and let Δ be the triangle $v_k v_1 v_2$. Then the cycle $C_1 = (v_2, v_3, \ldots, v_k)$ with $k-1$ vertices has the faces $f_1' = f_1 + \Delta$ and $f_2' = f_2 - \Delta$.

We argue that there cannot be diagonals of C_1 entirely in f_1'. Such a diagonal e would have to intersect Δ. When adding e to $C \cup \{v_k v_2\}$ and removing all edges crossed by e, we obtain a plane graph F in which v_1 has degree 0 or 1. By Lemma 1.1, there must be another edge between v_1 and C_1, and this would be a diagonal of C entirely in f_1. Thus, by induction,

any diagonal v_iv_j of C_1 is entirely in f_2' and hence also in f_2. It remains to see that the diagonals with endpoint v_1 are also in f_2. By our induction hypothesis, diagonal v_2v_4 is in f_2' and thus also in f_2. Hence, applying the hypothesis on the cycle $C_3 = (v_1, v_2, v_4, \ldots, v_k)$ we deduce that all the diagonals incident to v_3 must be in f_2. So it remains to see that diagonal v_1v_3 is also in f_2. But also v_3v_5 is in f_2', so it is also in f_2, and again by induction on the cycle $(v_1, v_2, v_3, v_5, \ldots, v_k)$, all the diagonals not incident to v_4 are also in f_2. ◀

It was previously known that even for the case where there are diagonals intersecting both faces, there are at least $\lceil k/3 \rceil$ of them not crossing C (cf. [6, Corollary 6.6]); Proposition 3.3 implies that for $k \geq 6$, there are at least$\lceil k/2 \rceil$ diagonals not crossing C.

To prove the next result we recall some definitions and properties of any 2-connected graph $G = (V, E)$. Two vertices v_1, v_2 are called a *separation pair* of G if the induced subgraph $G \setminus \{v_1, v_2\}$ on the vertices $V \setminus \{v_1, v_2\}$ is not connected. Let $G_1, \ldots, G_l$, with $l \geq 2$, be the connected components of $G \setminus \{v_1, v_2\}$. For each $i \in \{1, \ldots, l\}$, let G_i^* be the graph induced by $V(G_i) \cup \{v_1, v_2\}$. Observe that G_i^* contains at least one edge incident to v_1 and at least another incident to v_2.

▶ **Theorem 3.5.** *Let $\bar{F}$ be a maximal plane subgraph of D_n, $n \geq 3$. Then, for each separation pair v_1, v_2 of $\bar{F}$, at least one of the components $\bar{F}_i^*$ must be 2-connected.*

Proof. Suppose that v_1, v_2 is a separation pair of $\bar{F}$, that $\bar{F} \setminus \{v_1, v_2\}$ has the connected components $\bar{F}_1, \bar{F}_2, \ldots, \bar{F}_l$, and that none of the components $\bar{F}_i^*$ is 2-connected. Then each subgraph $\bar{F}_i^*$ contains at least one cut vertex u_i. Since $\bar{F}_i$ is connected and there exist edges in $\bar{F}_i^*$ incident to v_1 and v_2, vertex u_i is different from v_1 and v_2. The graph $\bar{F}_i^* \setminus \{u_i\}$ has exactly two components, one containing v_1 and the other containing v_2, as otherwise $\bar{F}$ would not be 2-connected. Thus, any path in graph $\bar{F}_i^*$ from v_1 to v_2 must use u_i. In particular, v_1v_2 cannot be an edge of $\bar{F}$. Besides, since v_1, v_2 are in different connected components of $\bar{F}_i^* \setminus \{u_i\}$, if R is the face of $\bar{F}_i^*$ where point u_i appears at least twice, then any continuous curve connecting v_1 to v_2 either contains u_i or some point of the interior of R.

Since $\bar{F}$ is 2-connected, graph $\bar{F} \setminus \{v_1\}$ is connected with v_2 as a cut vertex. As $\bar{F}$ is plane, we can suppose, w.l.o.g., that v_1 is in the outer face of $\bar{F} \setminus \{v_1\}$ and that around vertex v_1 clockwise first there appear the edges to some vertices of component $\bar{F}_1$, then edges connecting v_1 to points of $\bar{F}_2$ and so on. See Figure 4. Therefore v_1 and v_2 must be in the faces R_i of $\bar{F}$ placed between the last edge from v_1 to $\bar{F}_i$ and the first edge from v_1 to $\bar{F}_{i+1}$, for $i \in \{1, \ldots, l\}$. As, by maximality, no edge is entirely in any of those R_i faces, Lemma 3.4 implies that no point of the edge v_1v_2 in D_n can be inside any R_i. Thus, v_1v_2 must begin between two edges v_1v, v_1v' with both v and v' belonging to a common connected component $\bar{F}_i$. However, since u_i belongs to the faces R_{i-1} and R_i, any curve from v_1 and v_2 passes through point u_i or through the interior of R_{i-1} or R_i. Therefore, $\bar{F}$ cannot be maximal. ◀

We call a graph *essentially 3-edge-connected* if it stays connected after removing any two edges not sharing a vertex of degree 2 (i.e., the graph either stays connected or one component is a single vertex). Theorem 3.5 implies that a maximal plane subgraph is essentially 3-edge-connected: If the removal of two edges v_1v_2 and $v_1'v_2'$ results in two non-trivial components (see Figure 5), then the separation pair v_1, v_2' gives no 2-connected component.

▶ **Corollary 3.6.** *Any maximal plane subgraph of a simple topological drawing of K_n is essentially 3-edge-connected.*

Finally, we mention another interesting implication of Theorem 3.1; for a vertex v, we can augment $S(v)$ to a 2-connected plane graph, and thus the remaining part contains a tree.

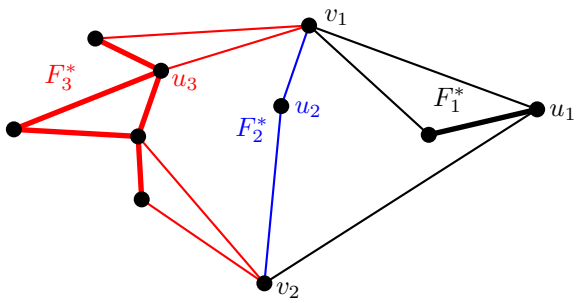


Figure 4 A plane graph with separating pair v_1, v_2 and three components F_i^*, none 2-connected.

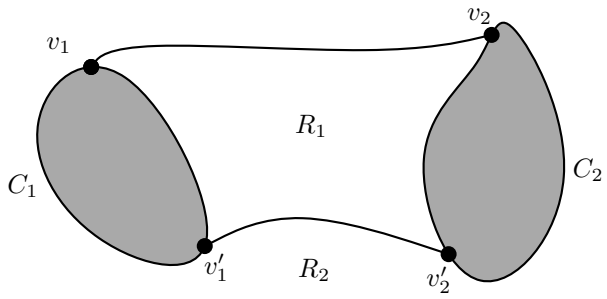


Figure 5 A graph that is not essentially 3-edge-connected. The separation pair v_1, v_2' gives two components $C_1 \cup \{v_1'v_2'\}$ and $C_2 \cup \{v_1v_2\}$, neither of which is 2-connected.

▶ **Corollary 3.7.** *Let $S(u)$ be the edges of D_n incident to a vertex u. There exist a tree T_u spanning the vertices $V \setminus \{u\}$, such that the edges of $S(u) \cup T_u$ form a plane subgraph of D_n.*

▶ **Open Problem 1.** *Given a not necessarily connected plane graph F in D_n, plus a vertex v not in F, can the edges of $S(v)$ incident to but not crossing F be found in $o(n^2)$ time?*

References

1 Jacob Fox and Benny Sudakov. Density theorems for bipartite graphs and related Ramsey-type results. *Combinatorica*, 29(2):153–196, 2009. doi:10.1007/s00493-009-2475-5.

2 Radoslav Fulek and Andres J. Ruiz-Vargas. Topological graphs: empty triangles and disjoint matchings. In Guilherme Dias da Fonseca, Thomas Lewiner, Luis Mariano Peñaranda, Timothy M. Chan, and Rolf Klein, editors, *Symp. on Computational Geometry (SoCG 2013)*, pages 259–266. ACM, 2013. doi:10.1145/2462356.2462394.

3 Jan Kynčl. Simple realizability of complete abstract topological graphs in P. *Discrete Comput. Geom.*, 45(3):383–399, 2011. doi:10.1007/s00454-010-9320-x.

4 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.

5 János Pach and Géza Tóth. Disjoint edges in topological graphs. In Jin Akiyama, Edy Tri Baskoro, and Mikio Kano, editors, *Combinatorial Geometry and Graph Theory, Indonesia-Japan Joint Conference (IJCCGGT 2003), Revised Selected Papers*, volume 3330 of *Lecture Notes in Computer Science*, pages 133–140. Springer, 2003. doi:10.1007/978-3-540-30540-8_15.

6 Jürgen Pammer. Rotation systems and good drawings. Master's thesis, Graz University of Technology, 2014.

7 Andres J. Ruiz-Vargas. Many disjoint edges in topological graphs. *Comput. Geom.*, 62:1–13, 2017. doi:10.1016/j.comgeo.2016.11.003.

8 Andrew Suk. Disjoint edges in complete topological graphs. *Discrete & Computational Geometry*, 49(2):280–286, 2013. doi:10.1007/s00454-012-9481-x.

Numerical Algorithm for the Topology of Singular Plane Curves

George Krait[1], Sylvain Lazard[1], Guillaume Moroz[1], and Marc Pouget[1]

1 Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France

Abstract

We are interested in computing the topology of plane singular curves. For this, the singular points must be isolated. Numerical methods for isolating singular points are efficient but not certified in general. We are interested in developing certified numerical algorithms for isolating the singularities. In order to do so, we restrict our attention to the special case of plane curves that are projections of smooth curves in higher dimensions. In this setting, we show that the singularities can be encoded by a regular square system whose isolation can be certified by numerical methods. This type of curves appears naturally in robotics applications and scientific visualization.

1 Introduction

Computing the topology of a curve C (i.e., a set of dimension one that is the zero locus of smooth maps) means computing a piecewise-linear graph that can be deformed continuously toward that curve. If C is smooth (i.e., the tangent space exists and is of dimension one at every point of C), there are several certified numerical methods for computing the topology. One can mention for example the global subdivision [8, 12] and certified continuation approaches [9]. On the other hand, if the curve is singular (i.e., not smooth), computing the topology is more complicated. We need, first, to isolate its singularities, second, to compute the topology in a neighborhood of those singularities and third to compute the topology of the smooth remaining part of that curve (see Figure 1). The main challenge is to isolate the singular points of the curve.

State-of-art methods to isolate the singular points of a curve are symbolic in general e.g., based in resultant and sub-resultant theory [2], Gröbner basis or rational univariate representations [13, 3]. However, symbolic approaches suffer from inefficiency while numerical methods fail to be certified for singular curves. Our goal is to develop efficient numerical methods that avoid losing the correctness that symbolic ones offer.

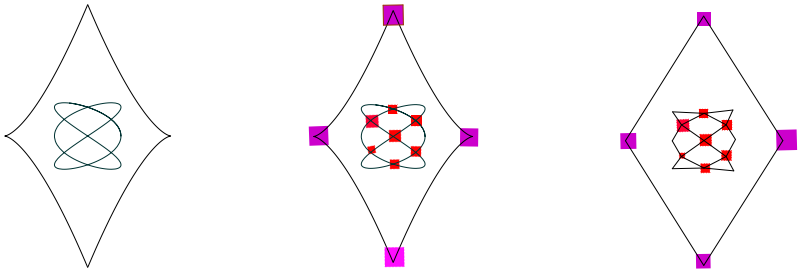
We show that this could be achieved for the specific class of plane curves that are projections of smooth curves in higher dimension. This is achieved by exhibiting a regular and square system that characterizes, under some generic assumptions, the singular points of the plane projection of a generic curve (see Theorem 4.2). This system, being square and regular, satisfies the conditions to apply certified numerical isolation methods [10, Chapter 8].

2 Assumptions

We first recall the definitions of a node and an ordinary cusp (see Figure 2).

▶ **Definition 2.1.** Let C be a curve in $\mathbb{R}^2$ and $p \in C$. We call p an ordinary cusp (resp. a node) if there exists an open subset V of $\mathbb{R}^2$, a smooth diffeomorphism $\varphi : V \to V$ such that V contains p and $\varphi(V \cap C) = V \cap C'$, where C' is the zero set of the map $x_1^2 - x_2^3$ (resp. $x_1^2 - x_2^2$), for some local coordinates system (x_1, x_2) at p.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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) (c)

Figure 1 (a): A given curve. (b) Isolating and classifying the singular points. (c) A graph homeomorphic to the given curve.

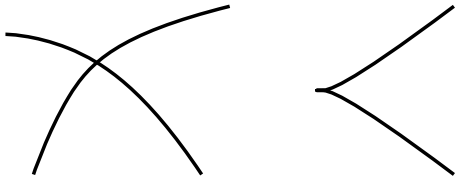


Figure 2 A node and an ordinary cusp.

Let $n \geq 3$ be an integer and U be a bounded open set in $\mathbb{R}^n$. Let $C^\infty(U, \mathbb{R}^{n-1})$ be the space of smooth maps (i.e., the set of maps that are differentiable infinitely many times) from U to $\mathbb{R}^{n-1}$ equipped with the weak topology defined in [4, §3.9.2] or [5, p.34]. Consider the map $P = (P_1, \ldots, P_{n-1}) \in C^\infty(U, \mathbb{R}^{n-1})$. We denote by C_P the solution set of the system $P_1(x) = \cdots = P_{n-1}(x) = 0$, with $x = (x_1, \ldots, x_n) \in U$. Also, consider the projection π from C_P to $\mathbb{R}^2$ that sends x to (x_1, x_2). Let J_P be the Jacobian matrix of size $(n-1) \times n$ of the system $P_1(x) = \cdots = P_{n-1}(x) = 0$. We assume that:

Assumption 1. For all $q \in C_P$, $\mathrm{rank}(J_P(q)) = n - 1$, which implies that C_P is a smooth curve.[1]

Assumption 2. The set of $q \in C_P$ that have tangent line orthogonal to $\mathbb{R}^2$ (the plane defined by the first two coordinates of $\mathbb{R}^n$), is finite.

Assumption 3. π is injective except at a finite number of points.

Assumption 4. The preimage of any point of $\mathbb{R}^2$ under π is at most two points (counted with multiplicity).

Assumption 5. Every singular point of $\pi(C_P)$ is either an ordinary cusp or a node.

▶ **Definition 2.2.** We denote by $\mathbb{U} \subset C^\infty(U, \mathbb{R}^{n-1})$ the set of maps for which all the above assumptions are satisfied.

We prove in Theorem 4.1 that our above assumptions are generic. Genericity in topology is similar to the notion of almost everywhere in measure theory, see Section 3.2.6. in [4] for more details. More formally:

[1] Note that the converse is not true as the vertical (double) line defined by $x_1^2 = x_2 = 0$ in $\mathbb{R}^3$ is smooth but the rank of its Jacobian is never full.

▶ **Definition 2.3.** [4, Definition 3.2.5] A subset of $C^\infty(U, \mathbb{R}^{n-1})$ is called residual if it contains the intersection of a countable family of dense open subsets in $C^\infty(U, \mathbb{R}^{n-1})$. A property is generic if it holds for a residual subset.

▶ Remark. By [4, Proposition 3.9.3], since $C^\infty(U, \mathbb{R}^{n-1})$ is a Baire space, every residual subset of $C^\infty(U, \mathbb{R}^{n-1})$ is dense.

Given a map $P \in \mathbb{U}$, our goal is to isolate the singular points of $\pi(C_P)$, that is to find a set of boxes (i.e., Cartesian products of 2 intervals) in $\mathbb{R}^2$ such that each box contains exactly one singular point of $\pi(C_P)$. This is the first step towards computing the topology of the curve $\pi(C_P)$.

3 Regular sytems

We recall the definition of regularity and illustrate it in Figure 3.

▶ **Definition 3.1.** A regular system is such that its Jacobian matrix is full rank at its solutions.

Figure 3 $p = (0, 0)$ is a regular solution of the system $\{x_1 + x_2 = x_1 - x_2 = 0\}$ (left) and it is not a regular solution of the system $\{x_1 + x_2^2 = x_1 - x_2^2 = 0\}$(right).

In order to isolate singular points of $\pi(C_P)$ using certified numerical methods [10, Chapter 8], we first need to encode them as the solutions of a regular zero-dimensional system of equations. To define such a system in Proposition 4.1, we introduce some notation.

▶ **Definition 3.2.** Let y, r be two sets of $n-2$ real variables and x_1, x_2, t be real variables. For a smooth map $f : U \to \mathbb{R}$, with $U \subseteq \mathbb{R}^n$, we define the maps:

$$S \cdot f(x_1, x_2, y, r, t) = \begin{cases} \frac{1}{2}[f(x_1, x_2, y + r\sqrt{t}) + f(x_1, x_2, y - r\sqrt{t})] & \text{for } t \neq 0 \\ f(x_1, x_2, y), & \text{for } t = 0. \end{cases}$$

$$D \cdot f(x_1, x_2, y, r, t) = \begin{cases} \frac{1}{2\sqrt{t}}[f(x_1, x_2, y + r\sqrt{t}) - f(x_1, x_2, y - r\sqrt{t})] & \text{for } t \neq 0 \\ \nabla f \cdot (0, 0, r), & \text{for } t = 0. \end{cases}$$

4 Contributions

▶ **Theorem 4.1.** *Assumptions* $1 - 5$ *in Section 2 are generic, that is,* $\mathbb{U}$ *is residual in* $C^\infty(U, \mathbb{R}^{n-1})$.

We omit the proof Theorem 4.1 which is based on Thom's Transversality Theorem [4, Theorem 3.9.4].

▶ **Theorem 4.2.** *For* $P \in \mathbb{U}$ *there exists a regular zero-dimensional system* Ball(P) *(defined in Proposition 4.1) of* $2n - 1$ *variables such that every real solution of* Ball(P) *projects bijectively to a singular point in* $\pi(C_P)$*, where the projection here is the one that sends every point in* $\mathbb{R}^{2n-1}$ *to its first two coordinates.*

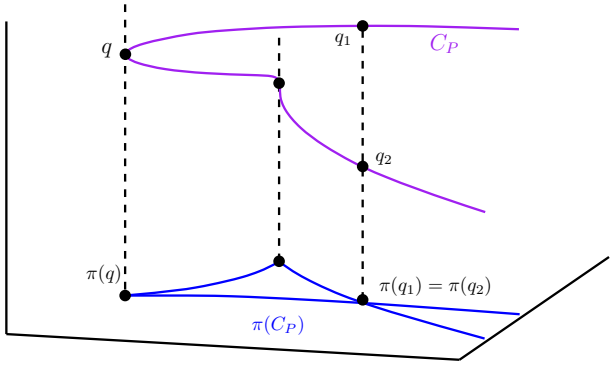


Figure 4 Illustration of Proposition 4.1 in the case $n = 3$.

More precisely, Proposition 4.1 explicits this bijection and Proposition 4.2 proves the regularity.

▶ Remark. Analogously to [7], we can check, using a semi-algorithm, whether a given map $P \in C^\infty(U, \mathbb{R}^{n-1})$ satisfies Assumptions $1-4$. This semi-algorithm stops if and only if Assumptions $1-4$ are satisfied. The reformulation of Assumption 5 as the regularity of system $\mathrm{Ball}(P)$ in Proposition 4.2 also enables to check it via a semi-algorithm.

▶ **Proposition 4.1.** *Consider a map $P = (P_1, \ldots, P_{n-1}) \in C^\infty(U, \mathbb{R}^{n-1})$ that satisfies Assumptions $1-5$. Let $\mathrm{Ball}(P)$ be the system:*

$$\begin{cases} S \cdot P_1 = \cdots = S \cdot P_{n-1} = 0 \\ D \cdot P_1 = \cdots = D \cdot P_{n-1} = 0 \\ r_1^2 + \cdots + r_{n-2}^2 = 1. \end{cases} \tag{1}$$

Denote by $M_P \subseteq \mathbb{R}^{2n-1}$ the set of real solutions of $\mathrm{Ball}(P)$. Let $X = (x_1, x_2, y, r, t) \in \mathbb{R}^{2n-1}$, with $r_1^2 + \cdots + r_{n-2}^2 = 1$ and consider $q_1 = (x_1, x_2, y + r\sqrt{t})$ and $q_2 = (x_1, x_2, y - r\sqrt{t})$ in $\mathbb{R}^n$. Then, X is in M_P if and only if one of the following cases holds:

- *$q_1 \neq q_2$, $q_1, q_2 \in C_P$ and $\pi(q_1)$ is a node.*
- *$q_1 = q_2$, C_P contains q_1, $(0, 0, r) \in T_{q_1} C_P$ and $\pi(q_1)$ is an ordinary cusp.*

▶ **Proposition 4.2.** *Assume that $P \in C^\infty(U, \mathbb{R}^{n-1})$ satisfies Assumptions $1-4$, then $\mathrm{Ball}(P)$ is regular at its solution if and only if Assumption 5 is satisfied.*

Proof. (Sketch) Let $X = (x_1, x_2, y, r, t)$ be a solution of $\mathrm{Ball}(P)$. We consider two cases depending on t and prove that X is a regular solution of $\mathrm{Ball}(P)$ if and only if (x_1, x_2) is either a node (when $t \neq 0$) or an ordinary cusp (when $t = 0$).

Case $t \neq 0$. Let $q_1 = (x_1, x_2, y + r\sqrt{t})$ and $q_2 = (x_1, x_2, y - r\sqrt{t})$ and $J_{\mathrm{Ball}(P)}$ be Jacobian matrix of $\mathrm{Ball}(P)$. By linear operations on $J_{\mathrm{Ball}(P)}$, we get that the latter is of full rank if and only if the matrix $M = \begin{pmatrix} N_P(q_1) & 0 & M_P(q_1) \\ N_P(q_2) & M_P(q_2) & 0 \end{pmatrix}$ is full rank, where $M_P(q_1), M_P(q_2)$ are the $((n-1) \times (n-2))$-submatrices that are obtained respectively by removing the first two columns from $J_P(q_1), J_P(q_2)$ and $N_P(q_1), N_P(q_2)$ are the $((n-1) \times 2)$-submatrices formed by the first two columns of $J_P(q_1), J_P(q_2)$ respectively.

Now, $\mathrm{Ball}(P)$ is not regular at X is equivalent to $\det(M) = 0$, which is equivalent to the fact that there exist $\alpha \in \mathbb{R}^2 \setminus \{0\}$ and $\beta, \gamma \in \mathbb{R}^{n-2}$ such that (α, β, γ) is in the kernel of M.

Under our assumption, the last statement is equivalent to say that (α, β) and (α, γ) are in $T_{q_1}C_P$ and $T_{q_2}C_P$ respectively and none of them is trivial. Equivalently, $\pi(q_1)$ is not a node. This concludes the proof in the case where $t \neq 0$ because we proved that X is not a regular solution of Ball(P) if and only if $\pi(q_1)$ is not a node. Hence, $\pi(q_1)$ is a node if and only if X is a regular solution of Ball(P).

Case $t = 0$. By Proposition 4.1, q_1 is in C_P. Using implicit function theorem [4, Corollary 2.7.2] and Hadamard's Lemma [4, Proposition 4.2.3] we can prove that, in a neighborhood of q in U, there exists a local coordinates system $(z_1, \ldots, z_n)$ such that a neighborhood of $\pi(q_1)$ in $\pi(C_p)$ is the zero set of the equation $z_1^2 - z_2^{2k+1} = 0$, for some integer $k \geq 1$. By computing explicitly the Jacobian of Ball(P) in this new coordinate system, we can see that Ball(P) is regular at X if and only if $k = 1$. Thus, in the above local coordinate system, $\pi(C_p)$ has equation $z_1^2 - z_2^3 = 0$ and thus $\pi(q_1)$ is an ordinary cusp by Definition 2.1. Hence, X is a regular solution of Ball(P) if and only if $\pi(q_1)$ is an ordinary cusp of $\pi(C_p)$. ◀

5 Algorithmic aspects

Propositions 4.1 and 4.2 pave the way to the following algorithm:

The input of the algorithm is an integer $n \geq 3$, a bounded open subset U in $\mathbb{R}^n$ and $P \in \mathbb{U}$. The output is a set of boxes S each of which lives in $\mathbb{R}^2$ and contains exactly one singular point of $\pi(C_P)$. The idea of the algorithm is, first, to compute the system Ball(P). Second, to isolate the solutions of Ball(P) using a certified numerical solver (see for example [11]), computing a set of boxes S_0 in $\mathbb{R}^{2n-1}$ (the Cartesian product of $2n-1$ intervals) each of which contains exactly one solution of Ball(P). We can shrink every box of S_0 in such a way that their projections to $\mathbb{R}^2$ are pairwise disjoint. Finally, we project each box of S_0 and we add the projection to S, where the projection here is the one that sends every point in $\mathbb{R}^{2n-1}$ to its first two coordinates. Thus, every projected box in $\mathbb{R}^2$ contains exactly one singular point of $\pi(C_P)$.

The bottleneck of this method is the resolution of Ball(P) in $\mathbb{R}^{2n-1}$. For the certified numerical solver we can use homotopy methods [1] or subdivision methods [11]. Moreover, if we use subdivision methods, we might try to use the structure of the Ball system to reduce the dimension, as done in [6] for $n = 3$.

References

1 Daniel J. Bates, Jonathan D. Hauenstein, Andrew J. Sommese, and Charles W. Wampler. Bertini: Software for numerical algebraic geometry. URL: `https://bertini.nd.edu`, `doi:10.7274/R0H41PB5`.

2 Jean-Daniel Boissonnat and Monique Teillaud, editors. *Effective computational geometry for curves and surfaces*. Mathematics and Visualization. Springer-Verlag, Berlin, 2007.

3 Jinsan Cheng, Sylvain Lazard, Luis Peñaranda, Marc Pouget, Fabrice Rouillier, and Elias Tsigaridas. On the topology of real algebraic plane curves. *Mathematics in Computer Science*, 4(1):113–137, Nov 2010. URL: `https://doi.org/10.1007/s11786-010-0044-3`, `doi:10.1007/s11786-010-0044-3`.

4 Michel Demazure. *Bifurcations and catastrophes*. Universitext. Springer-Verlag, Berlin, 2000. Geometry of solutions to nonlinear problems, Translated from the 1989 French original by David Chillingworth. URL: `https://doi.org/10.1007/978-3-642-57134-3`.

5 Morris William Hirsch. *Differential topology*. Springer-Verlag, New York-Heidelberg, 1976. Graduate Texts in Mathematics, No. 33.

6 Rémi Imbach, Guillaume Moroz, and Marc Pouget. Reliable location with respect to the projection of a smooth space curve. *Reliab. Comput.*, 26:13–55, 2018.

7 Rémi Imbach, Guillaume Moroz, and Marc Pouget. A certified numerical algorithm for the topology of resultant and discriminant curves. *Journal of Symbolic Computation*, 80, Part 2:285 – 306, 2017. URL: http://www.sciencedirect.com/science/article/pii/S0747717116300128, doi:http://dx.doi.org/10.1016/j.jsc.2016.03.011.

8 Chen Liang, Bernard Mourrain, and Jean-Pascal Pavone. Subdivision methods for the topology of 2d and 3d implicit curves. In *Geometric modeling and algebraic geometry*, pages 199–214. Springer, Berlin, 2008. URL: https://doi.org/10.1007/978-3-540-72185-7_11, doi:10.1007/978-3-540-72185-7_11.

9 Benjamin Martin, Alexandre Goldsztejn, Laurent Granvilliers, and Christophe Jermann. Certified parallelotope continuation for one-manifolds. *SIAM J. Numer. Anal.*, 51(6):3373–3401, 2013. URL: https://doi.org/10.1137/130906544, doi:10.1137/130906544.

10 Ramon E. Moore, R. Baker Kearfott, and Michael J. Cloud. *Introduction to interval analysis*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2009. URL: https://doi.org/10.1137/1.9780898717716, doi:10.1137/1.9780898717716.

11 Arnold Neumaier. *Interval Methods for Systems of Equations*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1991. doi:10.1017/CBO9780511526473.

12 Simon Plantinga and Gert Vegter. Isotopic approximation of implicit curves and surfaces. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '04, pages 245–254, New York, NY, USA, 2004. ACM. URL: http://doi.acm.org/10.1145/1057432.1057465, doi:10.1145/1057432.1057465.

13 Fabrice Rouillier. Solving zero-dimensional systems through the rational univariate representation. *Appl. Algebra Engrg. Comm. Comput.*, 9(5):433–461, 1999. URL: https://doi.org/10.1007/s002000050114, doi:10.1007/s002000050114.

Kinetic Volume-Based Persistence for 1D Terrains

Tim Ophelders[1], Willem Sonke[*2], Bettina Speckmann[†2], and Kevin Verbeek[‡2]

1 **Department of Computational Mathematics, Science and Engineering, Michigan State University, USA**
ophelder@egr.msu.edu

2 **Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands**
[w.m.sonke|b.speckmann|k.a.b.verbeek]@tue.nl

1 Introduction

The evolution of the channel network of *braided* rivers is an important topic in geomorphology. The merges and splits of braided river channels evolve over time under the influence of water pressure and sediment transport. Kleinhans *et al.* [6] recently presented algorithms to extract a (static) network of significant channels from the elevation data of a river bed; this method is already used in geomorphological studies [5, 8]. The challenge is to identify *significant* channels from the river bed, since measurement errors and small variations in the terrain generally cause a multitude of possible channels. Two channels can be considered similar if the volume of terrain between the channels is small, modelling the fact that a small volume of sediment can easily be eroded by water flow, merging the two channels into one. A network of significant channels consists of channels which are sufficiently dissimilar.

To analyze the evolution of significant channels over time we could kinetically maintain the network of significant channels computed by the method of Kleinhans *et al.* [6]. However, these networks are not stable over time and also prohibitively expensive to compute. We hence propose a simplified model which uses a *volume-simplified* terrain. Specifically, we *prune* topological features of the terrain (minima and maxima) that can be eliminated by removing only a small amount of volume. The idea is that the remaining topological features separate significant channels.

Pruning of the terrain based on the volume of the removed features resembles terrain simplification based on (height) persistence. The notion of topological persistence was introduced by Edelsbrunner *et al.* [4]. Persistence can be defined via measures other than the vertical distance between points. Carr *et al.* [3] describe a method to simplify contour trees (which capture the topological structure of a terrain) using so-called *local geometric measures*, such as (in 2D terrains) the line length of the contour, the area enclosed by the contour, or the volume of the enclosed region. This last type of persistence is exactly the one we use in this paper and we therefore refer to our simplified terrain as *volume-persistent*.

Our goal is to maintain a volume-persistent terrain over time, which is closely related to maintaining its topological structure over time, as represented, for example, by its contour tree or its split tree. Agarwal *et al.* [1] show how to maintain a 2D contour tree kinetically. They also argue that they can maintain height persistence over time. However, maintaining volume-persistence is much more challenging, because we have to detect the events that occur when a pruned part of the terrain attains a certain threshold volume. The complexity

[*] Supported by the Netherlands Organisation for Scientific Research (NWO); 639.023.208.
[†] Partially supported by the Netherlands Organisation for Scientific Research (NWO); 639.023.208.
[‡] Supported by the Netherlands Organisation for Scientific Research (NWO); 639.021.541.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

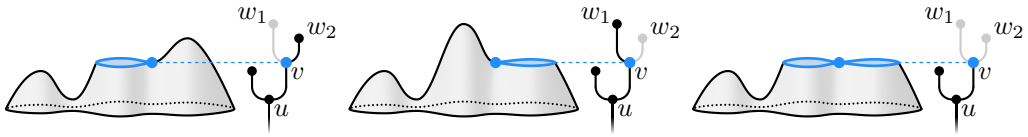


Figure 1 Pruning a terrain at the height of a saddle point v (split tree shown next to the terrain).

of the associated certificates can be high, since the volumes can be determined by linearly many vertices. Furthermore, the volume above a particular contour in the terrain is not continuous: when two contours merge the associated volumes are summed, whereas the associated height is simply the maximum of the two heights. Hence the volume-based pruning can be "stuck" at a critical point (see Fig. 1: pruning just below the saddle point requires a much higher volume-threshold than pruning exactly at the saddle point). Since these issues already manifest themselves for 1D terrains, we restrict ourselves to this setting.

Preliminaries. Let T be a 1D terrain with n vertices, of which each vertex v has a fixed x-coordinate x_v and height h_v that changes linearly over time according to a known *flight plan*: $h_v(t) = a_v\, t + b_v$. Between vertices, the height is interpolated linearly; $h(x,t)$ denotes the height of T at x on time t. At time t, the *superlevel set* of T at height h is the set of points in T with $h(x,t) \geq h$. A superlevel set may consist of several connected components. For $h = -\infty$, the superlevel set spans the entire terrain. If we continuously increase h, at certain moments topological changes happen to the superlevel set. The *split tree* S of T represents those changes: a component splitting is represented by an internal vertex, and a component disappearing is represented by a leaf [2] (see Fig. 2a). We consider S to be rooted at the vertex at $h = -\infty$, so every edge is directed upwards. Let $e = (u, v)$ be a split tree edge. Then the *parent edge* $p(e)$ of e is the incoming edge of u, and the *child edges* of e are the outgoing edges of v. The *subtree* rooted at e is the subtree rooted at v, plus e itself.

To prune the terrain we cut off a single split component at a particular height h (see Fig. 2b). We can equivalently view this operation as pruning the split tree: we identify some point on an edge of the split tree, and remove the entire subtree above it. If we prune at the height of a minimum v, then we need to specify which of the outgoing edges of v we want to prune. We define $\mathcal{A}(e,h)$ as the area of terrain that is cut off if we prune edge e at height h. Let $A \in \mathbb{R}$ be some (fixed) positive area threshold. We define the *area-persistent* terrain

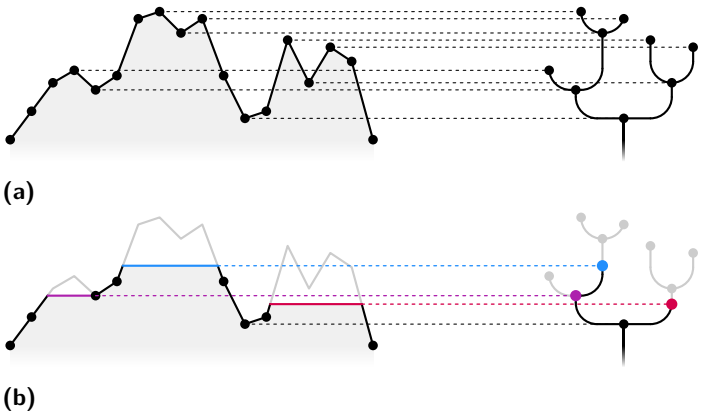


Figure 2 Area-simplifying a 1D terrain: **(a)** before, and **(b)** after.

as the terrain left after pruning all edges e in S at height h, whenever $\mathcal{A}(e,h) \leq A$. In the resulting terrain, all pieces that are cut off have area at most A.

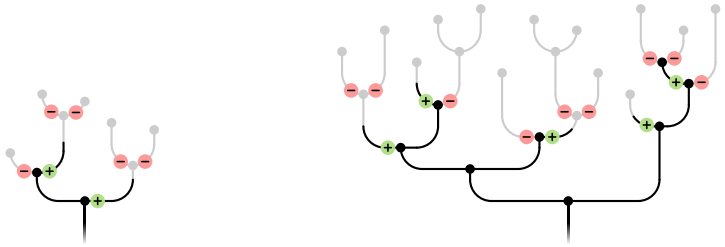
Results. We describe a KDS that maintains the pruned split tree of an area-persistent terrain under linear vertex motion. That is, for each edge $e = (u, v)$, we maintain if $\mathcal{A}(e, h_u)$ is smaller or larger than A. This KDS is compact, responsive, local, and efficient.

2 A KDS for maintaining an area-persistent 1D terrain

We adapt the KDS for 2D contour trees by Agarwal *et al.* [1] to 1D split trees. This KDS uses four types of events: shift, birth, death, and interchange events, which can all be handled by changing the split tree. We maintain the split tree S as a *link–cut* tree [7]. Next, we add *area certificates* to detect so-called *area events* when a pruning boundary moves from one split tree edge to the other.

We define $\mathcal{A}_e := \mathcal{A}(e, h_u)$, so $\mathcal{A}_e$ is the area that is removed when we prune away the subtree rooted at e from S, and we write $\mathcal{A}_e(t)$ to denote $\mathcal{A}_e$ at time t. If $\mathcal{A}_e(t) > A$, we say that e is *significant* at time t, otherwise we say that e is *insignificant*. We call a certificate $\mathcal{A}_e > A$ ($\mathcal{A}_e < A$) an *upper (lower) area certificate* for edge e (see Fig. 3a).

If an edge e is insignificant, then all edges in the subtree rooted at e are insignificant; similarly, if e is significant, then all ancestor edges of e are significant. Therefore we do not need to store area certificates for all edges in S: we store a lower area certificate for e only if $p(e)$ is significant, and an upper area certificate for e only if all children of e are insignificant (see Fig. 3b, only stored area certificates are shown). On each root-to-leaf path through S, at most two area certificates are stored: one upper and one lower certificate.



(a) **(b)**

Figure 3 Two area-persistent split trees with area certificates: **(a)** the terrain from Fig. 2b, and **(b)** a more complicated terrain. (+ and − denote upper and lower area certificates, respectively.)

Computing areas. We use an additional data structure that is based on the flight plans of the vertices and that needs to be updated only when a flight plan changes. Using this data structure, given two arbitrary terrain vertices v and w and some time t, we want to be able to compute $\int_{x_v}^{x_w} h(x,t)\,\mathrm{d}x$ in $O(\log n)$ time. If $e = (p, q)$ is a terrain edge, then

$$\begin{aligned}\int_{x_p}^{x_q} h(x,t)\,\mathrm{d}x &= \tfrac{1}{2}(x_q - x_p)(a_p t + b_p + a_q t + b_q) \\ &= \underbrace{\tfrac{1}{2}(x_q - x_p)(a_p + a_q)}_{=:\,a_e}\, t + \underbrace{\tfrac{1}{2}(x_q - x_p)(b_p + b_q)}_{=:\,b_e}.\end{aligned}$$

Then for arbitrary terrain vertices v and w,

$$\int_{x_v}^{x_w} h(x,t)\,\mathrm{d}x = \sum_{e=(p,q)} \int_{x_p}^{x_q} h(x,t)\,\mathrm{d}x = \Big(\sum_e a_e\Big)t + \sum_e b_e,$$

where the summations range over all terrain edges between v and w. To evaluate $\sum_e a_e$ and $\sum_e b_e$ efficiently, we compute a_e and b_e for each terrain edge e and store them in a balanced binary tree R, augmented with sums of subtrees, using $O(n)$ preprocessing time. We then support $O(\log n)$ time queries for the sum of a_e or b_e values for all edges e between two query vertices v and w, so we can compute $\int_{x_v}^{x_w} h(x,t)$ in $O(\log n)$ time for any v, w and t.

Detecting area events. We need to answer the following question: Given a split tree edge $e = (u, v)$ at time t_0, what is the next time t_{event} at which $\mathcal{A}_e(t_{\text{event}}) = A$? We assume without loss of generality that $e = (u, v)$ is a right-going edge in S. Let $x_{\text{ray}}(t)$ be the x-coordinate of the first point on the terrain to the right of u where $h(x,t) = h_u(t)$. That is, $x_{\text{ray}}(t)$ is the point where a horizontal ray from u to the right stabs the terrain at time t.

We first assume that we know the edge $e_{\text{event}} = (p, q)$ that $x_{\text{ray}}(t_{\text{event}})$ lies on, that is, the first edge intersected by the ray at the time that the area certificate fails (see Fig. 4a). In this case, we can compute the exact failure time t_{event} as follows. $\mathcal{A}_e(t)$ is the sum of the areas induced by the terrain edges between u and p, which are fully above the ray, and the area of the triangle between the ray and (p, q) (see Fig. 4b). This implies that $\mathcal{A}_e(t_{\text{event}}) = A$ is a quadratic equation with a closed-form solution, and can therefore be solved exactly. In other words, given e_{event}, we can compute t_{event} in $O(\log n)$ time.

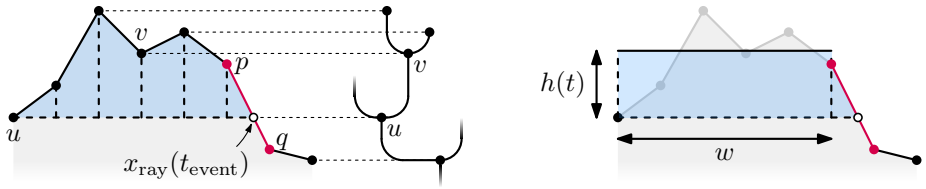


Figure 4 Computing the area $\mathcal{A}_e(t_{\text{event}})$ (shaded in blue).

Finding the stabbed chain. In the following we consider monotone *chains* of the terrain. We call the chain that contains $x_{\text{ray}}(t_0)$ the *stabbed chain* of edge $e = (u, v)$ and describe an algorithm StabbedChain that finds its lower endpoint m (see Fig. 5).

▶ **Lemma 1.** *Let $e = (u, v)$ be a right-going edge in S, and let (m, m') be the last left-going edge on the path π from the root to u. Then m is the lower endpoint of the stabbed chain of e.*

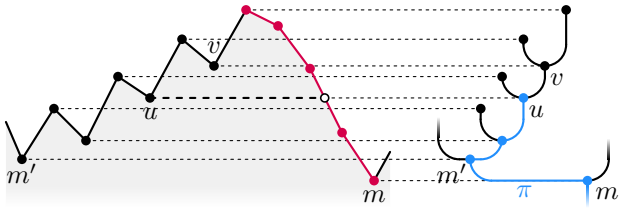


Figure 5 Finding the lower endpoint m of the stabbed chain (red) by traversing the split tree.

Since path π may have linear length we cannot traverse it to search for m. Instead we query the link–cut tree for a binary search tree containing all vertices on π (this is an Expose query). In this tree we find the rightmost vertex m' whose incoming edge is left-going in S, and return the parent m of m'. For this we augment the link–cut tree: we store for each edge whether it is left- or right-going. This results in a $O(\log n)$ running time for Stabbed-Chain.

Finding e_{event}. At current time t_0 we first use Stabbed-Chain to find vertex m. The predecessor of m in the split tree is the maximum M of the stabbed chain. We binary search between M and m for the edge that $x_{\text{ray}}(t_0)$ lies on and then find the exact value for $x_{\text{ray}}(t_0)$. Consider the point s on the terrain at x-coordinate $x_{\text{ray}}(t_0)$. We distinguish three cases based on how s moves over time. In the simplest case, s moves at the same speed as u and hence x_{ray} is constant, so we simply return the edge $x_{\text{ray}}(t_0)$ lies on. Otherwise, s may be moving upwards or downwards relative to u. If s moves upwards, x_{ray} is strictly increasing over time; if s moves downwards, x_{ray} is strictly decreasing. In the following we assume that x_{ray} is strictly increasing (see Fig. 6); the other case is symmetric.

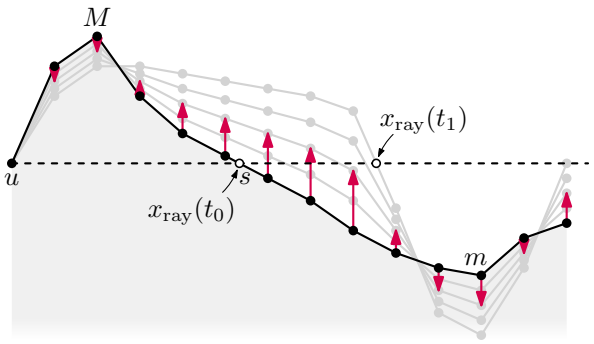


Figure 6 If s moves upwards, then x_{ray} is strictly increasing. Terrain at t_0 in black; terrain at later times in gray. The red arrows indicate how the vertices move relative to u.

We then find the time t_1 at which the next shift, birth or death event occurs involving vertices between u and m. (This can be done efficiently using a 1D range tree storing all such events, with their x-coordinate as the key and their failure time as the value.) Since therefore no events occur between t_0 and t_1, $x_{\text{ray}}(t_1)$ lies in the same chain as $x_{\text{ray}}(t_0)$. Therefore we can compute $x_{\text{ray}}(t_1)$ in the same way we computed $x_{\text{ray}}(t_0)$. Given some $x \in [x_{\text{ray}}(t_0), x_{\text{ray}}(t_1)]$, let $t_{\text{ray}}(x)$ be the time at which the point at x hits the ray. Being the inverse of x_{ray}, t_{ray} is also strictly increasing.

▶ **Lemma 2.** *The function $\mathcal{A}'_e(x) := \mathcal{A}_e(t_{ray}(x))$ is unimodal within $[x_{ray}(t_0), x_{ray}(t_1)]$: there exists an x_{fixed} such that $\mathcal{A}'_e(x)$ is strictly descending for $x \in [x_{ray}(t_0), x_{fixed})$ and strictly increasing for $x \in (x_{fixed}, x_{ray}(t_1)]$.*

Proof. We study the derivative $\mathrm{d}\mathcal{A}'_e/\mathrm{d}x$ by considering the following extension of $\mathcal{A}'_e(x)$:

$$\mathcal{A}'_e(x,t) = \int_u^x \big(h(x',t) - h_u(t)\big)\,\mathrm{d}x'.$$

$\mathcal{A}'_e(x,t)$ represents the area cut off by a horizontal ray to the right starting from vertex u, until x-coordinate x, so $\mathcal{A}'_e(x) = \mathcal{A}'_e(x, t_{\text{ray}}(x))$. We set $t = t_{\text{ray}}(x)$ and compute

$$\frac{\mathrm{d}\mathcal{A}'_e}{\mathrm{d}x} = \frac{\partial \mathcal{A}'_e}{\partial t}\frac{\mathrm{d}t}{\mathrm{d}x} + \frac{\partial \mathcal{A}'_e}{\partial x} = \frac{\partial \mathcal{A}'_e}{\partial t}\frac{\mathrm{d}t}{\mathrm{d}x}.$$

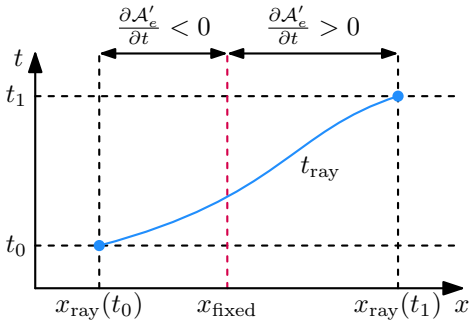


Figure 7 Sketch of the function t_{ray} drawn in the (x, t)-plane.

The last equality follows because $\partial \mathcal{A}'_e/\partial x = 0$, as for $x = x_{\text{ray}}(t)$ by definition $h(x, t) = h_u(t)$. Because t_{ray} is increasing, $\mathrm{d}t/\mathrm{d}x > 0$, so $\mathrm{d}\mathcal{A}'_e/\mathrm{d}x$ has the same sign as $\partial \mathcal{A}'_e/\partial t$.

Let x_{fixed} be the x-coordinate within $[x_{\text{ray}}(t_0), x_{\text{ray}}(t_1)]$ such that $\mathcal{A}'_e(x_{\text{fixed}}, t)$ is constant in t, if such a coordinate exists. The average terrain height on $[x_u, x_{\text{fixed}}]$ is hence constant, and all terrain points in $[x_{\text{fixed}}, x_{\text{ray}}(t_1)]$ are moving upwards relative to u. Therefore, for $x \in (x_{\text{fixed}}, x_{\text{ray}}(t_1)]$, the average terrain height on $[x_u, x]$ is growing, so $\partial \mathcal{A}'_e/\partial t > 0$ (see Fig. 7). Similarly, within $[x_{\text{ray}}(t_0), x_{\text{fixed}})$, $\partial \mathcal{A}'_e/\partial t < 0$. If no x_{fixed} exists, then $\partial \mathcal{A}'_e/\mathrm{d}t > 0$ or $\partial \mathcal{A}_e/\mathrm{d}t < 0$ throughout $[x_{\text{ray}}(t_0), x_{\text{ray}}(t_1)]$. Hence, $\mathcal{A}'_e(x)$ is unimodal. ◀

First we find x_{fixed} by binary searching within the range $[x_{\text{ray}}(t_0), x_{\text{ray}}(t_1)]$, using tree R. Then, we compute $\mathcal{A}'_e(x_{\text{ray}}(t_0))$ and $\mathcal{A}'_e(x_{\text{ray}}(t_1))$ to determine whether e_{event} lies within $[x_{\text{ray}}(t_0), x_{\text{fixed}})$ or $(x_{\text{fixed}}, x_{\text{ray}}(t_1)]$, and do a binary search (again using R) on that range to find e_{event}. The binary search takes $O(\log n)$ steps, each taking constant time, so it takes $O(\log n)$ time to compute $\mathcal{A}'_e(x)$.

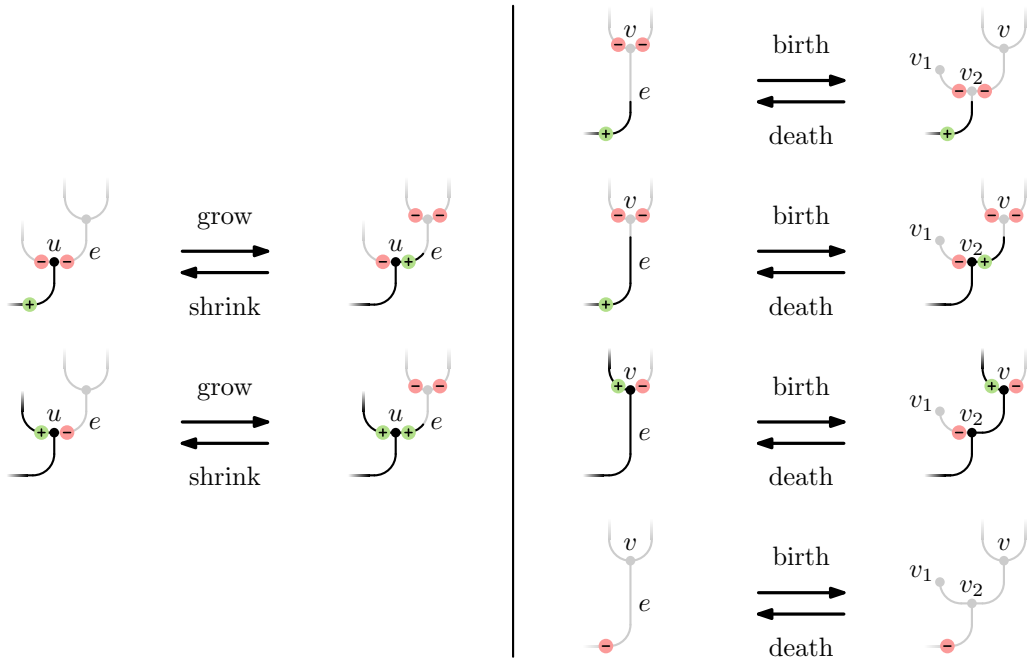


Figure 8 Handling area events *(left)* and birth / death events *(right)*.

Event handling. There are two complementary types of area events: firstly, $\mathcal{A}_e < A$ may fail (a *grow event*), and secondly, $\mathcal{A}_e > A$ may fail (a *shrink event*). Both of them can be handled by inserting and removing area certificates (see the left of Fig. 8). Shift, birth, death, and interchange events are handled like in the KDS from Agarwal *et al.*, but some additional actions are required to ensure that the area certificates are updated (see the right of Fig. 8).

Analysis. The KDS is still compact, responsive, local, and efficient. In particular, each vertex is involved in a constant number of area certificates. Indeed, a vertex is involved in the area certificate of a split tree edge $e = (u, v)$ if it lies between u and m (see Fig. 5). A chain is stabbed by only one upper area certificate, since any rays stabbing the same chain need to originate from vertices on the same root-to-leaf path in the split tree (by Lemma 1); as we store upper area certificates for an edge only if all its children are insignificant, this path can contain only one upper area certificate. Therefore, each vertex in the interior of the chain is involved in at most one upper area certificate and by similar reasoning, at most one lower area certificate. A minimum is part of two chains and hence involved in at most two upper and two lower area certificates. This implies that the locality is $O(1)$.

References

1 Pankaj Agarwal, Thomal Mølhave, Morten Revsbæk, Issam Safa, Yusu Wang, and Jungwoo Yang. Maintaining contour trees of dynamic terrains. In *Proc. 31st International Symposium on Computational Geometry (SoCG)*, pages 796–811, 2015.

2 Hamish Carr, Jack Snoeyink, and Ulrike Axen. Computing contour trees in all dimensions. In *Proc. 11th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 918–926, 2000.

3 Hamish Carr, Jack Snoeyink, and Michiel van de Panne. Simplifying flexible isosurfaces using local geometric measures. In *Proc. 15th IEEE Visualization Conference (VIS)*, pages 497–504, 2004.

4 Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. *Discrete Computational Geometry*, 28:511–533, 2002.

5 Matthew Hiatt, Willem Sonke, Elisabeth Addink, Wout van Dijk, Marc van Kreveld, Tim Ophelders, Kevin Verbeek, Joyce Vlaming, Bettina Speckmann, and Maarten Kleinhans. Geometry and topology of estuary and braided river channel networks extracted from topographic data. Abstract EP32A-08 presented at 2018 Fall Meeting, AGU, Washington, D.C., 10-14 Dec.

6 Maarten Kleinhans, Marc van Kreveld, Tim Ophelders, Willem Sonke, Bettina Speckmann, and Kevin Verbeek. Computing representative networks for braided rivers. In *Proc. 33rd International Symposium on Computational Geometry (SoCG)*, pages 48:1–48:16, 2017.

7 Daniel Sleator and Robert Tarjan. A data structure for dynamic trees. *Journal of Computer and System Sciences*, 26:362–391, 1983.

8 Wout van Dijk, Jasper Leuven, Jana Cox, Jelmer Cleveringa, Marcel Taal, Matthew Hiatt, Willem Sonke, Kevin Verbeek, Bettina Speckmann, and Maarten Kleinhans. The effects of dredging and disposal activity on the resilience of estuary morphodynamics. Abstract EP23C-2305 presented at 2018 Fall Meeting, AGU, Washington, D.C., 10-14 Dec.

Stability analysis of kinetic oriented bounding boxes

Wouter Meulemans[1], Kevin Verbeek[1], and Jules Wulms[1]

1 **Department of Mathematics and Computer Science, TU Eindhoven, the Netherlands**
[w.meulemans|k.a.b.verbeek|j.j.h.m.wulms]@tue.nl

1 Introduction

There exist many applications that desire an algorithm to be *stable*: small changes in the input lead to small changes in the output. Stability is especially important when analyzing or visualizing time-varying data. A sudden change to the input renders an unstable visualization completely ineffective, as it will be hard or impossible to follow the temporal patterns. It is therefore of interest to develop stable algorithms that deal with such changes in the input in an elegant way, so that the output does not lose its effectiveness over time.

Shape descriptors are simplified representations of (complex) shapes and can be used to effectively summarize data, even as it changes over time. They play an important role in fields that rely on shape analysis, like computer vision (shape recognition) [4, 6, 29], computer graphics (bounding boxes for broad-phase collision detection) [1, 12, 16, 24], medical imaging (diagnosis or surgical planning) [7, 13, 15, 30], and machine learning (shape classification) [22, 26, 27, 28]. In this abstract we focus on a shape descriptor that captures the overall orientation of the underlying shape (represented by a point cloud). Specifically, we study the *oriented bounding box* (OBB): an oriented bounding box that contains all points (see Figure 1 left). We say that oriented bounding boxes of smaller area are of better *quality*. Unfortunately, oriented bounding boxes of optimal quality are unstable and may have discrete "flips" in their orientation, even for continuously moving point sets (Figure 1 right). Hence, we want to sacrifice some quality for stability.

Problem description. The main goal of this abstract is to formally analyze the trade-off between quality and stability for an OBB. Our input consists of a set of n moving points $P = P(t) = \{p_1(t), \ldots, p_n(t)\}$ in 2 dimensions, where each $p_i(t)$ is a function $p_i \colon [0, T] \to \mathbb{R}^2$. We assume that, at each time t, not all points are at the same position. We further assume that each point moves with at most unit speed, that is, $\|p_i'(t)\| \leq 1$ for all times t. As output we want to compute an OBB containing all the input points. The oriented bounding box essentially describes an optimization function, which captures the quality of the shape descriptor, that is, how well the descriptor represents (the orientation of) the underlying shape. The optimization function is the area of the bounding box and it can be minimized over all orientations. That is, we consider the optimization of function $f_{\text{OBB}}(\alpha, P)$ which represents the area of the smallest bounding box with orientation α on point set P.

We can now also consider bounding boxes of suboptimal quality, with slightly larger area than the smallest one. This allows us to make a trade-off between quality and other desirable properties of OBB, such as its stability. Since the optimization function optimizes over orientations, the output consists of an orientation $\alpha(t)$ for every time step $t \in [0, T]$. This orientation is an element of the real projective line $\mathbb{RP}^1$, but we represent $\alpha(t)$ by a unit vector in $\mathbb{R}^2$ and implicitly identify opposite vectors, which is equivalent. Furthermore, we assume that the output $\alpha(t)$ is computed for all real values $t \in [0, T]$.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

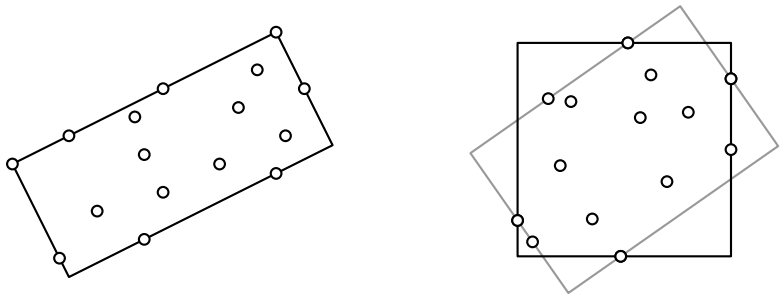


Figure 1 Examples of OBB on different point sets. On the right two optimal boxes exist.

For OBB we typically compute more than just an orientation. However, it is easy to see that the stability of OBB is mostly affected by the optimal orientation. We therefore ignore other aspects of the shape descriptor to analyze the stability, and assume that these aspects are chosen optimally for the given orientation without any cost with regard to the stability.

Stability analysis. To analyze the stability of OBB we use the framework introduced by Meulemans et al. [18]. This framework defines various stability measures, one of which is the Lipschitz stability. An algorithm is *K-Lipschitz stable* if its output changes at most K units, when the input points move one unit. We study the Lipschitz stability of OBB and slightly reformulate the definition by restricting it to our setting. Let $\mathcal{A}$ be an algorithm that takes a point set as input and computes an orientation. The definition of the Lipschitz stability ratio depends on the metrics (distance functions) on the input and output space. This is straightforward: the input space is $\mathbb{R}^{2n}$ and the output space is topologically equivalent to S^1, hence we can use the Euclidean and circle metric respectively. We define the K-Lipschitz stability ratio of OBB as follows:

$$\rho_{LS}(\text{OBB}, K) = \inf_{\mathcal{A}} \sup_{P(t)} \max_{t\in[0,T]} \frac{f_{\text{OBB}}(\mathcal{A}(P(t)), P(t))}{\min_\gamma f_{\text{OBB}}(\gamma, P(t))} \tag{1}$$

where the infimum is taken over all algorithms $\mathcal{A}$ for which $\mathcal{A}(P(t))$ is K-Lipschitz. Thus, the difference in orientation or angle is at most K radians per unit of change in the input. Informally, ρ_{LS} is the best approximation ratio any algorithm can achieve for a given K.

Kinetic algorithms. Algorithms for kinetic (moving) input can adhere to different models, which may influence the results of the stability analysis. We focus on *state-aware* algorithms, which map input $I(t)$ on time t to output, but also maintain a state S (typically the output at the previous time step) over time. This contrasts *stateless* and *clairvoyant* algorithms, which depend only on the input $I(t)$ at a particular time or the complete function $I(t)$ respectively.

Interestingly, for stateless algorithms the K-Lipschitz stability ratio is unbounded for any constant K. This can be argued topologically by realizing that a stable stateless algorithm defines a continuous map between the input and output space, and we can show that an algorithm with bounded approximation ratio must define a map between two subspaces that are not homotopic. We therefore move our attention to state-aware algorithms.

Our results. A state-aware algorithm can enforce continuity by keeping the output at the previous time step as the state. We define a *chasing algorithm*, which moves the output of the previous time step with maximum allowed speed towards the optimal solution at the current time. In the remainder of this abstract we analyze the quality and stability of such

an algorithm: Section 2.1 explains how stability is ensured, while Section 2.2 shows how the quality is affected.

Related work. Computing OBB for static point sets is a classic problem in computational geometry. In two dimensions, one side of the optimal box aligns with a side of the convex hull and it can be computed in linear time after finding the convex hull [11, 23]; a similar property holds in three dimensions, allowing a cubic-time algorithm [19]. The relevance of bounding boxes in 3D as a component of other algorithms led to efficient approximation algorithms, such as a $(1+\epsilon)$-approximation algorithm in $O(n+1/\epsilon^{4.5})$ time or an easier algorithm with running time $O(n \log n + n/\epsilon^3)$ [2]. Bounding boxes find applications in tree structures for spatial indexing [3, 14, 20, 21] and in collision detection and ray tracing [1, 12, 24]. Results on the stability of facility location problems [5, 9, 10, 8] and the medial axis [17] all predate a framework for analyzing stability introduced by Meulemans et al. [18]. In [18] the authors apply the framework to maintaining the Euclidean minimum spanning tree on a set of moving points. They show a bounded topological stability for various topologies on the space of spanning trees, and that the Lipschitz stability is at most linear, but also at least linear if the allowed speed for the changes in the tree is too low. Bounds on the topological stability of a problem essentially are lower bounds on its Lipschitz stability. Van der Hoog et al. study the topological stability of the k-center problem [25], showing upper and lower bounds on the stability ratio for various measures. They also provide an algorithm to determine the best ratio attainable for a given set of moving points.

2 Lipschitz stability of a state-aware algorithm

To derive meaningful bounds on the Lipschitz stability ratio, the relation between distances/speeds in input and output space should be *scale-invariant* [18]. This is currently not the case: if we scale the coordinates of the points, then the distances in the input space change accordingly, but the distances in the output space (between orientations) do not. To remedy this problem, we require that diameter D of $P(t)$ is at least 1 for every time t.

We use a chasing algorithm as introduced in Section 1. However, instead of chasing the orientation of OBB, we chase the orientation of a diametrical pair. Although chasing the optimal shape descriptor would be better in general, chasing a diametrical pair is easier to analyze and sufficient to obtain a bounded Lipschitz stability ratio for OBB.

2.1 Chasing the diametrical pair

We denote the orientation of the diametrical pair as $\alpha = \alpha(t)$ and the diameter as $D = D(t) \geq 1$. Furthermore, let $W = W(t)$ be the width of the thinnest strip with orientation $\alpha(t)$ covering all points in $P(t)$, and let $z = z(t) = W(t)/D(t)$ be the *aspect ratio* of the *diametric box* with orientation $\alpha(t)$. We will generally omit the dependence on t if t is clear from the context. Finally, we have a chasing algorithm that has orientation $\beta = \beta(t)$ and the difference in orientation is at most a constant K, when the input has moved one unit.

Approach. The main goal is to keep β as close to α as possible, specifically within a sufficiently small interval around α. The challenge lies with the discrete flips of α. We must argue that, although flips can happen instantaneously, they cannot happen often within a short time-span – otherwise we can never keep β close to α with a bounded speed. Furthermore, the size of the interval must depend on the aspect ratio z, since if $z = 0$, the interval around α must have zero size as well to guarantee a bounded approximation ratio.

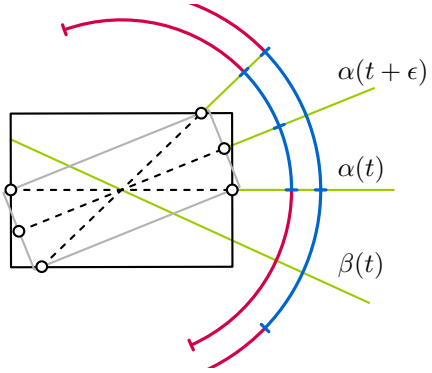


Figure 2 Interval I at time t (outer) and time $t+\epsilon$ (inner). The safe/danger zones and orientations are indicated in blue, red and green respectively. Diametrical pairs are connected by dashed lines.

For the analysis we introduce three functions depending on z: $T(z)$, $H(z)$, and $J(z)$. Function $H(z)$ defines an interval $[\alpha - H(z), \alpha + H(z)]$ called the *safe zone*. We aim to show that, if β leaves the safe zone at some time t, it must return to the safe zone within the time interval $(t, t + T(z)]$. We also define a larger interval $I = [\alpha - H(z) - J(z), \alpha + H(z) + J(z)]$. We refer to the parts of I outside of the safe zone as the *danger zone* (see Figure 2). Although β may momentarily end up in the danger zone due to discontinuous changes, it must quickly find its way back to the safe zone. We aim to guarantee that β stays within I at all times. Let $E = E(t)$ refer to an endpoint of I. We call $J(z)$ the *jumping distance* and say it is valid if $J(z)$ upper bounds how far E can "jump" in a single time step. Note that $J(z)$ is defined recursively through E, so we need to be careful to choose the right function for $J(z)$. For the other functions we choose $T(z) = z/4$ and $H(z) = c\arcsin(z)$ for a constant c (chosen later).

Changes in orientation and aspect ratio. To verify that the chosen functions $T(z)$ and $H(z)$ satisfy the intended requirements, and to define the function $J(z)$, we need to bound how much α and z can change over a time period of length Δt. We refer to these bounds as $\Delta\alpha(z, \Delta t)$ and $\Delta z(z, \Delta t)$, respectively. Note that, since the diameter can change discontinuously, we generally have that $\Delta\alpha(z, 0) > 0$ and $\Delta z(z, 0) > 0$.

▶ **Lemma 2.1.** $\Delta\alpha(z, \Delta t) \leq \arcsin(z + \Delta t(1 + z))$ *for* $\Delta t \leq (1 - z)/(1 + z)$.

▶ **Lemma 2.2.** $\Delta z(z, \Delta t) \leq z - \frac{\sin(\frac{1}{2}\arcsin(z)) - 2\Delta t}{1+2\Delta t}$ *for* $\Delta t \leq \sin(\frac{1}{2}\arcsin(z))/2$.

Jumping distance. Using Lemma 2.1 and Lemma 2.2 we can derive a valid function for $J(z)$. Recall that we require that $J(z)$ is at least the amount E can move in $\Delta t = 0$ time.

▶ **Lemma 2.3.** $J(z) = (c + 2)\arcsin(z)$ *is a valid jumping distance function.*

Proof. By Lemma 2.1 and Lemma 2.2 we get that $\Delta E(z, 0) \leq \Delta\alpha(z, 0) + H(z) - H(z - \Delta z(z, 0)) + J(z) - J(z - \Delta z(z, 0))$. Since $\Delta\alpha(z, 0) \leq \arcsin(z)$ and $\Delta z(z, 0) \leq z - \sin(\frac{1}{2}\arcsin(z))$, we get after simplification that $\Delta E(z, 0) \leq (1 + c/2)\arcsin(z) + J(z) - J(\sin(\frac{1}{2}\arcsin(z)))$. Since we require that $J(z) \geq \Delta E(z, 0)$, it suffices to show that the following holds: $J(\sin(\frac{1}{2}\arcsin(z))) \geq (1 + c/2)\arcsin(z)$. Using the provided function, we get that $J(\sin(\frac{1}{2}\arcsin(z))) = (c + 2)\arcsin(z)/2$ as required. ◀

▶ **Corollary 2.4.** *If* β *is in* I*, then* $|\alpha - \beta| \leq (2c + 2)\arcsin(z)$.

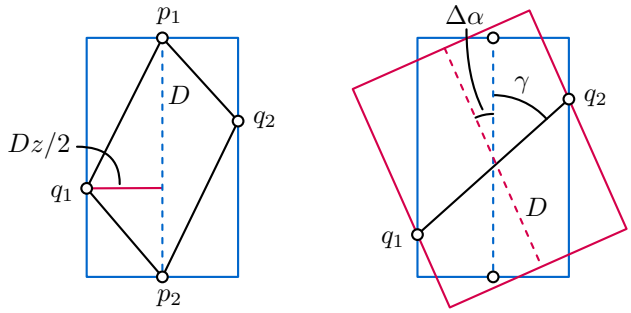


Figure 3 Illustrations supporting proof of Lemma 2.7.

Bounding the speed. To show that the orientation β stays within the interval I, we argue that over a time period of $T(z)$ we can rotate β at least as far as E. As the endpoint of the safe zone moves at most as fast as E, this implies that if β leaves the safe zone at time t, it returns to it in the time period $(t, t + T(z)]$. Thus we require that $KT(z) \geq \Delta E(z, T(z))$. We need to keep up only when the safe zone does not span all orientations, that is, the above inequality must hold only when $H(z) \leq \pi/2$ or $z \leq \sin(\frac{\pi}{2c})$. For the following speed bound we choose a specific value $c = 3$. Hence we only need to chase α when $z \leq \sin(\frac{\pi}{6}) = \frac{1}{2}$.

▶ **Lemma 2.5.** *If $K \geq 40$, then $|\beta(t) - \alpha(t)| \leq 8 \arcsin(z)$ (using $c = 3$) for all times t.*

2.2 Lipschitz stability ratio

What remains is to analyze the approximation ratio of the chasing algorithm for OBB. Corollary 2.4 implies that the orientation β of the chasing algorithm is at most an angle $(2c + 2) \arcsin(z)$ away from the orientation of a diametrical pair of points.

▶ **Lemma 2.6.** $\sin(\lambda \arcsin(x)) \leq \lambda x$ *for* $\lambda \geq 1$ *and* $0 \leq x \leq 1$*:*

▶ **Lemma 2.7.** *If $|\beta - \alpha| \leq (2c + 2) \arcsin(z)$, then $f_{\text{OBB}}(\beta, P) \leq (4c + 6) \min_x f_{\text{OBB}}(x, P)$.*

Proof. Assume that at some time t we have a diametric box with diameter D and aspect ratio z, and let (p_1, p_2) be a diametrical pair. The smallest OBB must contain p_1 and p_2 and must hit the sides of the diametric box at, say, q_1 and q_2 (see Figure 3). Since the smallest OBB must contain the triangles formed by $\{p_1, p_2, q_1\}$ and $\{p_1, p_2, q_2\}$, the area of this box must be at least the sum of the areas of these two triangles, which is $D^2 z/2$.

Now consider the box of the chasing algorithm, where $\Delta\alpha = |\beta - \alpha| \leq (2c + 2) \arcsin(z)$. We assume that the major axis of the box has length D, which is worst possible. Let the minor axis of the box be bounded by two points q_1 and q_2, where the angle between the line through q_1 and q_2 and the line through the diametrical pair is γ. Note that the distance between q_1 and q_2 is bounded by $\min(D, zD/\sin(\gamma))$. The angle between the minor axis of the box and the line through q_1 and q_2 is $\pi/2 - \gamma - \Delta\alpha$. Thus, the length of the minor axis is $\min(D, zD/\sin(\gamma)) \cos(\pi/2 - \gamma - \Delta\alpha) = \min(D \sin(\gamma + \Delta\alpha), zD \sin(\gamma + \Delta\alpha)/\sin(\gamma))$. Since the function $\sin(\gamma + \Delta\alpha)/\sin(\gamma)$ is decreasing in γ, we attain the maximum when $z/\sin(\gamma) = 1$ or $\gamma = \arcsin(z)$. Hence, the area of this box is at most $D^2 \sin((2c + 3) \arcsin(z))$, which is at most $D^2 z(2c + 3)$ by Lemma 2.6. Thus, $f_{\text{OBB}}(\beta, P) \leq (4c + 6) min_x f_{\text{OBB}}(x, P)$. ◀

By combining Lemmata 2.5 and 2.7, we obtain this bound on the Lipschitz stability of OBB.

▶ **Theorem 2.8.** *The Lipschitz stability ratio for* OBB *is bounded by* $\rho_{\text{LS}}(\text{OBB}, 40) \leq 18$.

References

1 Gill Barequet, Bernard Chazelle, Leonidas Guibas, Joseph Mitchell, and Ayellet Tal. BOX-TREE: A hierarchical representation for surfaces in 3d. *Comput. Graph. Forum*, 15(3):387–396, 1996.

2 Gill Barequet and Sariel Har-Peled. Efficiently approximating the minimum-volume bounding box of a point set in three dimensions. *J. Algorithms*, 38(1):91–109, 2001.

3 Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In *Proc. 1990 ACM SIGMOD International Conference on Management of Data*, pages 322–331, 1990.

4 Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(4):509–522, 2002.

5 Sergei Bespamyatnikh, Binay Bhattacharya, David Kirkpatrick, and Michael Segal. Mobile facility location. In *Proc. 4th Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 46–53, 2000.

6 Michael Bronstein and Iasonas Kokkinos. Scale-invariant heat kernel signatures for non-rigid shape recognition. In *23rd IEEE Conference on Computer Vision and Pattern Recognition*, pages 1704–1711, 2010.

7 Dragana Brzakovic, Xiao Mei Luo, and P. Brzakovic. An approach to automated detection of tumors in mammograms. *IEEE Transactions on Medical Imaging*, 9(3):233–241, 1990.

8 Mark de Berg, Marcel Roeloffzen, and Bettina Speckmann. Kinetic 2-centers in the black-box model. In *Proc. 29th Symposium on Computational Geometry*, pages 145–154, 2013.

9 Stephane Durocher and David Kirkpatrick. The steiner centre of a set of points: Stability, eccentricity, and applications to mobile facility location. *International Journal of Computational Geometry & Applications*, 16(04):345–371, 2006.

10 Stephane Durocher and David Kirkpatrick. Bounded-velocity approximation of mobile Euclidean 2-centres. *International Journal of Computational Geometry & Applications*, 18(03):161–183, 2008.

11 Herbert Freeman and Ruth Shapira. Determining the minimum-area encasing rectangle for an arbitrary closed curve. *Commun. ACM*, 18(7):409–413, 1975.

12 Stefan Gottschalk, Ming Lin, and Dinesh Manocha. OBBTree: A hierarchical structure for rapid interference detection. In *Proc. 23rd Annual Conference on Computer Graphics and Interactive Technique*, pages 171–180, 1996.

13 Xianfeng Gu, Yalin Wang, Tony Chan, Paul Thompson, and Shing-Tung Yau. Genus zero surface conformal mapping and its application to brain surface mapping. *IEEE Trans. Med. Imaging*, 23(8):949–958, 2004.

14 Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In *Proc. 1984 ACM SIGMOD International Conference on Management of Data*, pages 47–57, 1984.

15 András Kelemen, Gábor Székely, and Guido Gerig. Elastic model-based segmentation of 3-D neuroradiological data sets. *IEEE Trans. Med. Imaging*, 18(10):828–839, 1999.

16 James Klosowski, Martin Held, Joseph Mitchell, Henry Sowizral, and Karel Zikan. Efficient collision detection using bounding volume hierarchies of k-DOPs. *IEEE Trans. Vis. Comput. Graph.*, 4(1):21–36, 1998.

17 Kyle Sykes David Letscher and Kyle Sykes. On the stability of medial axis of a union of disks in the plane. In *Proc. 28th Canadian Conference on Computational Geometry*, pages 29–33, 2016.

18 Wouter Meulemans, Bettina Speckmann, Kevin Verbeek, and Jules Wulms. A framework for algorithm stability and its application to kinetic euclidean MSTs. In *Proc. 13th LATIN*, LNCS 10807, pages 805–819, 2018.

19 Joseph O'Rourke. Finding minimal enclosing boxes. *International Journal of Parallel Programming*, 14(3):183–199, 1985.

20 Nick Roussopoulos and Daniel Leifker. Direct spatial search on pictorial databases using packed r-trees. In *Proc. 1985 ACM SIGMOD International Conference on Management of Data*, pages 17–31, 1985.

21 Timos Sellis, Nick Roussopoulos, and Christos Faloutsos. The R+-tree: A dynamic index for multi-dimensional objects. In *Proc. 13th International Conference on Very Large Data Bases*, pages 507–518, 1987.

22 Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3D shape recognition. In *2015 IEEE International Conference on Computer Vision*, pages 945–953, 2015.

23 Godfried Toussaint. Solving geometric problems with the rotating calipers. In *Proc. IEEE Melecon*, volume 83, page A10, 1983.

24 Gino van den Bergen. Efficient collision detection of complex deformable models using AABB trees. *J. Graphics, GPU, & Game Tools*, 2(4):1–13, 1997.

25 Ivor van der Hoog, Marc van Kreveld, Wouter Meulemans, Kevin Verbeek, and Jules Wulms. Topological stability of kinetic k-centers. *CoRR*, abs/1810.00794, 2018.

26 Manik Varma and Debajyoti Ray. Learning the discriminative power-invariance trade-off. In *Proc. IEEE 11th International Conference on Computer Vision*, pages 1–8, 2007.

27 Jin Xie, Guoxian Dai, Fan Zhu, Edward Wong, and Yi Fang. DeepShape: Deep-learned shape descriptor for 3D shape retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(7):1335–1345, 2017.

28 Hao Zhang, Alexander Berg, Michael Maire, and Jitendra Malik. SVM-KNN: discriminative nearest neighbor classification for visual category recognition. In *Proc. 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2126–2136, 2006.

29 Yu Zhong. Intrinsic shape signatures: A shape descriptor for 3d object recognition. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference*, pages 689–696. IEEE, 2009.

30 Barbara Zitová and Jan Flusser. Image registration methods: a survey. *Image Vision Comput.*, 21(11):977–1000, 2003.

A Poisson sample of a smooth surface is a good sample

Olivier Devillers[1] and Charles Duménil[1]

1 **Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France**
Olivier.Devillers@inria.fr, charles.dumenil@inria.fr

Abstract

The complexity of the 3D-Delaunay triangulation (tetrahedralization) of n points distributed on a surface ranges from linear to quadratic. When the points are a deterministic good sample of a smooth compact generic surface, the size of the Delaunay triangulation is $O(n \log n)$ [2]. Using this result, we prove that when points are Poisson distributed on a surface under the same hypothesis, whose expected number of vertices is λ, the expected size is $O(\lambda \log^2 \lambda)$.

1 Introduction

While the complexity of the Delaunay triangulation of n points is strictly controlled in two dimensions to be between n and $2n$ triangles (depending on the size of the convex hull) the gap between the lower and upper bound ranges from linear to quadratic in dimension 3. The worst case is obtained using points on the moment curve[1] and the best case by using the center of spheres defining a packing.[2]

To get a more precise result on the size of the 3D Delaunay triangulation, it is possible to make different kinds of hypotheses on the point set. A first possibility is to assume a random distribution in 3D and if the points are evenly distributed in a sphere [6], (resp. in a cube [3]), Dwyer (resp. Bienkoswski et al.) proved that the expected size is $\Theta(n)$. But this hypothesis of random distribution is not relevant for all applications, for example when dealing with 3D reconstruction the Delaunay triangulation is an essential tool and it is much more natural to assume that the points are not distributed in space but on a surface [4]. If the points are evenly distributed on the boundary of a polyhedron, the expected size was proved to be $\Theta(n)$ in the convex case [9] and between $\Omega(n)$ and $\tilde{O}(n)$ in the non convex case by Golin and Na [8].

Instead of using probabilistic hypotheses one can assume that the points are a good sampling of the surface, namely an (ϵ, η)-sample where any ball of radius ϵ centered on the surface contains at least one and at most η points of the point-set. Under such hypothesis Attali and Boissonnat proved that the complexity of the Delaunay triangulation of a polyhedron is linear [1]. Attali, Boissonnat, and Lieutier extend this result to smooth surfaces verifying some genericity hypotheses with an upper bound of $O(n \log n)$ [2]. The genericity hypothesis is crucial since Erickson proved that there exists good sample of a cylinder with a triangulation of size $\Omega(n\sqrt{n})$ [7]. In the example by Erickson the point set is placed in a very special position on an helix, nevertheless, even with an unstructured point set it is possible to reach a supra-linear triangulation since Erickson, Devillers, and Goaoc proved that the triangulation of points evenly distributed on a cylinder has expected size $\Theta(n \log n)$ [5].

[1] The moment curve is parameterized by (t, t^2, t^3). When computing the Delaunay triangulation of points on this curve, any pair of points define a Delaunay edge.

[2] The kissing number in 3D is 12, thus in such a point set, the number of edges is almost $6n$.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 19–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

Contribution

In this paper we prove that a Poisson sample of parameter λ on a smooth surface of finite area is an (ε, η)-sample for $\varepsilon = 3\sqrt{\frac{\log \lambda}{\lambda}}$ and $\eta = 1000 \log \lambda$ with high probability. Using the result of Attali, Boissonnat, and Lieutier, it yields that the complexity of the Delaunay triangulation of a Poisson sample of a generic surface is $O(\lambda \log^2 \lambda)$ losing an extra logarithmic factor with respect to the case of good sampling (see Section 3).

2 Notation, definitions, previous results

We consider a surface Σ embedded in $\mathbb{R}^3$, compact, smooth, oriented and without boundary. At a point $p \in \Sigma$, for a given orientation, we denote by $\kappa_1(p)$ and $\kappa_2(p)$ the principal curvatures at p with $\kappa_1(p) > \kappa_2(p)$. We assume that the curvature is bounded and define $\kappa_{\sup} = \sup_{p\in\Sigma} \max(|\kappa_1(p)|, |\kappa_2(p)|)$. We denote by $\sigma(p, R)$ the sphere of center p and radius R. We denote by $B(\sigma)$ the closed ball whose boundary is the sphere σ, by $\mathring{E}$ the interior of a set E and, for $p \in \Sigma$, by $D(p, R)$ the intersection between Σ and the $\mathring{B}(\sigma(p, R))$. Abusively we call $D(p, R)$ a disk. For a discrete set X, we denote $\sharp(X)$ the cardinality of X. If X is a set of points, $\mathrm{Del}(X)$ denotes the Delaunay triangulation of X. In the $3D$ case, $\sharp(\mathrm{Del}(X))$ is the sum of the number of tetrahedra, triangles, edges and vertices belonging to the Delaunay triangulation.

Without loss of generality, we assume that $Area(\Sigma) = 1$ and consider that the set of points X is a Poisson point process with parameter $\lambda > 0$ over Σ.

We recall classical properties of a Poisson sample:

▶ **Observation 2.1.** *For two regions R and R' of Σ,*

- $\mathbb{P}[\sharp(X \cap R) = k] = \frac{(\lambda \operatorname{Area}(R))^k}{k!} e^{-\lambda \operatorname{Area}(R)}$,
- $\mathbb{E}[\sharp(X \cap R)] = \lambda \operatorname{Area}(R)$,
- $R \cap R' = \emptyset \Rightarrow \sharp(X \cap R)$ *and* $\sharp(X \cap R')$ *are independent random variables.*

In particular, we have $\mathbb{P}[\sharp(X \cap R) = 0] = e^{-\lambda \operatorname{Area}(R)}$ and $\mathbb{E}[\sharp(X)] = \lambda$.

We consider the same definition of genericity as Attali, Boissonnat and Lieutier, roughly: the set of points where one of the principal curvatures is locally maximal is a finite set of curves whose total length is bounded and, the number of contacts of any medial ball with the surface is finite.

Then we define what is a good-sampling of a surface and precise the result by Attali, Boissonnat and Lieutier.

▶ **Definition 2.2** (Good sample)**.** A point-set on a surface is an (ε, η)-sample if any ball of radius ε centered on the surface contains at least one and at most η points of the sample.

▶ **Theorem 2.3** ([2])**.** *The 3D Delaunay triangulation of an (ε, η)-sample of a generic smooth surface has complexity $O\left(\frac{\eta^2}{\varepsilon^2} \log \frac{1}{\varepsilon}\right)$.*

While the result of Attali et al. provides a bound $O(N \ln N)$ on complexity of the Delaunay triangulation of an (ε, η)-sample of N points and a constant η, by looking more carefully at the result [2, Eq.(14)], we notice that the actual complexity can be expressed by $C(\frac{\eta}{\varepsilon})^2 \log(\varepsilon^{-1})$ for C being a constant of the surface.

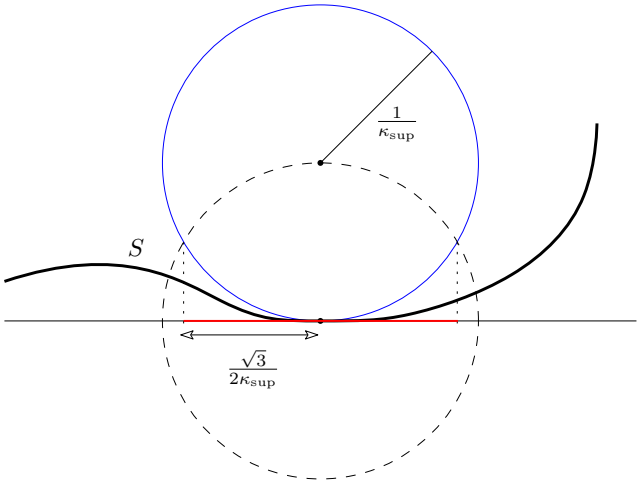


Figure 1 Illustration of the proof of Lemma 3.1 for the 2D case.

3 Is a random sample a good sample?

In a Poisson sampling of parameter λ on the surface, a disk of radius $\varepsilon = \frac{1}{\sqrt{\lambda}}$ is expected to contain π points, but with constant probability it can be empty or contains more than η points. Thus with high probability there will be such disks even if their number is limited. Thus such a sample is likely not to be a good sample with $\varepsilon^2 = \frac{1}{\lambda}$ and η constant. Nevertheless, it is possible to not consider η as a constant, namely, we take $\eta = \Theta(\log(\lambda))$. In a first Lemma, we bound the area of $D(p, R)$, for any $p \in \Sigma$ and $R > 0$ sufficiently small.

▶ **Lemma 3.1.** *Let Σ be a smooth surface of curvature bounded by $\kappa_{\sup}$, and consider $p \in \Sigma$ and $R > 0$ smaller than $\frac{1}{\kappa_{\sup}}$. The area of $D(p, R)$ is greater than $\frac{3}{4}\pi R^2$.*

Proof. The bound is obtained by considering the fact that the surface must stay in between the two tangent spheres of curvature κ_{sup} tangent to the surface at p. The tangent disk at p of radius $\frac{\sqrt{3}}{2}R > \frac{\sqrt{3}}{2}\frac{1}{\kappa_{\sup}}$ is included in the projection of $D(p, r)$ on the tangent plane and thus has a smaller area than $D(p, R)$. ◀

▶ **Lemma 3.2.** *Let Σ be a $\mathcal{C}^3$ surface of curvature bounded by $\kappa_{\sup}$.*
For R small enough, $Area(D(p, R)) < \frac{5}{4}\pi R^2$.

Proof. Let $z = f(x, y) := \frac{1}{2}\kappa_1 x^2 + \frac{1}{2}\kappa_2 y^2 + O(x^3 + y^3)$ be the Monge of Σ patch [10] at a point p. We denote by $d\sigma$ an element of surface and by $A(p, R)$ the projection of $D(p, R)$ on the xy-plane. Since on $D(p, R)$ the slope of the normal to Σ is bounded, we have:

$$Area\,(D(p,R)) = \int_{D(p,R)} d\sigma = \int\int_{A(p,R)} \sqrt{1 + (\frac{\partial f}{\partial x}(x,y))^2 + (\frac{\partial f}{\partial y}(x,y))^2}dxdy$$

That is smaller than $\int\int_{x^2+y^2\leq R^2} \sqrt{1 + (\frac{\partial f}{\partial x}(x,y))^2 + (\frac{\partial f}{\partial y}(x,y))^2}dxdy$, since $D(p, R) \subset B(p, R)$.

Since f is in $\mathcal{C}^3$ and $\frac{\partial f}{\partial x}(x, y) \sim \kappa_1 x$, we can say that there exists a neighborhood of p on which $|\frac{\partial f}{\partial x}| \leq \sqrt{2}\kappa_1|x| \leq \sqrt{2}\kappa_{sup}|x|$, i.e., $\left(\frac{\partial f}{\partial x}\right)^2 \leq 2(\kappa_{sup}x)^2$. Applying the same for y, and turning to polar coordinates, we get:

$$Area(D(p,R)) \leq \int_{\theta=0}^{2\pi}\int_{r=0}^{R} r\sqrt{1 + 2(r\kappa_{\sup})^2}drd\theta = \frac{\pi}{3}\frac{(2(R\kappa_{\sup})^2 + 1)^{\frac{3}{2}} - 1}{\kappa_{\sup}^2}$$

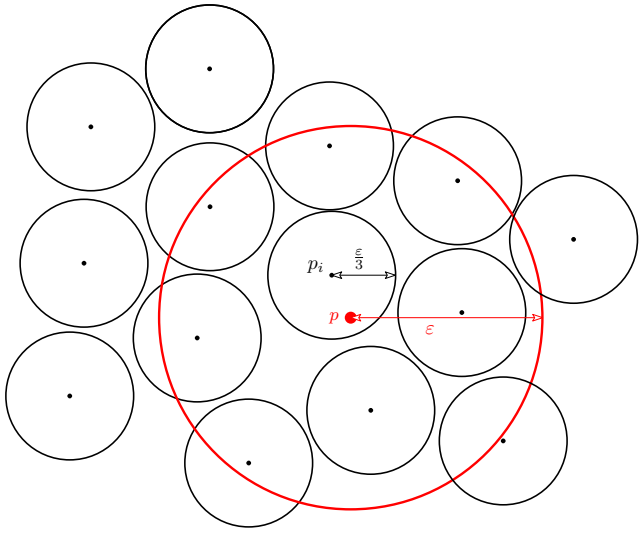


Figure 2 A disk of radius ε always contains a disk of a maximal set of disks of radius $\frac{\varepsilon}{3}$,

Noticing that $(a+1)^{\frac{3}{2}} - 1 = a\frac{a+\sqrt{a+1}+2}{\sqrt{a+1}+1} \leq \frac{15}{8}a$ for $a < 1$, we can conclude that for any R small enough,

$$Area(D(p,R)) \leq \frac{\pi}{3}\frac{\frac{15}{4}(R\kappa_{\sup})^2}{\kappa_{\sup}^2} = \frac{5}{4}\pi R^2.$$

◀

▶ **Lemma 3.3.** *Let Σ be a $\mathcal{C}^3$ surface with $Area(\Sigma) = 1$. Let M_R be a maximal set of k_R disjoint disks $D(p_i, R)$ on Σ. If R is small enough then $k_R \leq \frac{4}{3\pi R^2}$.*

Proof. By Lemma 3.1, for R small enough, we have $D(p,R) \geq \frac{3}{4}\pi R^2$. Thus:

$$k_R \cdot \frac{3}{4}\pi R^2 \leq \sum_{i=1}^{i=k_R} \text{Area}\left(D(p_i,R)\right) \leq \text{Area}\left(\Sigma\right) = 1,$$

and we can deduce the following bound: $k_R \leq \frac{4}{3\pi R^2}$. ◀

▶ **Lemma 3.4.** *Let X be a Poisson sample of parameter λ distributed on a $\mathcal{C}^3$ smooth closed surface Σ of area 1. If λ is large enough, the probability that there exists $p \in \Sigma$ such that $D\left(p, 3\sqrt{\frac{\log\lambda}{\lambda}}\right)$ does not contain any point of X is $O(\lambda^{-1})$.*

Proof. We prove that a Poisson sample has no empty disk of radius $3\sqrt{\frac{\log\lambda}{\lambda}}$ with probability $O(\lambda^{-1})$. In a first part we use a packing argument. On the one hand, for any $\varepsilon > 0$ small enough and given a maximal set $M_{\varepsilon/3}$ and any point $p \in \Sigma$, the disk $D(p,\varepsilon)$ contains entirely one of the disks $D(p_i, \frac{\varepsilon}{3})$ belonging to $M_{\varepsilon/3}$. Indeed, by maximality of $M_{\varepsilon/3}$, the disk $D(p, \varepsilon/3)$ intersects a disk of $M_{\varepsilon/3}$ whose diameter is $2\varepsilon/3$ so $D(p,\varepsilon)$ contains it entirely. On the other hand, remember from Lemma 3.1 that if ε is small enough then $Area(D(p,\varepsilon)) \geq \frac{3}{4}\pi\varepsilon^2$. Then we can bound the probability of existence of an empty disk for ε small enough:

$$\begin{aligned}\mathbb{P}\left[\exists p \in \Sigma,\ \sharp\left(X \cap D(p,\varepsilon)\right) = 0\right] &\leq \mathbb{P}\left[\exists i < k_{\varepsilon/3},\ \sharp\left(X \cap D(p_i,\varepsilon/3)\right) = 0\right] \\ &\leq k_{\varepsilon/3}\,\mathbb{P}\left[\sharp\left(X \cap D(c,\varepsilon/3)\right) = 0\right] \text{ for a point } c \text{ on } \Sigma \\ &\leq \tfrac{4}{3\pi(\varepsilon/3)^2} e^{-\lambda\frac{3}{4}\pi(\frac{\varepsilon}{3})^2} = \tfrac{12}{\pi\varepsilon^2} e^{-\lambda\frac{1\pi\varepsilon^2}{12}}.\end{aligned}$$

By taking $\varepsilon = 3\sqrt{\frac{\log\lambda}{\lambda}}$ we get:

$$\mathbb{P}\left[\exists p \in \Sigma,\ \sharp\left(X \cap D(p, 3\sqrt{\tfrac{\log\lambda}{\lambda}})\right) = 0\right] \leq \tfrac{4\lambda}{3\pi\log\lambda} e^{-\frac{3\pi\log\lambda}{4}} = O(\lambda^{-1}).$$

◀

We have proved that when a Poisson sample is distributed on a surface, the points sufficiently cover the surface, i.e., there is no large empty disk on the surface with high probability. Now we have to verify the other property of a good sample, namely, a Poisson sample does not create large concentration of points in a small area.

▶ **Lemma 3.5.** *Let X be a Poisson sample of parameter λ distributed on a $\mathcal{C}^3$ closed surface of area 1. If λ is large enough, the probability that there exists $p \in \Sigma$ such that $D(p, 3\sqrt{\frac{\log\lambda}{\lambda}})$ contains more than $1000\log(\lambda)$ points of X is $O(\lambda^{-2})$.*

Proof. Consider an M_ε maximal set, we can notice that for any $p \in \Sigma$, the disk $D(p,\varepsilon)$ with $p \in \Sigma$ is entirely contained in one disk $D(p_i, 3\varepsilon)$ that is an augmented disk of M_ε. Indeed, by maximality of M_ε, the disk $D(p,\varepsilon)$ intersects a disk from M_ε say $D(p_j,\varepsilon)$ so $D(p_j, 3\varepsilon)$ contains entirely $D(p,\varepsilon)$.

Then we can bound the probability of existence of a disk containing more than η points:

$$\begin{aligned}\mathbb{P}\left[\exists p \in \Sigma,\ \sharp\left(X \cap D(p,\varepsilon)\right) > \eta\right] &\leq \mathbb{P}\left[\exists i < k_\varepsilon,\ \sharp\left(X \cap D(p_i,3\varepsilon)\right) > \eta\right] \\ &\leq k_\varepsilon\,\mathbb{P}\left[\sharp\left(X \cap D(c,3\varepsilon)\right) > \eta\right] \text{ for a point } c \text{ on } \Sigma \\ &\leq \frac{4}{3\pi\varepsilon^2}\,\mathbb{P}\left[\sharp\left(X \cap D(c,3\varepsilon)\right) > \eta\right]\end{aligned}$$

We use a Chernoff inequality [11] to bound $\mathbb{P}\left[\sharp\left(X \cap D(c,3\varepsilon)\right) > \eta\right]$: If V follows a Poisson law of mean v_0, then $\forall v > v_0$,

$$P(V > v) \leq e^{v-v_0}\left(\frac{v_0}{v}\right)^v.$$

From Lemmas 3.1 and 3.2, we have that: $\frac{27}{4}\pi\varepsilon^2 \leq Area(D(c,3\varepsilon)) \leq \frac{45}{4}\pi\varepsilon^2$ for ε small enough. Consequently we can say that the expected number of points v_0 in $D(c,3\varepsilon)$ verifies $\frac{27}{4}\lambda\pi\varepsilon^2 \leq v_0 \leq \frac{45}{4}\lambda\pi\varepsilon^2$.

Then we apply the above Chernoff bound with $v = \frac{45}{4}e\pi\lambda\varepsilon^2$ (chosen for the convenience of the calculus)

$$\begin{aligned}\mathbb{P}\left[\sharp\left(X \cap D(c,3\varepsilon)\right) > \frac{45}{4}e\pi\lambda\varepsilon^2\right] &\leq e^{\frac{45}{4}e\pi\lambda\varepsilon^2 - v_0}\left(\frac{v_0}{\frac{45}{4}e\pi\lambda\varepsilon^2}\right)^{\frac{45}{4}e\pi\lambda\varepsilon^2} \\ &\leq e^{\frac{45}{4}e\pi\lambda\varepsilon^2 - \frac{27}{4}\pi\lambda\varepsilon^2}\left(\frac{\frac{45}{4}\pi\lambda\varepsilon^2}{\frac{45}{4}e\pi\lambda\varepsilon^2}\right)^{\frac{45}{4}e\pi\lambda\varepsilon^2} = e^{-\frac{27}{4}\pi\lambda\varepsilon^2}\end{aligned}$$

So for $\varepsilon = 3\sqrt{\frac{\log\lambda}{\lambda}}$ and $\eta = \frac{45}{4}e\pi\lambda\varepsilon^2 = \frac{405}{4}e\pi\log\lambda$, we have:

$$\mathbb{P}\left[\exists p \in \Sigma,\ \sharp\left(X \cap D(p, 3\sqrt{\tfrac{\log\lambda}{\lambda}})\right) > \frac{405}{4}e\pi\log\lambda\right] \leq \tfrac{4\lambda}{27\pi\log\lambda}e^{-\frac{243}{4}\pi\log\lambda} = O(\lambda^{-189})$$

Since $\frac{405}{4}e\pi < 1000$, it is sufficient for our purpose to say:

$$\mathbb{P}\left[\exists p \in \Sigma,\ \sharp\left(X \cap D(p, 3\sqrt{\tfrac{\log\lambda}{\lambda}})\right) > 1000\log(\lambda)\right] = O(\lambda^{-2})$$

◀

▶ **Theorem 3.6.** *On a $\mathcal{C}^3$ closed surface, a Poisson sample of parameter λ large enough is a $(3\sqrt{\frac{\log\lambda}{\lambda}}, 1000\log\lambda)$-sample with probability $1 - O(\lambda^{-1})$.*

Proof. From Lemmas 3.4 and 3.5, we have that a Poisson sample is not a $(3\sqrt{\frac{\log\lambda}{\lambda}}, 1000\log\lambda)$-sample with probability $O(\lambda^{-1})$. ◀

▶ **Theorem 3.7.** *For λ large enough, the Delaunay triangulation of a point set Poisson distributed with parameter λ on a closed smooth generic surface of area 1 has $O(\lambda\log^2\lambda)$ expected size.*

Proof. Given a Poisson sample X we distinguish two cases:

- If X is a good sample, i.e., an (ε,η)-sample with $\varepsilon = 3\sqrt{\frac{\log\lambda}{\lambda}}$ and $\eta = 1000\log\lambda$, we apply the $O((\frac{\eta}{\varepsilon})^2\log(\varepsilon^{-1}))$ bound from the paper by Attali et al., that is $O(\lambda\log^2\lambda)$.
- If X is not a good sample, which arises with small probability by Lemma 3.6, we bound the triangulation size by the quadratic bound:

$$\sum_{k\in\mathbb{N}} k^2\,\mathbb{P}\left[\sharp\left(X\right) = k\right] = \sum_{k\in\mathbb{N}} k^2\frac{\lambda^k}{k!}e^{-\lambda} = \lambda(\lambda+1) = O(\lambda^2)$$

Combining the two results, we get

$$\begin{aligned}\mathbb{E}\left[\sharp\left(\mathrm{Del}\left(X\right)\right)\right] &= \mathbb{E}\left[\sharp\left(\mathrm{Del}\left(X\right)\right)|X \text{ good sample}\right]\mathbb{P}\left[X \text{ good sample}\right]\\ &\quad + \mathbb{E}\left[\sharp\left(\mathrm{Del}\left(X\right)\right)|X \text{ not good sample}\right]\mathbb{P}\left[X \text{ not good sample}\right]\\ &\leq O\left(\lambda\log^2\lambda\right)\times 1 + O(\lambda^2)\times O\left(\lambda^{-1}\right) = O(\lambda\log^2\lambda)\end{aligned}$$

◀

References

1 Dominique Attali and Jean-Daniel Boissonnat. A linear bound on the complexity of the Delaunay triangulation of points on polyhedral surfaces. *Discrete and Computational Geometry*, 31:369–384, 2004. doi:10.1007/s00454-003-2870-4.

2 Dominique Attali, Jean-Daniel Boissonnat, and André Lieutier. Complexity of the Delaunay triangulation of points on surfaces: The smooth case. In *Proc. 19th Annual Symposium on Computational Geometry*, pages 201–210, 2003. doi:10.1145/777792.777823.

3 Marcin Bienkowski, Valentina Damerow, Friedhelm Meyer auf der Heide, and Christian Sohler. Average case complexity of voronoi diagrams of n sites from the unit cube. In *EuroCG*, pages 167–170, 2005.

4 Frédéric Cazals and Joachim Giesen. Delaunay triangulation based surface reconstruction. In Jean-Daniel Boissonnat and Monique Teillaud, editors, *Effective Computational Geometry for Curves and Surfaces*, pages 231–276. Springer-Verlag, Mathematics and Visualization, 2006.

5 Olivier Devillers, Jeff Erickson, and Xavier Goaoc. Empty-ellipse graphs. In *Proc. 19th ACM-SIAM Sympos. Discrete Algorithms*, pages 1249–1256, 2008. URL: http://hal.inria.fr/inria-00176204.

6 R. Dwyer. The expected number of k-faces of a Voronoi diagram. *Internat. J. Comput. Math.*, 26(5):13–21, 1993. URL: http://www.sciencedirect.com/science/article/pii/0898122193900687, doi:10.1016/0898-1221(93)90068-7.

7 Jeff Erickson. Dense point sets have sparse Delaunay triangulations or "...but not too nasty". *Discrete & Computational Geometry*, 33:83–115, 2005. doi:10.1007/s00454-004-1089-3.

8 Mordecai J. Golin and Hyeon-Suk Na. The probabilistic complexity of the Voronoi diagram of points on a polyhedron. In *Proc. 18th Annual Symposium on Computational Geometry*, 2002. URL: http://www.cse.ust.hk/~golin/pubs/SCG_02.pdf, doi:10.1145/513400.513426.

9 Mordecai J. Golin and Hyeon-Suk Na. On the average complexity of 3d-Voronoi diagrams of random points on convex polytopes. *Computational Geometry: Theory and Applications*, 25:197–231, 2003. URL: http://www.cse.ust.hk/~golin/pubs/3D_Voronoi_I.pdf, doi:10.1016/S0925-7721(02)00123-2.

10 Peter W Hallinan, Gaile Gordon, Alan L Yuille, Peter Giblin, and David Mumford. *Two- and three-dimensional patterns of the face*. AK Peters/CRC Press, 1999.

11 Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge university press, 2005.

The k-Fréchet distance revisited and extended

Hugo A. Akitaya[1], Maike Buchin[2], Leonie Ryvkin[3], and Jérôme Urhausen[4]

1 **Department of Computer Science, Tufts University**
hugo.alves_akitaya@tufts.edu

2 **Faculty of Computer Science, TU Dortmund**
maike.buchin@tu-dortmund.de

3 **Department of Mathematics, Ruhr-Universität Bochum**
leonie.ryvkin@rub.de

4 **Department of Information and Computing Sciences, Universiteit Utrecht**
j.e.urhausen@uu.nl

Abstract

We recently introduced a new distance measure for polygonal curves, the k-Fréchet distance. It bridges between Hausdorff distance and weak Fréchet distance, and allows us to compare objects of rearranged pieces such as chemical structures or hand-written characters. Here we distinguish between two variants of the k-Fréchet distance, the *cut distance* and the *cover distance*. For the first one, we presented an NP-hardness proof at EuroCG 2018 [6]. The approximation algorithm we presented in that paper, however, only works for the cover distance. Here, we now prove NP-hardness of the cover version and present an XP- as well as an FPT-algorithm for this version.

1 Introduction

During the last decades, several methods for comparing geometrical shapes have been studied in a variety of applications, for example analysing geographic data, such as trajectories, or comparing chemical structures, e.g., protein chains or human DNA. The (weak) Fréchet distance has been well-studied in the past twenty years since it has proven to be helpful in several of the mentioned applications. The Hausdorff distance, another similarity measure, has also proven useful in applications and can be computed more efficiently than the Fréchet distance [2]. However, it provides us with less information by taking only the overall shape of curves into consideration, not how they are traversed.

We introduce both variants of the k-Fréchet distance as distance measures in between Hausdorff and weak Fréchet distance. They allow us to compare shapes consisting of several parts. Variants of partial curve matching using the Fréchet distance have been studied before [4, 5, 9]. Gheibi et al. used the weak Fréchet distance and minimized the length of the subcurves on which backtracking is necessary [8]. For the cover distance, we cover the input curves by at most k (possibly overlapping) subcurves each and ask for a matching of the subcurves such that each matched pair of subcurves has at most weak Fréchet distance ε (for given $\varepsilon > 0$). For the cut version, the subcurves may not overlap. This implies that, as opposed to the cover distance, both curves have to be cut into the same number of subcurves.

The paper is organized as follows: after recalling some basic definitions we formally define both variants of our k-Fréchet distance and state decision and optimization problems. In Section 2 we briefly discuss NP-hardness of the cover distance and sketch the proof. In Section 3 we present an FPT algorithm for the cover distance.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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 Definitions

Recall the weak Fréchet distance [1], a well-known measure for curves, which is defined as follows: For curves $P, Q\colon [0,1] \to [0,1]$, the weak Fréchet distance is given by

$$\delta_{\mathrm{wF}}(P,Q) = \inf_{\sigma,\tau} \max_{t\in[0,1]} \|P(\sigma(t)) - Q(\tau(t))\|,$$

where the reparametrisations $\sigma, \tau\colon [0,1] \to [0,1]$ range over all continuous surjective functions.

A well-known characterisation which is key to efficient algorithms for computing the weak Fréchet distance [1] uses the free space diagram. First we recall 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 and m are the number of segments in P and Q, respectively.

The weak Fréchet distance of two curves is at most a given value ε if there exists a continous path through the free space containing $(0,y), (1,y'), (x,0)$ and $(x',1)$ for some $x, x', y, y' \in [0,1]$. The Hausdorff distance δ_{H} can be characterised as the free space projecting surjectively onto both parameter spaces.

We define further terms connected to the free space diagram below: A *component* of a free space diagram is a connected subset $c \subseteq F_\varepsilon(P,Q)$. A set S 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 said 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 1.1.** For polygonal chains P, Q we define the cut version of the k-Fréchet distance 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 surjective functions that are piecewise defined, such that the images of the continuous parts partition the curve into at most k pieces. Note that σ and τ have to consist of the same number of continous pieces, namely at most k many.

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 can insert the cuts on our curves as horizontal and vertical grid lines. For every row and column of this "cutting grid", we need to select exactly one cell. The cell corresponds to a matched pair of subcurves and therefore needs to contain (a part of) a free space component that projects surjectively onto both subcurves. Next we define the cover distance:

▶ **Definition 1.2.** For polygonal chains P, Q we define the cover version of the k-Fréchet distance as

$$\delta_{\mathrm{cover}}(k,P,Q) = \inf_{\sigma,\tau} \max_{t\in[0,1]} \|P(\sigma(t)) - Q(\tau(t))\|,$$

where σ, τ range over all surjective functions that are piecewise defined, such that the images of the (at most k many) continuous parts may overlap but their union equals the curve.

Thus we define the cover distance as the minimal ε such that there is a set of at most k components of $F_\varepsilon(P,Q)$ that covers both parameter spaces. In other words, we cover the curves P and Q by at most k pieces (i.e., subcurves) such that there is a matching of the pieces where two matched subcurves have small weak Fréchet distance.

For the decision problem of both distance measures, 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).

By definition both variants lie in between Hausdorff and weak Fréchet distance:

$$\delta_{\mathrm{H}}(P,Q) \leq \delta_{\mathrm{cover}}(k,P,Q) \leq \delta_{\mathrm{cut}}(k,P,Q) \leq \delta_{\mathrm{wF}}(P,Q).$$

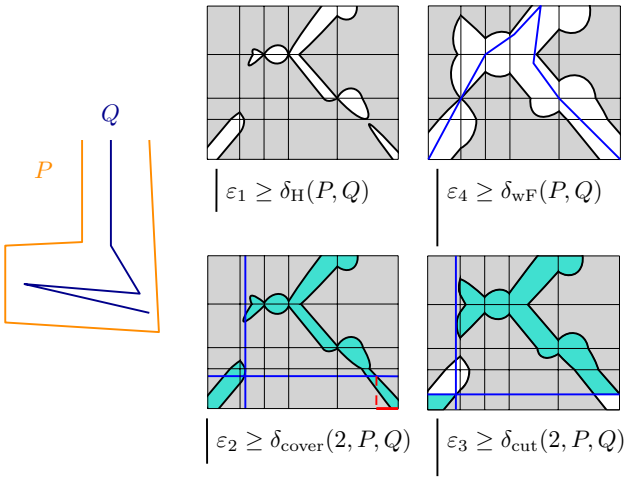


Figure 1 Comparison of distance measures. In the bottom left diagram, two components are sufficient to cover the parameter spaces, but cutting does not work, because by choosing the bottom left and top right cell the red section on the bottom parameter space would not be covered.

2 NP-hardness

In [6], Buchin and Ryvkin proved that the cut distance is NP-complete by reducing from Minimum Common String Partition (MCSP). For the cover distance, we reduce from the following variant of 3-SAT, which was proven to be NP-complete by de Berg and Khosravi [3].

Rectilinear monotone planar 3-SAT:
Input: 3-SAT formula with strictly positive and strictly negative clauses, embedded as a graph with only rectilinear, non-crossing edges; variables are drawn as vertices on a horizontal line, positive clauses are vertices drawn above this line, negative clauses are drawn below;
Output: "Yes" if there exists a satisfying assignment for the variables, "No" otherwise.

Our goal is to construct curves that mimic our input graph (see Figure 2) and show that in the free space resulting from these curves we can find a covering selection of components of size k iff there exists a satisfying assignment for the underlying 3-SAT formula.

2.1 Construction

The construction is intricate, so we only give a brief overview here. A full version of this paper, named "The k-Fréchet distance", will be made available on arXiv.

Overall we create wires and clause gadgets to represent variables and clauses. They are connected as the given embedding of the 3-SAT instance. Wire gadgets allow a boolean

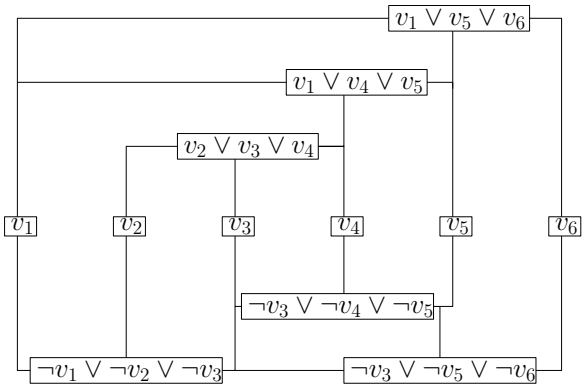


Figure 2 Instance of rectilinear monotone planar 3-SAT.

choice that is propagated consistently throughout the wire. Clause gadgets test whether at least one incoming wire carries an appropriate choice.

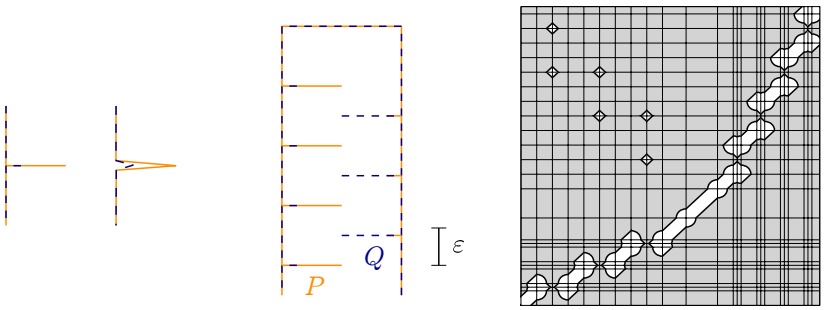


Figure 3 (Left) A spike and a small perturbation of it. (Right) The wire gadget and its corresponding free space diagram. Note that we connected the curves to give a small example, but the horizontal segment on top is not part of the gadget itself.

Figure 3 shows a wire gadget. Both the yellow and the blue curves run along the sides (the vertical parts of the curves, which we call *base curves*) and form *spikes*. The value ε is chosen such that two adjacent spikes are just within distance ε. It follows that the spikes induce components in the free space diagram that form a staircase. We say that a spike s is *covered* by an adjacent spike t of the other curve if the component of the free space diagram that covers the two intervals induced by these spikes is chosen for the covering selection. We choose k such that each blue spike in any gadget can only be covered by one single adjacent yellow spike. The choice for one blue spike must be consistent along a wire and encodes the assignment of the corresponding variable.

Of course, we need a number of other gadgets, too. The wires correspond to edges in the rectilinear monotone planar 3-SAT instance, but to draw them coherently we need to make sure we can make 90° turns (so called *bends*) and do T-crossings, i.e., *split* a wire into two. Last but not least we need to build a *clause gadget* where three wires connect. In Figure 4, we show a bend and a clause gadget, connected through wires. The only basic gadget not shown in Figure 4 is the split, which looks similar to the clause.

Additionally, we need to make sure that both curves are connected and follow the embedding of the input graph G. In order to do so, we establish a number of other gadgets which are explained in the full version of this paper, where we also explain how to connect all gadgets and build the curves.

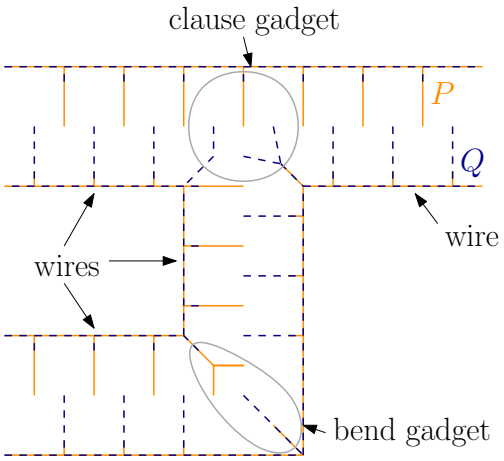


Figure 4 Small example part of the construction consisting of wires, a bend and a clause gadget.

2.2 Correctness

▶ **Theorem 2.1.** *For given polygonal curves P and Q, integer k, and $\varepsilon > 0$, it is NP-complete to decide whether $\delta_{\text{cover}}(k, P, Q) \leq \varepsilon$.*

Due to limited space, we only sketch the proof. We set k such that we can cover every blue spike exactly once, so the choice made for one spike is propagated throughout its wire (and corresponding gadgets). Therefore an assignment implies a unique selection of components and vice versa; following the choices made for the blue spikes we can derive a variable assignment for the underlying 3-SAT formula.

We can test in polynomial time whether the union of a selection of components covers the parameter spaces. Thus the problem of deciding the cover distance lies in NP.

3 Algorithmic approaches

3.1 Earlier results

In [6], Buchin and Ryvkin presented a straight-forward XP-algorithm that simply checked for all possible selections of size k, whether one of them covered both parameter sizes, which takes $\mathcal{O}(k \cdot n^{2k})$ time. As obvious, this approach does not apply to the cut version of the k-Fréchet distance: a selection of components does not imply cutting the curves, the subcurves may overlap. Cutting one curve affects the possible cuts of the other one, so determining correct cuts even for a given selection of components is non-trivial.

In the same paper the authors also gave an approximation algorithm for the optimization problem: by sweeping twice to determine the optimal selections to cover either parameter space the algorithm outputs a selection of components that is at worst doubly the size of an optimal one. Again, this only works for the cover distance, not for cutting.

3.2 Fixed-parameter tractability

Next we present an algorithm for deciding whether $\delta_{\text{cover}}(k, P, Q) \leq \varepsilon$ for given ε and k. The runtime of our algorithm is polynomial in the complexity of our curves P and Q, but exponential in the two parameters k (the selection size) and z (the neighborhood complexity).

We define the *neighborhood complexity* z of P and Q as the maximum number of segments of one curve that intersect with the ε-neighborhood of any point of the other curve. Now, in the free space diagram each horizontal and each vertical line intersects at most z components.

First of all, we build two bounded search trees T_P and T_Q (as described in Chapter 3 of [7]), see Figure 5 below. Consider the tree T_P. The root is labelled by the left boundary point of the parameter space of P (we assume without loss of generality that the bottom boundary of the free space diagram corresponds to P). Now we use a sweep line initialized at the left boundary of the free space diagram. We assign a node in the tree to all components intersecting the sweep line, so the root has as many children as there are components touching the left boundary. Whenever the sweep line is tangent to a component C, said component either becomes *active* (sweep line touches the leftmost point) or *inactive* (its rightmost point is touched). Becoming active does not immediately affect the tree; becoming inactive results in new entries in the tree: for every node labelled C we insert a child for each currently active component. The next time a compone By definition a node can never have more than z children. Also, with every node we store its depth (the root has depth 0) and stop assigning children at depth k - or as soon as a component touches the right boundary of the free space diagram. If a leaf v_l corresponds to a component touching the right boundary, the path from the root to v_l encodes a *feasible* selection of components for T_P. The tree T_Q is built analogously by sweeping from bottom to top.

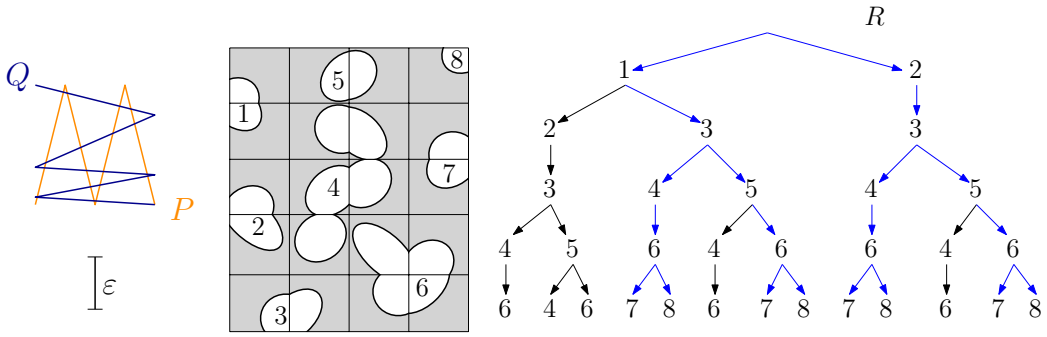


Figure 5 Curves P, Q, their free space diagram and the tree T_P. Feasible paths are marked blue.

We store the feasible selections obtained from T_P and T_Q in sorted lists L_P^T and L_Q^T. For each pair of selections $S_{P,i}$, $S_{Q,j}$, where $1 \leq i, j \leq z^k$, we test whether $|S_{P,i} \cup S_{Q,j}| \leq k$ and output this union if the answer is positive. The proof of the following theorem can also be found in the full version.

▶ **Theorem 3.1.** *The algorithm described above returns a selection S of k components in the free space that surjectively projects onto both parameter spaces if and only if such a selection exists. Therefore it decides whether $\delta_{\text{cover}}(k, P, Q) \leq \varepsilon$ in time $\mathcal{O}(nk + kz^{2k})$.*

References

1 Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *Internat. J. Comput. Geom. Appl.*, 5(1-2):75–91, 1995.

2 Helmut Alt, Christian Knauer, and Carola Wenk. Comparison of distance measures for planar curves. *Algorithmica*, 38(1):45–58, 2004.

3 Mark de Berg and Amirali Khosravi. Optimal binary space partitions for segments in the plane. *Internat. J. Comput. Geom. Appl.*, 22(3):187–205, 2012. doi:10.1142/S0218195912500045.

4 Kevin Buchin, Maike Buchin, and Yusu Wang. Exact algorithms for partial curve matching via the Fréchet distance. In *Proc. 20th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '09, pages 645–654, 2009. URL: `http://dl.acm.org/citation.cfm?id=1496770.1496841`.

5 Maike Buchin, Anne Driemel, and Bettina Speckmann. Computing the Fréchet distance with shortcuts is NP-hard. In *Proc. 30th Annual Symposium on Computational Geometry*, SOCG'14, pages 367:367–367:376, 2014. `doi:10.1145/2582112.2582144`.

6 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/`.

7 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*. Springer, Cham, 2015. `doi:10.1007/978-3-319-21275-3`.

8 Amin Gheibi, Anil Maheshwari, Jörg-Rüdiger Sack, and Christian Scheffer. Minimum backward Fréchet distance. pages 381–388, 11 2014. `doi:10.1145/2666310.2666418`.

9 Christian Scheffer. More flexible curve matching via the partial Fréchet similarity. *Int. J. Comput. Geometry Appl.*, 26:33–52, 2016.

Coresets for (k, l)-Clustering under the Fréchet Distance

Maike Buchin and Dennis Rohde

TU Dortmund University
maike.buchin@tu-dortmund.de dennis.rohde@cs.tu-dortmund.de

Abstract

We investigate the problem of clustering a set T of n polygonal curves in $\mathbb{R}^d$ under the Fréchet distance, with respect to the (k, l)-center and the (k, l)-median objective functions. These were recently defined by Driemel et al. as an adaption of the well-known k-center and k-median objectives with the restriction that the center-curves are composed of up to l line segments. Driemel et al. already developed approximation-schemes for these objectives, for $d = 1$. Recently Buchin et al. developed a constant-factor approximation algorithm for the (k, l)-center objective for general d. Further they provide hardness results for that objective. We tie in with these results by providing construction-techniques for small size ε-coresets for the (k, l)-center objective, if the given curves are of well-behaved structure, and for the discrete k-median objective. That is, we restrict the possible center-sets to all subsets of T of cardinality k and thus ignore the restriction on the complexity of the center-curves.

1 Introduction

Clustering is a thoroughly studied topic that has a great impact in the field of data analysis. Every problem in this topic has an intrinsic property: Given a collection P of n objects and an integer k, one wants to divide P into k pieces, the so called clusters, such that the objects in those clusters are some kind of related, cf. [4]. In many problem-formulations each cluster is induced by a representative object. In our setting, these representatives are given by an objective function over which one optimizes. There are three such objective functions that are well-known: k-means, k-median and k-center. Initially these functions were defined in the context of clustering points in the Euclidean space. There are also definitions of the k-median and the k-center in the context of clustering points in general metric spaces.

In our setting, we are given a set T of n polygonal curves in $\mathbb{R}^d$ endowed with the Fréchet distance and an integer k, as well as an integer l. Again we want to divide T into k clusters, i.e., we are looking for a partition of T of cardinality k. Driemel et al. [2] already studied this setting for $d = 1$. They introduce two restrictions, one on the input-curves and one on the representatives, namely the input-curves are composed of up to m line segments each and the representative curves of the clusters are composed of up to l line segments each. The respective objective functions that enforce these restrictions are called (k, l)-center and (k, l)-median. The authors develop quasi linear-time approximation-schemes for these objectives. Recently, Buchin et al. [1] developed a 3-approximate algorithm for the (k, l)-center objective for $d \in \mathbb{N}$, as well as hardness-results, i.e., for $d = 1$ the (k, l)-center is hard to approximate within a factor of $1.5 - \delta$ and for $d > 1$ within a factor of $2.25 - \delta$, for $\delta > 0$.

Let f be one of the objective functions and C be a set of k representative curves. A set S is an ε-coreset for f, if for all choices of C it holds that $|f(T, C) - f(S, C)| \leq \varepsilon \cdot f(T, C)$. Such a coreset is particularly important when clustetring queries shall be answered efficiently, i.e., return the cost for a given center-set C. In this work we give an overview of our results on small cardinality ε-coresets for the (k, l)-center objective and the discrete k-median objective,

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

i.e., we restrict all possible center-sets to the subsets of T of cardinality k and therefore ignore the restriction on the complexity of the center-curves. For a set of line segments we provide ε-coresets of cardinality dependent on $\frac{1}{\varepsilon^{2d}}$, with respect to the (k,l)-center objective. For a set of polygonal curves of complexity at most m each we provide ε-coresets of cardinality dependent on $\frac{l^m}{\varepsilon^{dm}} + m^m$ and the ratio $\frac{\delta}{\alpha}$, where α is the value of a c-approximate solution and δ is the length of a longest line segment of any center-curve associated with that solution, with respect to the (k,l)-center objective, but only if $\frac{\delta}{\alpha} \in \mathcal{O}(\sqrt[2m]{n})$. Finally, for a set of n polygonal curves we provide ε-coresets of cardinality dependent on $\frac{\ln(n)}{\varepsilon^2}$, with respect to the discrete k-median objective. All results presented here stem from the Master thesis of the second author [6] (available on arXiv) and all proofs can be found there.

Related Work To the best of our knowledge, clustering polygonal curves under the (k,l)-center or the (k,l)-median objective has only been studied in [2] and [1], in that order. As it was already mentioned, Driemel et al. introduce the (k,l)-center and the (k,l)-median objective functions. Additionally, they develop $(1+\varepsilon)$-approximation algorithms for these objectives under the restriction that $d = 1$ and ε, k and l are fixed. The algorithms have running-time $\widetilde{\mathcal{O}}(n \cdot m)$. They also provide first hardness results for the (k,l)-center and the (k,l)-median objectives. Finally, they prove that the Fréchet space (Δ, d_F) (a formal definition follows) has unbounded doubling dimension. The 3-approximation algorithm for the (k,l)-center that is developed by Buchin et al. has running-time $\mathcal{O}\left(km(nl\log(l+m)) + m^2\log(m)\right)$. Additional to this algorithm and the already mentioned hardness results they provide similar hardness results for the discrete Fréchet distance and on the minimum enclosing ball problem for polygonal curves under the Fréchet distance.

2 Preliminaries

▶ **Definition 2.1.** A polygonal curve with vertices $v_1, \ldots, v_m \in \mathbb{R}^d$ is defined as the parametric curve connecting each contiguous pair of vertices by a line segment, which we call the edges of the curve. The number of vertices is called complexity of the curve. By Δ_m we denote the equivalence class of polygonal curves of complexity at most m and by $\Delta := \cup_{m\in\mathbb{N}_{\geq 2}}\Delta_m$ we denote the equivalence class of all polygonal curves.

▶ **Definition 2.2.** Let $\mathcal{F}$ be the set of all continuous, injective and non-decreasing functions $f\colon [0,1] \to [0,1]$ with $f(0) = 0$ and $f(1) = 1$. The Fréchet distance between polygonal curves τ and σ is defined as $d_F(\tau,\sigma) = \inf_{f\in\mathcal{F}} \max_{t\in[0,1]} \|\tau(f(t)) - \sigma(t)\|$, where $\|\cdot\|$ is the Euclidean norm.

▶ **Definition 2.3.** Given a set T of n polygonal curves of complexity at most m each and two integers k and l, the (k,l)-center objective is to return the optimal cost of $\min_{C\subset\Delta_l,|C|=k} \max_{\tau\in T} \min_{c\in C} d_F(\tau,c)$. The discrete k-median objective is to return the optimal cost of $\min_{C\subset T,|C|=k} \sum_{\tau\in T} \min_{c\in C} d_F(\tau,c)$.

▶ **Definition 2.4.** Let T be a given set of n polygonal curves. Also, let f be an objective function and C be a set of k cluster-representatives. A set S of polygonal curves is called ε-coreset for T with respect to f, if for all possible choices of C it holds that $|f(T,C) - f(S,C)| \leq \varepsilon \cdot f(T,C)$. S is called *weighted* ε-coreset if every $s \in S$ is assigned a weight $w_s \in \mathbb{R}$, that flows into the value of f.

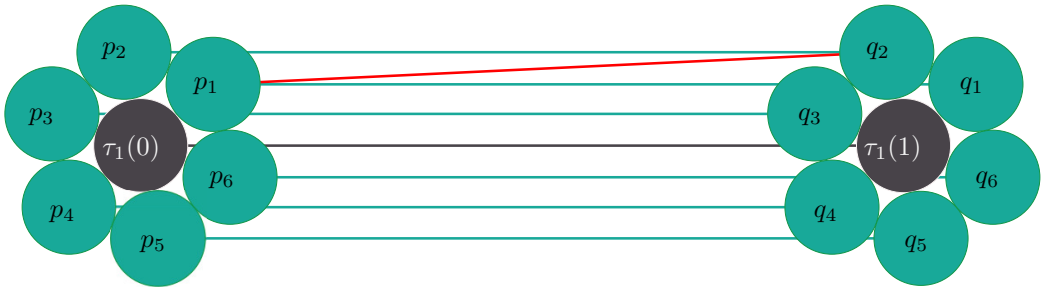


Figure 1 This is the construction used for Theorem 3.1, for $d = 2$. The curves are defined with respect to the center points of the balls. The set T, which cannot be embedded into $(\mathbb{R}^d, d_E)$, where d_E is the Euclidean distance, consists of $\overline{\tau(0)\tau(1)}$ (the common nearest neighbor), the line segments $\overline{p_i q_i}$, for $i \in \{1, \ldots, 6\}$, plus $\overline{p_1 q_2}$. The segment $\overline{p_1 q_2}$ breaks every possible isometric embedding.

3 What is the Difference between Points and Curves?

At first, we investigate whether we can transform the given polygonal curves into points in the Euclidean space through an isometric embedding. Such a transformation would have multiple benefits: When constructing an ε-coreset for any application, often one simply thins out the input-set as much as possible. When such an embedding is available we could apply existing construction-techniques, track which points are thrown out and then throw out all curves that map to these points. If only the value of a clustering is needed this would give us the opportunity to directly obtain the value through one of the numerous algorithms for points in $\mathbb{R}^d$. Unfortunately, such an isometric embedding does not exist for every possible set of polygonal curves, even if we restrict ourselves to line segments.

▶ **Theorem 3.1.** *For any $d \in \mathbb{N}$, there exists a set of polygonal curves in $\mathbb{R}^d$ that cannot be isometrically embedded into $(\mathbb{R}^d, d_E)$, where d_E is the Euclidean distance.*

This result is achieved by proving that an isometric embedding, if existent, would violate the d-dimensional kissing number, given certain sets of polygonal curves, cf. Fig. 1. The d-dimensional kissing number is the maximum number of points in $\mathbb{R}^d$ that can share a common nearest neighbor point (cf. [7]). We note that a similar result is implied due to the fact that certain four-point graphs endowed with the shortest-path metric cannot be embedded into $\mathbb{R}^d$, for any d, while they can be embedded into (Δ_{13}, d_F) with ambient space $\mathbb{R}$, cf. [3]. Nevertheless, our result is stronger because it holds for (Δ_m, d_F), for any $m \geq 2$.

4 Coresets for the (k, l)-center Objective

There is a common technique for constructing ε-coresets for the k-center objective, given a set P of points in the Euclidean space: Run a c-approximate algorithm on P to obtain a value α of the objective function. Let C be the center-set associated with this value. By the structure of the objective function we have that $P \subseteq \cup_{q \in C} \{p \in \mathbb{R}^d \mid \|p - q\| \leq \alpha\} =: E$. Additional, if α^* is the optimal cost for P under the k-center objective, then $\frac{\alpha}{c}$ is a lower bound on this number.

Now around every $q \in C$ we place a grid G_q of edge length $2 \cdot \alpha$, thus $E \subseteq \cup_{q \in C} G_q$. We set the edge-length of the cells of every grid to $\varepsilon \cdot \frac{1}{\sqrt{d}} \cdot \frac{\alpha}{c}$, thus we cannot move a point contained in a cell more than $\varepsilon \cdot \alpha^*$ without leaving the cell. At last we go through every cell of all G_q and if it contains more than one point from P we remove all but one of those from P. The resulting set P' is an ε-coreset for the k-center objective of cardinality $\mathcal{O}\left(\frac{1}{\varepsilon^d}\right)$.

This scheme can easily be adapted for a set T of polygonal curves, utilizing the 6-approximation algorithm by Buchin et al. [1]. For line segments this is particularly easy: Place such a grid around each end point of every center-curve. Again, by the structure of the objective function the end points of the input-curves are contained in those grids. For a curve τ call $\tau(0)$ its initial point and $\tau(1)$ its end point, further call the grid around $\tau(0)$ the initial point grid and the grid around $\tau(1)$ the end point grid. Now, successively for each center-curve, we go through every pair of a cell of the initial point grid and a cell of the end point grid and remove all but one line segment from T that have their initial point, respective end point in those cells. The resulting set T' is an ε-coreset for the $(k,2)$-center objective of cardinality $\mathcal{O}\left(\frac{1}{\varepsilon^{2d}}\right)$.

▶ **Theorem 4.1.** *There exists an algorithm that, given a set of n line segments in d-dimensional Euclidean space and a parameter $\varepsilon \in (0,1)$, computes an ε-coreset for the $(k,2)$-center objective of cardinality $\mathcal{O}\left(\frac{1}{\varepsilon^{2d}}\right)$, in time $\mathcal{O}\left(\frac{n}{\varepsilon^{2d}}\right)$.*

For polygonal curves of complexity at least 3 this scheme has a flaw: The vertices of the input-curves do not necessarily lie within distance α to any vertex of a center-curve. Thus, we have to cover the whole center-curves with grids and therefore the cardinality of the resulting ε-coreset depends on the ratio of roughly the length δ of a longest edge of any center-curve and the value α of the c-approximate solution. For the ε-coreset to have sublinear cardinality we have to check if $\frac{\delta}{2\alpha}$ exceeds, say $\sqrt[2m]{n}$, in advance. If this is the case we are not able to provide an ε-coreset utilizing this technique. If this is not the case we now have to consider any combination of m cells of the grids that cover a center-curve, therefore the cardinality of the resulting ε-coreset is exponential in m.

▶ **Theorem 4.2.** *There exists an algorithm that, given a set of n polygonal curves of complexity at least 3 and at most m each, in d-dimensional Euclidean space and a parameter $\varepsilon \in (0,1)$, computes an ε-coreset for the (k,l)-center objective of cardinality*

$\mathcal{O}\left(2^{3m} \cdot \sqrt{n} \cdot \frac{l^{12d^2m}}{\varepsilon^{dm}} + 2^m m^m\right)$ *in time*

$\mathcal{O}\left(\left(2^{3m} \cdot n^{1.5} \cdot \frac{l^{12d^2m} m}{\varepsilon^{dm}} + 2^m m^{m+1} n\right) + nm\log(m) + m^3\log(m)\right)$, *if* $\frac{\delta}{\alpha} \in \mathcal{O}(\sqrt[2m]{n})$. *Otherwise, the algorithm fails and then has running-time* $\mathcal{O}\left(nm\log(m) + m^3\log(m)\right)$.

5 Coresets for the discrete k-median Objective

For the (discrete) k-median objective we use standard-techniques for approximating sums. In [5] Langberg and Schulman define a sensitivity sampling framework and show how it can be used to approximate the value of sum-based clustering objectives such as the k-median or the k-means. The proofs are formulated with respect to point-sets from the Euclidean space and some arbitrary norm. Nevertheless, they also work for polygonal curves under the Fréchet distance.

However, to the best of our knowledge, there are no results on the VC dimension of the Fréchet space (Δ, d_F) yet[1]. Therefore, to bound the probability that a sample is an ε-coreset,

[1] Though there are results to appear at SoCG '19 by Driemel et al.

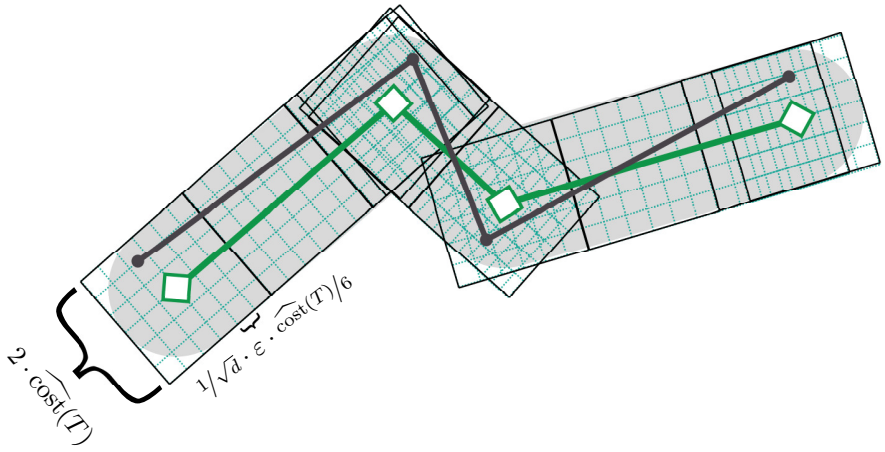


Figure 2 Exemplary grid-cover that is used for the algorithm of Theorem 4.2, for general polygonal curves and $d = 2$. A center-curve is depicted in green with cubes in black and associated grids in light blue. A curve with Fréchet distance less than $\widehat{\text{cost}}(T) := \alpha$ is also depicted. It can be observed that the vertices of this curve lie in at least one cell of a grid.

in particular that it can be used to approximate the value of the objective function for *all* possible center-sets, we restricted ourselves to the *discrete* k-median objective.

The sensitivity sampling framework works as follows: We are given a set T of n polygonal curves and run a 6-approximation algorithm to obtain a value α on the k-median objective function and the associated center-set C. The approximation algorithm we use is a local-search heuristic that uses a solution from the 6-approximation algorithm for the (k, l)-center objective by Buchin et al. as initial guess and thus has running-time $\mathcal{O}(n^2m^2 \log(m))$. We use α and C to assign every $\tau \in T$ a sensitivity value s_τ. These sensitivities and the total sensitivity $S := \sum_{\tau \in T} s_t$ suffice to build a probability distribution $\psi \colon T \to [0, 1]$, such that a sample of cardinality $\ell(\varepsilon, n) \in \Omega\left(\frac{\ln(n)}{\varepsilon^2}\right)$ from T with respect to ψ is a weighted ε-coreset for the discrete k-median objective with probability at least $\frac{2}{3}$, where every member of the sample is weighted by $\frac{n}{\ell(\varepsilon,n)}$. This is due to the fact that curves which have high impact on the value of the objective function for at least one center-set are sampled with higher probability, i.e., the probability to sample a curve is proportional to its "importance".

▶ **Theorem 5.1.** *There exists an algorithm that, given a set of n polygonal curves of complexity at most m each, and a parameter $\varepsilon \in (0, 1)$, computes an ε-coreset for the discrete k-median objective of cardinality $\mathcal{O}\left(\frac{\ln(n)}{\varepsilon^2}\right)$ in time $\mathcal{O}\left(n^2 \cdot m^2 \log(m) + \frac{\ln^2(n)}{\varepsilon^2}\right)$, with probability at least $\frac{2}{3}$.*

References

1 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. 10.1137/1.9781611975482.181.

2 Anne Driemel, Amer Krivošija, and Christian Sohler. Clustering Time Series Under the Fréchet Distance. In *Proceedings of the Twenty-seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 766–785. Society for Industrial and Applied Mathematics, 2016. ISBN 978-1-611974-33-1.

3 Piotr Indyk, Piotr Indyk, and Jiri Matousek. Low-Distortion Embeddings of Finite Metric Spaces. *Handbook of Discrete and Computational Geometry*, pages 177—-196, 2004.

4 Anil K. Jain. Data Clustering: 50 Years Beyond k-means. *Pattern Recognition Letters*, 31 (8):651–666, 2010. ISSN 0167-8655. 10.1016/j.patrec.2009.09.011.

5 Michael Langberg and Leonard J. Schulman. Universal Epsilon-Approximators for Integrals. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 598–607, 2010. 10.1137/1.9781611973075.50.

6 Dennis Rohde. Coresets for (k, l)-Clustering under the Fréchet Distance. Master's thesis, TU Dortmund University, December 2018.

7 Kenneth Zeger and Allen Gersho. Number of Nearest Neighbors in a Euclidean Code. *IEEE Transactions on Information Theory*, 40(5):1647–1649, 1994. ISSN 0018-9448. 10.1109/18.333884.

On the hardness of finding an average curve

Kevin Buchin[1], Anne Driemel[2], and Martijn Struijs[1]

1 **Department of Mathematics and Computing Science, TU Eindhoven, The Netherlands**
{k.a.buchin,m.a.c.struijs}@tue.nl
2 **University of Bonn, Hausdorff Center for Mathematics, Germany**
driemel@cs.uni-bonn.de

Abstract

We study the complexity of clustering curves under k-median and k-center objectives in the metric space of the Fréchet distance and related distance measures. The k-center problem has recently been shown to be NP-hard, even in the case where $k = 1$, i.e. the minimum enclosing ball under the Fréchet distance. We extend these results by showing that also the k-median problem is NP-hard for $k = 1$. Furthermore, we show that the 1-median problem is W[1]-hard with the number of curves as parameter. We show this under the discrete and continuous Fréchet and Dynamic Time Warping (DTW) distance. Moreover, closing some gaps in the literature, we show positive results for the (k, ℓ)-center variant under the discrete Fréchet distance. In particular, we give an $\widetilde{O}(mn)$-time $(1 + \varepsilon)$-approximation algorithm and a polynomial-time exact algorithm for fixed k, ℓ and ε.

1 Introduction

Clustering is an important tool in data analysis, used to split data into groups of similar objects. Their dissimilarity is often based on distance between points in Euclidean space. However, the dissimilarity of (polygonal) curves is more accurately measured by specialised measures: Dynamic Time Warping (DTW) [9], continuous and discrete Fréchet distance [1, 6].

We focus on *centroid-based clustering*, where each cluster has a centre curve and the quality of the clustering is based on the similarity between the centre and the elements inside the cluster. In particular, given a distance measure δ, we consider the following problems:

▶ Problem 1 (k-**median for curves with distance** δ). Given a set $\mathcal{G} = \{g_1, \ldots, g_m\}$ of polygonal curves, find a set $\mathcal{C} = \{c_1, \ldots c_k\}$ of polygonal curves that minimizes $\sum_{g \in \mathcal{G}} \min_{i=1}^{k} \delta(c_i, g)$.

▶ Problem 2 (k-**center for curves with distance** δ). Given a set $\mathcal{G} = \{g_1, \ldots, g_m\}$ of polygonal curves, find a set $\mathcal{C} = \{c_1, \ldots c_k\}$ of polygonal curves that minimizes $\max_{g \in \mathcal{G}} \min_{i=1}^{k} \delta(c_i, g)$.

We call the 1-median problem the *average curve* problem. Clustering on points for general k in the plane or higher dimension is often NP-hard [8] and clustering curves tends to be hard even when $k = 1$ and the curves lie in 1D. For instance, Buchin et. al. [2] show that the 1-center problem for the discrete and continuous Fréchet distance in 1D is NP-hard and that for the discrete Fréchet distance, it is NP-hard to approximate with a ratio better than 2. In this paper, we show that the average curve problem for discrete and continuous Fréchet distance in 1D is NP-complete and W[1]-hard when parametrised in the number of curves m.

Denote the set of all warping paths (or alignments, see also [9]) between curves x and y by $\mathcal{W}_{x,y}$. For any integers $p, q \geq 1$, define $\mathrm{DTW}_p^q(x, y) := \left(\min_{W \in \mathcal{W}_{x,y}} \sum_{(i,j) \in W} |x_i - y_j|^p\right)^{q/p}$. We call DTW_p^q the (p, q)-DTW distance.

The average curve problem for the $(2, 2)$-DTW distance has resisted efficient algorithms so far, which motivated several heuristic approaches [7, 9]. A formal proof of NP-hardness has only recently been given by Bulteau et. al. [3], who additionally show the $(2, 2)$-DTW

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

problem is W[1]-hard when parametrised in the number of input curves m and there exists no $f(m) \cdot n^{o(m)}$-time algorithm unless the ETH fails. In this paper, we prove the same hardness results of the average curve problem for the (p, q)-DTW distance for any $p, q \in \mathbb{N}$, with a different method. While Bulteau et. al. [3] note at the end of Section 5 their method might generalise to more variants of the DTW distance, when $p \neq q$, the (p, q)-DTW distance does not fit in their framework since then q/p is a non-trivial exponent. Furthermore, when $p = q = 1$, the variant has the form required in their framework, but the condition required for their hardness proof of an intermediate problems fails.

Since we still want efficient algorithms to do curve clustering, we look at a variant of these problems: we only look for centre curves with at most a fixed complexity, denoted by ℓ. So, the (k, ℓ)-center problem is to find a set of curves $\mathcal{C} = \{c_1, \ldots c_k\}$, each with at most ℓ vertices that minimizes $\max_{g \in \mathcal{G}} \min_{i=1}^{k} \delta(c_i, g)$ and the (k, ℓ)-median problem is defined analogously. Finding short centre curves is also useful for applications, as it can prevent overfitting the centre to details of individual input curves.

Although the general case for this variant is still NP-hard, we can find efficient algorithms when k and ℓ are fixed. The (k, ℓ)-center and (k, ℓ)-median problems were introduced by Driemel et. al. [5], who obtained an $\widetilde{O}(mn)$-time $(1 + \varepsilon)$-approximation algorithm for the (k, ℓ)-center and (k, ℓ)-median problem under the Fréchet distance for curves in 1D, assuming k, ℓ, ε are constant. In [2], we gave polynomial-time constant-factor approximation algorithms for the (k, ℓ)-center problem under the discrete and continuous Fréchet distance for curves in arbitrary dimension. In this paper, we give a $(1 + \varepsilon)$-approximation algorithm that runs in $\widetilde{O}(mn)$ time and a polynomial-time exact algorithm to solve the (k, ℓ)-center problem for the discrete Fréchet distance, when k, ℓ and ε are fixed.

2 Hardness of finding average curves

To show the hardness of the average curve problem for the Fréchet and DTW distance, we reduce from a variant of the NP-hard *Shortest Common Supersequence* (SCS) problem [10, 11], which we will call the *Fixed Character Common Supersequence* (FCCS) problem. If s is a string and x is a character, $\#_x(s)$ denotes the number of occurrences of x in s.

▶ Problem 3 (**Shortest Common Supersequence (SCS)**). Given a set S of m strings with length at most n over the alphabet Σ and an integer t, does there exists a string s^* of length t that is a supersequence of each string $s \in S$?

▶ Problem 4 (**Fixed Character Common Supersequence (FCCS)**). Given a set S of m strings with length at most n over the alphabet $\Sigma = \{A, B\}$ and $i, j \in \mathbb{N}$, does there exists a string s^* with $\#_A(s^*) = i$ and $\#_B(s^*) = j$ that is a supersequence of each string $s \in S$?

▶ **Lemma 1.** *The FCCS problem is NP-hard. The FCCS problem with m as parameter is W[1]-hard. There exists no $f(m) \cdot n^{o(m)}$ time algorithm for FCCS unless ETH fails.*

The proof idea is to reduce from SCS: given an instance (S, t) of SCS, construct $S' = \{s + AB^{2t}A + c(s) \mid s \in S\}$, where $c(s)$ denotes the string constructed by replacing all A's in s by B and vice versa. We reduce to the instance $(S', t + 2, 3t)$. If s^* is a common supersequence of length t for S, then $s^* + AB^{2t}A + c(s^*)$ is a supersequence of S' with the correct character count. Optimal supersequences of S' can be decomposed into this form. ◀

2.1 Complexity of the average curve under the Fréchet distance

We will show the hardness of finding the average curve under the discrete and continuous Fréchet distance d_{dF} and d_F via the following reduction from FCCS. Given an instance

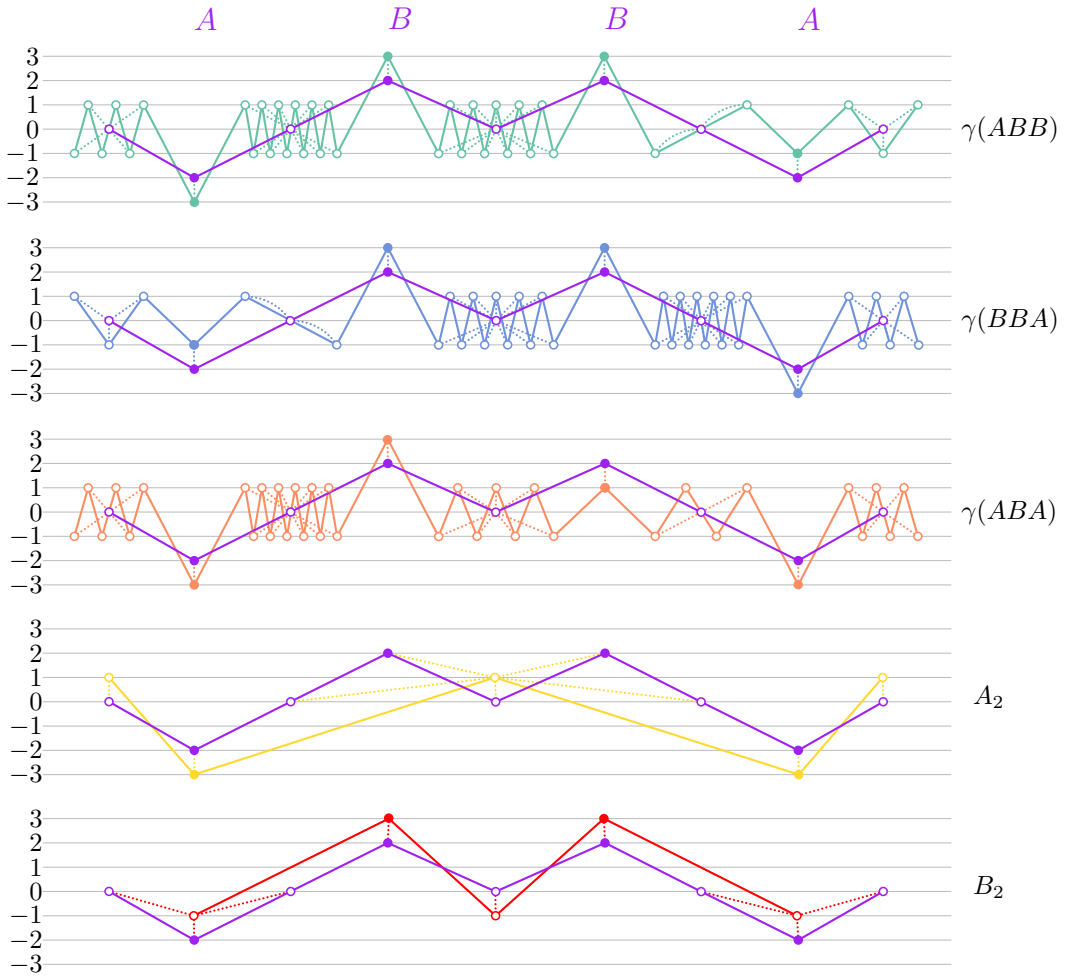


Figure 1 Five 1D-curves from $G \cup R_{i,j}$ in the reduction for the Fréchet average curve problem and a center curve constructed from $ABBA$ (purple) as in Lemma 2. Matchings are indicated by dotted lines. Note that each of these matchings achieves a (discrete) Fréchet distance of 1.

(S, i, j) of FCCS, we construct a set of curves using the following vertices in $\mathbb{R}$: $g_a = -1$, $g_b = 1$, $g_A = -3$, and $g_B = 3$. For a string $s \in S$, we map each character to a subcurve in $\mathbb{R}$:

$$A \to (g_a g_b)^{i+j} g_A (g_a g_b)^{i+j} \qquad B \to (g_b g_a)^{i+j} g_B (g_b g_a)^{i+j}.$$

The curve $\gamma(s)$ is constructed by concatenating the subcurves resulting from this mapping, $G = \{\gamma(s) \mid s \in S\}$ denotes the set of these curves. Additionally, we use the curves

$$A_i = g_b (g_A g_b)^i \qquad B_j = g_a (g_B g_a)^j.$$

We will call subcurves containing only g_A or g_B vertices *letter gadgets* and subcurves containing only g_a or g_b vertices *buffer gadgets*. Let $R_{i,j}$ contain curves A_i and B_j, both with multiplicity $\alpha = |S|(|S| - 1) + 1$. We reduce to the instance $(G \cup R_{i,j}, r)$ of the average curve problem, where $r = |S| + 2\alpha$. We use the same construction for the discrete and continuous case. For an example of this construction, take $S = \{ABB, BBA, ABA\}$, $i = 2$, $j = 2$. Then $ABBA$ is a supersequence of S with the correct number of characters, see Figure 1.

▶ **Lemma 2.** *If (S, i, j) is a true instance of FCCS, then $(G \cup R_{i,j}, r)$ is a true instance of the average curve problem for discrete and continuous Fréchet.*

Proof. Since $\mathrm{d}_F(x, y) \leq \mathrm{d}_{dF}(x, y)$ for all curves x, y, considering the discrete version suffices.

Since (S, i, j) is a true instance of FCCS, there exists a common supersequence s^* of S with $\#_A(s^*) = i$ and $\#_B(s^*) = j$. Construct the curve c of complexity $2|s^*| + 1$, given by

$$c_l = \begin{cases} 0 & \text{if } l \text{ is odd} \\ -2 & \text{if } l \text{ is even and } s^*_{l/2} = A \\ 2 & \text{if } l \text{ is even and } s^*_{l/2} = B \end{cases},$$

for each $l \in \{1, \ldots, 2|s^*| + 1\}$. Note s^* is a supersequence of the sequence of letter gadgets in any curve $g \in G \cup R_{i,j}$ and therefore we can match all letter gadgets from g within distance 1 such that we get $\mathrm{d}_{dF}(c, g) \leq 1$. This means $\sum_{g \in G \cup R_{i,j}} \mathrm{d}_{dF}(c, g) \leq |S| + 2\alpha = r$. ◀

For the converse, we can show that if there is a curve c^* with $\sum_{g \in G \cup R_{i,j}} \mathrm{d}_F(c^*, g) \leq r$, then $\mathrm{d}_F(c^*, g) < 2$ for all $g \in G \cup R_{i,j}$. This means we can apply the hardness proof for the 1-center problem under the Fréchet distance from [2] to partition c^* into A-parts, B-parts and buffer parts and construct a supersequence for S from the sequence of A/B-parts in c^*.

▶ **Lemma 3.** *If $(G \cup R_{i,j}, r)$ is a true instance of the average curve problem for discrete and continuous Fréchet, then (S, i, j) is a true instance of FCCS.*

Since the reduction runs in polynomial time and the number of input curves is bounded by a quadratic function in $|S|$, we get the following result.

▶ **Theorem 4.** *The average curve problem for discrete and continuous Fréchet distance is NP-hard. When parametrised in the number of input curves m, this problem is W[1]-hard.*

2.2 Complexity of the average curve under the DTW distance

We will show that the average curve problem for (p, q)-DTW is NP-hard for all $p, q \in \mathbb{N}$. We use the same reduction from Section 2.1, but now map the characters of $s \in S$ to

$$A \to g_0^\beta g_a^\beta g_0^\beta \qquad B \to g_0^\beta g_b^\beta g_0^\beta,$$

use the curves $A_i = g_0^\beta (g_a^\beta g_0^\beta)^i$ and $B_j = g_0^\beta (g_b^\beta g_0^\beta)^j$, and set $r = \sum_{s \in S} (i + j - |s|)^{q/p} + \alpha(i^{q/p} + j^{q/p})$, $\beta = \lceil r/\varepsilon^q \rceil + 1$, $\alpha = |S|$, where $\varepsilon > 0$ is chosen sufficiently small and depends only on i, j, p, q. See Figure 2 for an example with $S = \{ABB, BBA, ABA\}$ and $i = j = 2$.

▶ **Lemma 5.** *If (S, i, j) is a true instance of FCCS, then $(G \cup R_{i,j}, r)$ is a true instance of (p, q)-DTW average curve.*

Proof. This is analogous to Lemma 2. ◀

For the converse, we identify vertices in a satisfying curve c^* that are close to g_a or g_b, such that g_a^β and g_b^β subcurves must be matched to them and construct a supersequence s' out of them. The curves A_i and B_j are used to show that $\#_A(s') = i$ and $\#_B(s') = j$.

▶ **Lemma 6.** *If $(G \cup R_{i,j}, r)$ is a true instance of (p, q)-DTW average curve, then (S, i, j) is a true instance of FCCS.*

Since the reduction runs in polynomial time and the number of input curves is bounded by a linear function in $|S|$, we get the following result:

▶ **Theorem 7.** *The average curve problem for the (p, q)-DTW distance is NP-hard, for any $p, q \in \mathbb{N}$. When parametrised in the number of input curves m, this problem is W[1]-hard. There exists no $f(n) \cdot n^{o(m)}$ time algorithm for this problem unless ETH fails.*

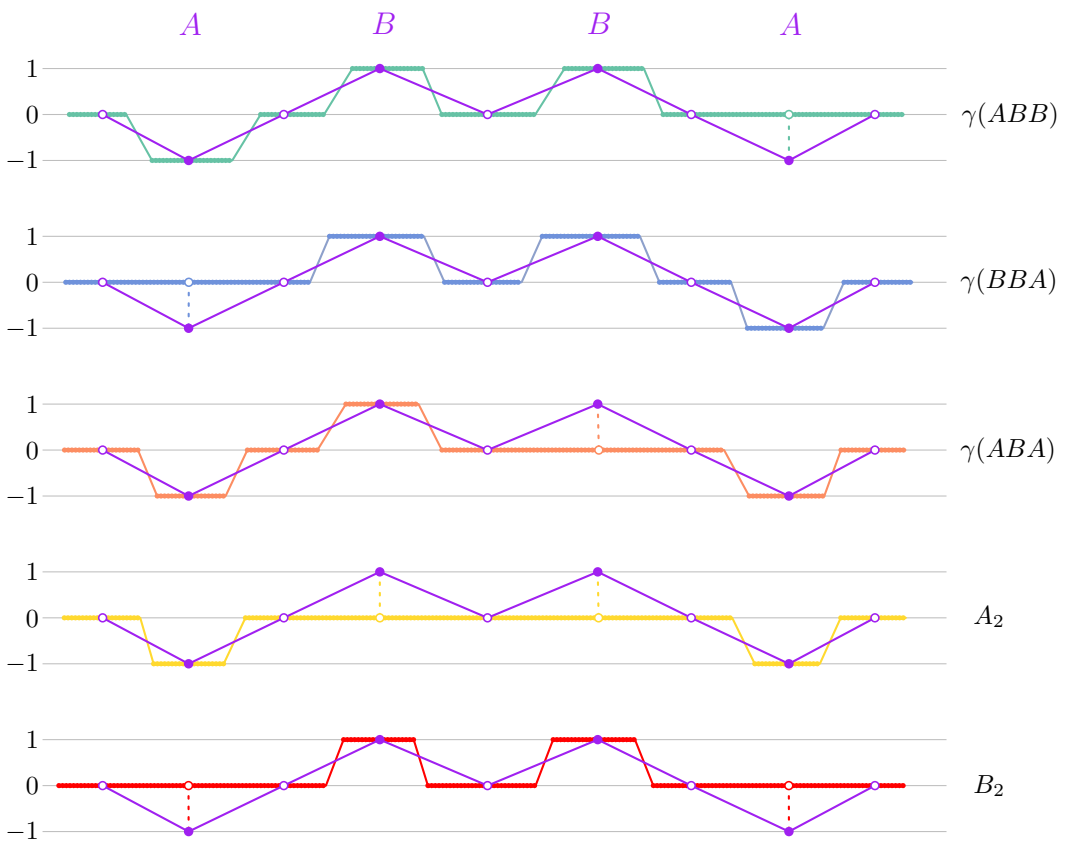


Figure 2 Five 1D-curves from $G \cup R_{i,j}$ and a center curve constructed from $ABBA$ (purple) as in Lemma 5. Fat horizontal lines indicate β consecutive vertices. Vertices that match at distance 0 touch, those matching at distance 1 are indicated by dotted lines. The center has 1 mismatch with the first 3 curves and 2 with the final two, so the total cost here is $3 \cdot (1^p)^{q/p} + 2\alpha \cdot (2 \cdot 1^p)^{q/p} = 3 + 2\alpha \cdot 2^q$.

3 $(1+\varepsilon)$-approximation for (k,ℓ)-center clustering for the discrete Frechet distance in $\mathbb{R}^d$

In this section, we develop a $(1+\epsilon)$-approximation algorithm for the (k,ℓ)-center problem under the discrete Fréchet distance that runs in $O(mn\log(n))$ time for fixed k,ℓ,ϵ.

Given a set $\mathcal{G}$ of m input curves in $\mathbb{R}^d$ of complexity at most n each, use the algorithm by Buchin et. al. [2] to compute a set $\mathcal{C}$ of k curves that forms a 3-approximation for the (k,ℓ)-center problem in $O(km\cdot \ell n\log(\ell+n))$ time. Call the cost of these centers Δ. Let $\mathcal{C}^*$ be an optimal solution that achieves cost O. For each vertex p^* in $\mathcal{C}^*$, there is a vertex q on an input curve with $\|p^*-q\|\leq O$ and there is a vertex p in $\mathcal{C}$ with $\|p-q\|\leq\Delta$. So, by the triangle inequality, all vertices of $\mathcal{C}^*$ lie within a ball of radius 2Δ centred at a vertex of $\mathcal{C}$.

We can cover these balls with a regular grid of $O(\varepsilon^{-d})$ vertices with distance of $\varepsilon\cdot 2\Delta/(3\sqrt{d})$, so that there exists a vertex $g(p^*)$ on such a grid with $\|p^*-g(p^*)\|\leq \varepsilon\Delta/3=\varepsilon O$. So, for every curve $c^*\in\mathcal{C}^*$, there exists a single curve $g(c^*)$ of gridpoints with $\mathrm{d}_{dF}(g(c^*),c^*)\leq\varepsilon O$, which means that for all $g\in\mathcal{G}$, there exists a curve c^* such that $\mathrm{d}_{dF}(g,g(c^*))\leq(1+\varepsilon)O$. This means the set $\{g(c^*)\mid c^*\in\mathcal{C}^*\}$ gives a $(1+\varepsilon)$-approximation, which we can find by iterating over all curves using the gridpoints. We conclude with the following theorem:

▶ **Theorem 8.** *Given m curves in $\mathbb{R}^d$, each of complexity at most n, and $k,\ell\in\mathbb{N}$ and some $0<\varepsilon\leq 1$, we can compute an $(1+\varepsilon)$-approximation to the (k,ℓ)-center problem for the discrete Fréchet distance in $O\left(((Ck\ell)^{k\ell}+\log(\ell+n))\cdot k\ell\cdot mn\right)$ time, with $C=\left(\frac{6\sqrt{d}}{\varepsilon}\right)^d$.*

4 Exact algorithm for (k,ℓ)-center for discrete Fréchet in 2D

We give an algorithm that solves the (k,ℓ)-center problem for the discrete Fréchet distance in 2D in polynomial time for fixed k and ℓ. We first show how to solve the decision version.

The main idea of the algorithm for the decision version is based on the following observation: for a given r, we have $\min_{c\in\mathcal{C}}\mathrm{d}_{dF}(c,g)\leq r$ for all $g\in\mathcal{G}$ if and only if each vertex p of a curve in $\mathcal{C}$ lies in the intersection of the disks of radius r around all vertices q from curves in $\mathcal{G}$ that p is matched with. Furthermore, it does not matter where the vertex p lies within the intersection region. This means we can select one vertex for each region and exhaustively test all sets with k curves of ℓ vertices that can be constructed by using only the selected vertices to determine if there exists a set of curves $\mathcal{C}$ such that $\min_{c\in\mathcal{C}}\mathrm{d}_{dF}(c,g)\leq r$ for all $g\in\mathcal{G}$.

The corresponding arrangement of circles has complexity $O((nm)^2)$, and can be computed in that time [4], see Figure 3 for an example. We solve the optimisation version by performing a binary search over the at most $O((mn)^3)$ values of r at which the arrangement changes combinatorially, which occurs only when some disks intersect at a single point.

▶ **Theorem 9.** *Given a set of m curves G in the plane with at most n vertices each, we can find a solution to the (k,ℓ)-center problem in $O((mn)^{2k\ell+1}k\ell\log(mn))$ time.*

References

1 Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5:75–91, 1995. doi:10.1142/S0218195995000064.

2 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 30th ACM-SIAM Symposium on Discrete Algorithms*, pages 2922–2938, 2019. doi:10.1137/1.9781611975482.181.

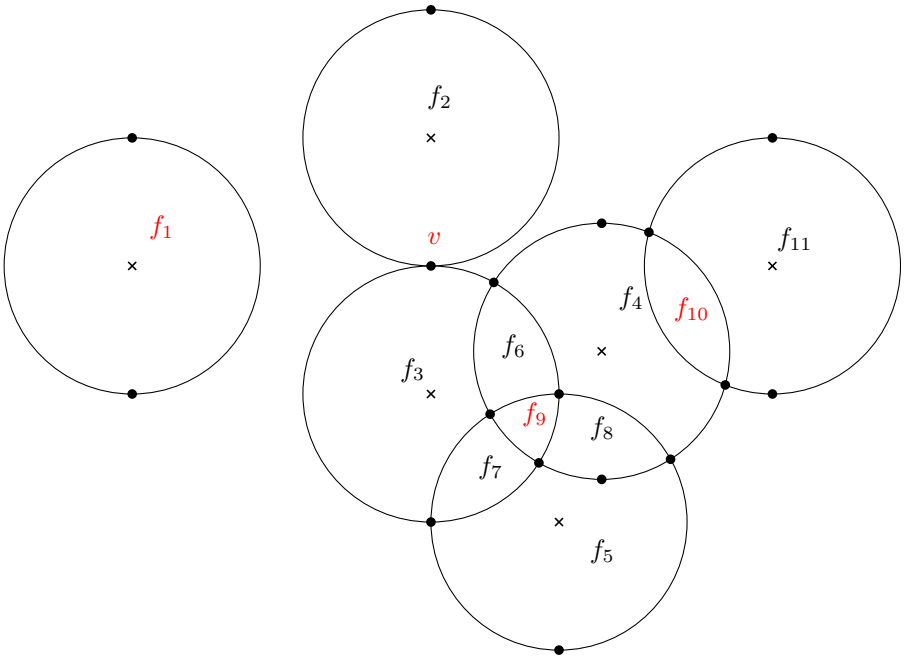


Figure 3 A possible arrangement of circles. Crosses indicate the vertices from the curves in $\mathcal{G}$, all bounded faces are numbered. The relevant intersection regions for the Fréchet distance are in red.

3 Laurent Bulteau, Vincent Froese, and Rolf Niedermeier. Tight hardness results for consensus problems on circular strings and time series. *arXiv preprint arXiv:1804.02854*, 2018. URL: `http://arxiv.org/abs/1804.02854`.

4 Bernard Marie Chazelle and Der-Tsai Lee. On a circle placement problem. *Computing*, 36(1-2):1–16, 1986.

5 Anne Driemel, Amer Krivošija, and Christian Sohler. Clustering time series under the Fréchet distance. In *Proceedings of the 27th ACM-SIAM Symposium on Discrete Algorithms*, pages 766–785. Society for Industrial and Applied Mathematics, 2016.

6 Thomas Eiter and Heikki Mannila. Computing discrete Fréchet distance. Technical Report CD-TR 94/64, Christian Doppler Laboratory for Expert Systems, TU Vienna, Austria, 1994.

7 Lalit Gupta, Dennis L Molfese, Ravi Tammana, and Panagiotis G Simos. Nonlinear alignment and averaging for estimating the evoked potential. *IEEE Transactions on Biomedical Engineering*, 43(4):348–356, 1996.

8 Nimrod Megiddo and Kenneth J. Supowit. On the complexity of some common geometric location problems. *SIAM Journal of Computing*, 13(1):182–196, 1984. `doi:10.1137/0213014`.

9 François Petitjean and Pierre Gançarski. Summarizing a set of time series by averaging: From Steiner sequence to compact multiple alignment. *Theoretical Computer Science*, 414(1):76 – 91, 2012. `doi:10.1016/j.tcs.2011.09.029`.

10 Krzysztof 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.

11 Kari-Jouko Räihä and Esko Ukkonen. The shortest common supersequence problem over binary alphabet is NP-complete. *Theoretical Computer Science*, 16(2):187 – 198, 1981. `doi:10.1016/0304-3975(81)90075-X`.

Maximum Physically Consistent Trajectories

Bram Custers[1], Mees van de Kerkhof[2], Wouter Meulemans[1], Bettina Speckmann[1], and Frank Staals[2]

1 **TU Eindhoven, the Netherlands**
[b.a.custers,w.meulemans,b.speckmann]@tue.nl
2 **Utrecht University, the Netherlands**
[m.a.vandekerkhof,f.staals]@uu.nl

Abstract

We study the problem of detecting outlying measurements in a GPS trajectory. Our method considers the physical possibility for the tracked object to visit combinations of measurements, using simplified physics models. We aim to compute the maximum subsequence of the measurements that is consistent with a given physics model. We give an $O(n \log^3 n)$ time algorithm for 2D-trajectories in a model with unbounded acceleration but bounded velocity, and an output-sensitive algorithm for any model where consistency checks can be done in $O(1)$ time and consistency is transitive.

1 Introduction

The use of GPS trajectories in computing has greatly increased in recent years, with many algorithms and applications using them as input for analysis. However, since trajectories are created by performing real-world measurements there is always some amount of spatial error. Thus, no matter the application, it is important to preprocess trajectories to try and reduce the impact measurement errors have on the result. There are several types of preprocessing that can be applied to a trajectory to limit the impact of errors. We focus on *outlier detection*, where we find and remove data points with a large distance to the rest of the data. These outliers are likely to have been the result of large measurement errors.

We approach the problem from a physics perspective. Trajectories track real-world objects with real-world physical properties. By considering if the tracked object could physically travel between its measured locations in the given time interval we can find outlying measurements. We call a subsequence of the measurements of a trajectory *consistent* if there is a path the object could have taken that visits the points in order and does not violate the constraints of (is consistent with) the physics model. Our problem is then to find the maximum consistent subsequence[1] of the trajectory.

Contributions. We give an output-sensitive $O(nd)$ time algorithm for finding the maximum consistent subsequence of a trajectory, where d is the number of outliers, assuming a physics model that allows consistency checks for a pair of measurements in $O(1)$ time and has transitive consistency. For the physics model where the object has bounded speed but no other restrictions we additionally give an $O(n \log^3 n)$ time algorithm.

Related work. Outlier detection is part of any application that must cope with imprecise data. Hence, many different methods have been developed for many different contexts. A general survey of outlier detection methodologies is given in [6]. Gupta et al. [5] give a survey for outlier detection in data with a temporal component, including trajectories.

[1] This subsequence does not need to be a consecutive subsequence.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

Outlier detection for trajectories has mainly been studied as finding outlying trajectories in a data set of trajectories, rather than finding outlying measurements in a single trajectory [4, 8, 9, 13]. Although detection of outlying measurements in a single trajectory has not been studied as such, the problem shows similarities to trajectory simplification and trajectory smoothing, both well-studied topics. We refer to [14] for a survey.

2 Definitions & Notation

f A trajectory $T = \langle p_1, p_2, \ldots, p_n \rangle$ is an ordered sequence of n measurements, each of which contains at least a location $\mathbf{x}_i$ and timestamp t_i. Additional data such as acceleration and velocity may also be present. The measurements are ordered by their timestamp, so for any pair (p_i, p_j) with $i < j$ we have that $t_i < t_j$.

We use physics models to describe possible states that can be reached in the future, given some measurement at a given time. Under a particular model, a subsequence $\langle p_i, \ldots, p_j \rangle$ is *consistent* iff a path exists that connects all measurements and obeys the constraints of the physics model. We will denote this by $C(p_i, \ldots, p_j)$. We will restrict ourselves to models where existence of such a path per measurement pair can be checked in $O(1)$. In addition, we require that the consistency is transitive. That is, for all $i < j < k$ we have that

$$C(p_i, p_j) \wedge C(p_j, p_k) \iff C(p_i, p_j, p_k). \tag{1}$$

For example, a model that bounds only the maximum speed meets the above criteria. We use the generic model in Section 3 and the speed-bounded model in Section 4. An example of a model without transitivity is one where both speed and acceleration are bounded. Here, the situation can occur that reaching p_j from p_i requires a velocity at t_j that makes reaching p_k within the physics model impossible, while the separate pairs are consistent.

3 Output-sensitive algorithm for models with transitive consistency

We assume the generic physics model with an $O(1)$-time consistency check and transitive consistency. First, we observe that we can follow a similar methodology to the Imai-Iri simplification algorithm [7]. Let $G = (V, A)$ be a directed acyclic graph with a vertex for each measurement of T and an arc from a vertex v_i to v_j if $C(p_j, p_i)$. This graph has $O(n^2)$ complexity and we can test for the existence of each arc in constant time: construction takes $O(n^2)$ as well. Since consistency is transitive, a path in G corresponds to a consistent subsequence. We can determine the longest path, and thus the maximum consistent subsequence, in this graph in $O(|V| + |A|) = O(n^2)$ time.

We now show that we can further use transitivity to obtain an output-sensitive variant of this algorithm. Rather than constructing the full DAG, we build a subgraph where each vertex has at most one incoming arc coming from the vertex with the longest subsequence that is consistent with it. This is captured in the theorem below.

▶ **Theorem 1.** *Consider a physics model that allows checking the consistency of a pair of measurements in constant time, and where consistency between measurements is a transitive relation. The maximum consistent subsequence of a trajectory $T = \langle p_1, p_2, \ldots, p_n \rangle$ can be computed in $O(nd)$ time, where d is the number of outliers.*

Proof. We handle the measurements in chronological order. We maintain a linked list that stores all previously handled measurements, sorted in decreasing order of the largest length of a path that ends in that measurement. For a new measurement, we go down this list in

order, stopping as soon as we find a measurement that is consistent with the new one. We then know that the length of the longest path ending in the new measurement is equal to the length of the path ending in this previous measurement plus 1. During the list traversal, we maintain the first occurrence of the latest unique path length, so that we can insert the new element in O(1) time. After we have handled all measurements in this way we know the longest path. Each of the $(n-d)$ measurements that ends up in the longest path requires only 1 successful check preceded by at most d failed checks, and the d outliers require at most n checks, giving $O(nd)$ total checks which is also the time bound. ◀

4 Subquadratic algorithm for the speed-bounded model

Here we consider the speed-bounded model for 2D trajectories. We denote the maximum speed by s_{max}. A subsequence S is consistent under this model if a path $r(t) = (x(t), y(t))$ exists with $||\frac{d}{dt}r(t)|| < s_{max}$ and $(x(t_i), y(t_i)) = (x_i, y_i)$ for all $p_i \in S$.

We develop an insertion-only data structure that, given a measurement q, can determine the length k of the maximum subsequence ending at q in $O(\log^3 n)$ time. Insertions are supported in $O(\log^3 n)$ time. By incrementally building the data structure in chronological order, we can determine the maximum consistent trajectory in $O(n \log^3 n)$ time.

4.1 Consistency data structure

We may view the measurements as points in 3D space, with the third axis being time, i.e. $p_i = (x_i, y_i, t_i)$. For all $t > t_i$, measurements that are consistent with p_i must lie inside a cone, starting at p_i with radius $s_{max}(t-t_i)$ at time $t \geq t_i$. This representation of consistency bears some similarity to the space-time prisms [10].

To determine if a measurement p_i is consistent with any measurement of a set P of previous measurements, we use an *additively weighted Voronoi diagram* (AWVD) for the elements. Given a set of points $\{r_1, \ldots, r_n\}$ and weights $\{w_1, \ldots, w_n\}$, this diagram partitions the plane into cells $\{c_1, \ldots, c_n\}$ associated with the points, such that for any point $r' \in c_i : d(r', r_i) - w_i \leq d(r', r_j) - w_j$, for all $r_j \neq r_i$. Here, d is a distance measure (in our case the Euclidean distance), and the equality holds only on boundaries between cells.

We use the locations of the measurements in the set P for $\{r_i\}$ and for the weights, we pick $s_{max}(t'-t_i)$ for every measurement for some $t' > t_n$. This results in the construction depicted in Figure 1. On this AWVD, we construct a point location structure $\mathcal{D}$. A consistency query with p_i on $\mathcal{D}$ consists of two steps: first, we retrieve the element r_c such that cell c_c contains the location of p_i; then, we return p_c if it is consistent with p_i, or otherwise the query "fails".

▶ **Lemma 2.** *Let $\mathcal{D}$ be a consistency data structure on a set P of measurements. If a consistency query with p_i fails on $\mathcal{D}$, no measurement of P is consistent with p_i.*

Proof sketch. The definition of AWVD tells us that the distance between the found location in $\mathcal{D}$ and the query point, minus the points weight is at most the same value for any other location. Using the strict inequality we get from inconsistency, we can derive that all other locations in $\mathcal{D}$ are also inconsistent. ◀

We can construct the AWVD in $O(m \log m)$ time on m elements, where the complexity of the AWVD is $O(m)$ [3]. We may then construct a point location structure in $O(m \log m)$ time on the AWVD that allows for querying in $O(\log m)$ time in which cell a point is situated [2]. As a consistency check costs $O(1)$ time, we obtain the lemma below.

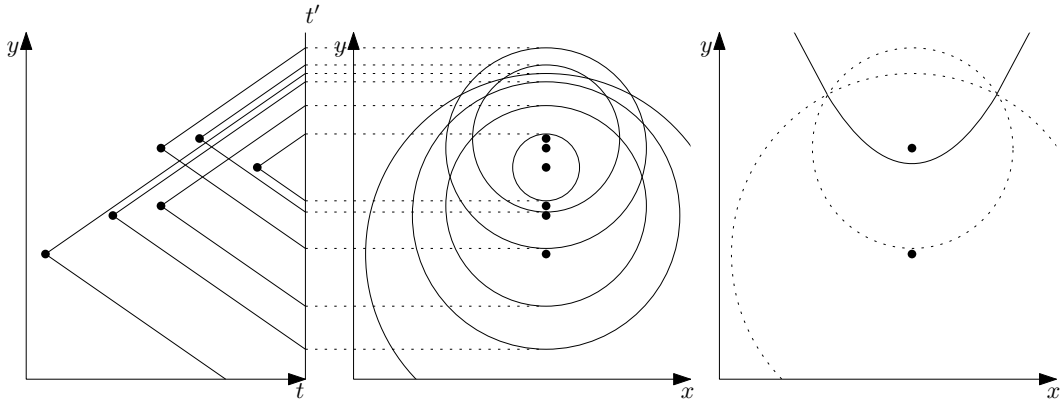


Figure 1 Example of the AWVD on measurements with equal x-coordinates. (Left) The y-coordinate of the measurements in time and the maximum speed cones progressing in time. (Middle) A cross-section of the (x, y, t) cones at time t' with accompanying centers (x, y). (Right) The corresponding additively weighted Voronoi diagram.

▶ **Lemma 3.** *Let P be a subset of T with $|P| = m$. We can construct a data structure $\mathcal{D}$ for querying consistency of a later measurement with any of the measurements in P in $O(m \log m)$ time. The data structure supports a consistency query in $O(\log m)$ time.*

4.2 Insertion-only consistency data structure

Testing whether a measurement is consistent with any previous measurement of a subsequence of T is a decomposable search problem. Thus, we use the approach by Bently and Saxe [1] to turn our consistency data structure into an efficient insertion-only data structure.

For a set of m measurements, we maintain $O(\log m)$ instances of our static data structure $\mathcal{D}_1, .., \mathcal{D}_{O(\log m)}$. Every measurement is in one of these $O(\log m)$ data structures. Data structure $\mathcal{D}_i$ has size 2^i. On insertion, we create a new $\mathcal{D}_1$ with the inserted measurement. When we get two data structures of same size 2^i, we remove both and replace them by a single data structure of size 2^{i+1}. We repeat this process until no duplicates exist. To answer a query we simply query all $O(\log m)$ data structures.

The above construction together with the consistency query structure gives $O(\log^2 m)$ time for a query and $O(\log^2 m)$ amortized time for an insertion. By the results of Overmars and Van Leeuwen [12], the insertion time can be made a worst-case running time.

▶ **Lemma 4.** *We can modify the consistency data structure $\mathcal{D}$ to be insertion-only, providing a $O(\log^2 m)$ query time and $O(\log^2 m)$ insertion time in the worst-case.*

4.3 A BB[α] tree for maximum subsequences

We combine processed measurements and the consistency data structure in a BB[α] tree [11]. We store the measurements in the leaves along with the length of the maximum consistent subsequence ending at the measurement, k. In the intermediate nodes we keep an insertion-only consistency structure as described before, built on all measurements in the subtree. The leaves are kept sorted on k in ascending order.

We use the BB[α] tree to keep the tree of depth $O(\log n)$, while avoiding tree rotations at insertions which require full recomputation of the intermediate node data structures.

Instead, we rebuild a subtree when it becomes sufficiently unbalanced, which gives us a better amortized running time.

We can query the tree with a measurement p to determine a predecessor q with largest k such that $C(q, p)$: at each level, we query the right child data structure or leaf measurement if a measurement is consistent with p. If so, we traverse the right subtree, otherwise the left subtree. We continue until we reach a leaf, giving us the largest k with associated measurement that is consistent with the query measurement. Since we query $O(\log n)$ intermediate nodes, this process takes $O(\log^3 n)$ time.

After a query, we can insert the measurement p with value $k+1$ in the tree. We insert the measurement in the consistency data structures of all ancestor nodes and possibly rebalance the tree by rebuilding an unbalanced subtree. Using the BB[α] tree and the insertion-only data-structures, this takes $O(\log^3 n)$ amortized time.

▶ **Lemma 5.** *Consider a BB[α] tree storing an insertion-only consistency structure at each internal node. We can insert a new node into the tree, rebalance it and rebuild the necessary consistency structures in $O(\log^3 n)$ amortized time per insertion.*

4.4 Running-time analysis

To determine the maximum consistent subsequence, we need to process all measurements, querying for their predecessor in the tree and then inserting it into the tree. Finally, we query for the largest k value. By maintaining pointers from measurements to their predecessors during processing, we obtain the maximum consistent subsequence. Since we do a query and an insert per measurement, the algorithm runs in $O(n \log^3 n)$ as stated before.

▶ **Theorem 6.** *Given a trajectory T with n measurements, we can determine the maximum consistent subsequence of T for the 2D speed-bounded model in $O(n \log^3 n)$ time.*

5 Acknowledgement

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.

References

1 J. L. Bentley and J. B. Saxe. Decomposable searching problems i. static-to-dynamic transformation. *Journal of Algorithms*, 1(4):301–358, 1980.
2 H. Edelsbrunner, L. J. Guibas, and J. Stolfi. Optimal point location in a monotone subdivision. *SIAM Journal on Computing*, 15(2):317–340, 1986.
3 S. Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica*, 2(1-4):153, 1987.
4 Y. Ge, H. Xiong, Z.-h. Zhou, H. Ozdemir, J. Yu, and K. C. Lee. Top-eye: Top-k evolving trajectory outlier detection. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1733–1736, 2010.
5 M. Gupta, J. Gao, C. C. Aggarwal, and J. Han. Outlier detection for temporal data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 26(9):2250–2267, 2014.
6 V. Hodge and J. Austin. A survey of outlier detection methodologies. *Artificial intelligence review*, 22(2):85–126, 2004.
7 H. Imai and M. Iri. Polygonal approximations of a curve—formulations and algorithms. *Computational Morphology*, pages 71–86, 1988.

8 J.-G. Lee, J. Han, and X. Li. Trajectory outlier detection: A partition-and-detect framework. In *Proceedings of the IEEE 24th International Conference on Data Engineering*, pages 140–149, 2008.

9 X. Li, J. Han, S. Kim, and H. Gonzalez. Roam: Rule-and motif-based anomaly detection in massive moving object data sets. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 273–284, 2007.

10 H. J. Miller. Time geography and space-time prism. *International Encyclopedia of Geography: People, the Earth, Environment and Technology*, pages 1–19, 2017.

11 J. Nievergelt and E. M. Reingold. Binary search trees of bounded balance. *SIAM Journal on Computing*, 2(1):33–43, 1973.

12 M. H. Overmars and J. van Leeuwen. Worst-case optimal insertion and deletion methods for decomposable searching problems. *Information Processing Letters*, 12(4):168–173, 1981.

13 G. Yuan, S. Xia, L. Zhang, Y. Zhou, and C. Ji. Trajectory outlier detection algorithm based on structural features. *Journal of Computational Information Systems*, 7(11):4137–4144, 2011.

14 Y. Zheng. Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(3):29, 2015.

Distance Measures for Embedded Graphs - Revisited *

Hugo A. Akitaya[†1], Maike Buchin[2], and Bernhard Kilgus[‡3]

1 **Department of Computer Science, Tufts University, Medford, MA, USA**
hugo.alves_akitaya@tufts.edu
2 **Department of Computer Science, Technical University Dortmund, Dortmund, Germany**
Maike.Buchin@tu-dortmund.de
3 **Department of Mathematics, Ruhr-University Bochum, Bochum, Germany**
Bernhard.Kilgus@rub.de

Abstract

In this extended abstract, we present new hardness and algorithmic results for the graph distances presented at EuroCG 2017 [10]. We consider the case of the graph distance based on the Fréchet distance for plane graphs. We prove that deciding this distance is NP-hard and show how our general algorithmic approach yields an exact exponential time algorithm and a polynomial time approximation algorithm for this case.

* Full version available on arXiv [6].

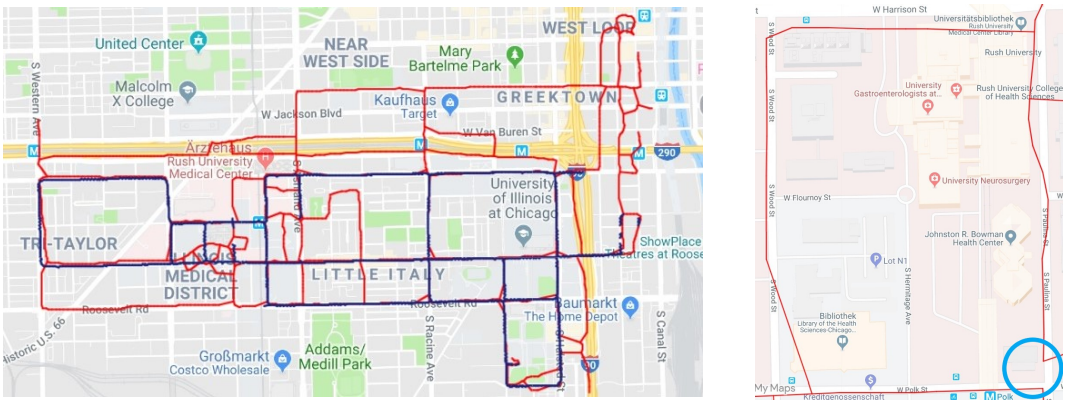
† Supported by National Science Foundation grants CCF-1422311 and CCF-1423615, and the Science Without Borders scholarship program.

‡ Supported by the Deutsche Forschungsgemeinschaft (DFG), project BU 2419/3-1

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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 Introduction

Motivation We study the task of 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], see Figure 1.



(a) Two partial map reconstructions of Chicago.

(b) Different topology.

Figure 1 Figure **(a)** shows the results of two map construction algorithms (blue: reconstuction by Davies et al. [12]; red: reconstruction by Ahmed et al. [5]). An appropriate measure for assessing the quality of the reconstruction should compare both the geometry and the topology of the reconstructions and the ground truth.

Related Work A few different approaches have been proposed for comparing such graphs. These are subgraph-isomorphism, edit distance [11], algorithms that compare all paths [1] or random samples of shortest paths [13], and the local persistent homology distance [2]. However, most of these capture only the geometry or only the topology of the embedded graphs. The sampling-based distance presented in [9] captures both, but it is not a formally defined distance. The traversal distance [7] is similar to the measures proposed here but captures the combinatorial structure of the graphs to a lesser extent.

Definitions and Previous Results 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 $s : G_1 \to G_2$ 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 $s(v)$. Our graph distances are generalizations of the (weak) Fréchet distance, popular distance measures for curves [8], to graphs: We define the *directed (weak) graph distance* $\vec{\delta}_{(w)G}$ as

$$\vec{\delta}_{(w)G}(G_1, G_2) = \inf_{s:G_1 \to G_2} \max_{e \in E_1} \delta_{(w)F}(e, s(e)),$$

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.

The general algorithm to compute the directed (weak) graph distances is based on the definition of valid ε-placements of the vertices and edges. 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.

Deciding the directed (weak) graph distance is NP-hard for general graphs, but we can compute the (weakly) valid ε-placements in polynomial time [6, 10]. 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). Therefore, the distances are decidable in polynomial time in these cases.

New Results In this paper, we show that deciding whether $\vec{\delta}_G(G_1, G_2) \leq \varepsilon$ remains NP-hard if G_1 and G_2 are plane graphs, that is the existence of a valid ε-placement for each vertex is not a sufficient criterion for $\vec{\delta}_G(G_1, G_2) \leq \varepsilon$ here. Furthermore, we prove an inapproximability result for this case. Subsequently, we present an exact exponential time algorithm and a polynomial time approximation algorithm based on the general algorithmic approach.

2 Hardness Results

▶ **Theorem 1.** *For plane graphs G_1, G_2, deciding whether $\vec{\delta}_G(G_1, G_2) \leq \varepsilon$ is NP-hard.*

Proof. Here, we give a concise version of the proof. For a more elaborated version, see [6]. We prove the NP-hardness by a reduction from Monotone-Planar-3-Sat (MP3S). That is, we construct straight-line embedded graphs G_1, G_2 based on a MP3S instance A, with edges of G_2 labeled True or False. We describe the construction of the subgraphs (gadgets) for the Variables and Clauses of A and prove which binary combinations can be realized such that all edges and their images are within Fréchet distance at most ε. Figure 2 and 3 illustrate the gadgets and a partial graph construction. We denote the ε-tube around the edge e by $T_\varepsilon(e) = e \bigoplus B_\varepsilon$. A path labeled True (False) is shortly denoted as True (False) signal. All vertices of the graph can be either placed arbitrarily within a given ε-surrounding or must lie at the intersection of two lines. This ensures that the construction uses rational coordinates only and can be computed in polynomial time.

For the Variable gadget, we draw two edges, e_1, e_2, of G_1 in a $90° - 120°$ angle incident to a vertex v and add vertices w_1 (w_2) of G_2 at the intersection of the outer boundary of $T_\varepsilon(e_2)$ ($T_\varepsilon(e_1)$) and a line through e_1 (e_2). Furthermore, we add a vertex w_3 of G_2 at the intersection of the boundaries of $T_\varepsilon(e_1)$ and $T_\varepsilon(e_2)$. We connect w_1 and w_2 with w_3 and draw an edge from w_1 and w_2 inside the ε-tubes around e_1 and e_2, labeled True. Analogously, we place two edges from w_3 labeled False. For the Variable gadget a True-True combination is not possible: The vertex v has two placements p_1 and p_2. Assume we choose p_1. Then, one can map e_1 to a path containing the edge of G_2 with the True labeling inside $T_\varepsilon(e_1)$. Now, we want to map e_2 to a path P starting at some point of p_1, where P contains the edge of G_2 with the True labeling inside $T_\varepsilon(e_2)$. Thus P must contain w_3 and w_1. As $\delta_F(e_2, P) > \varepsilon$ (here, we only have $\delta_{wF}(e_2, P) \leq \varepsilon$) for any such path P, this labeling is not realizable. It is easy to see that any other labeling of paths e_1 and e_2 are mapped to is realizable.

A Permute gadget is a differently labeled Variable gadget. For the Split gadget, we add a third edge e_3 of G_1 to the Variable gadget and add edges of G_2 from w_2 and w_3 inside the ε-tube around e_3. For the labeling, see Figure 2. A False signal can not be converted to a True signal in Split gadget or in the Permute gadget. However, a True signal can but does not need to be converted to a False signal in the Permute gadget.

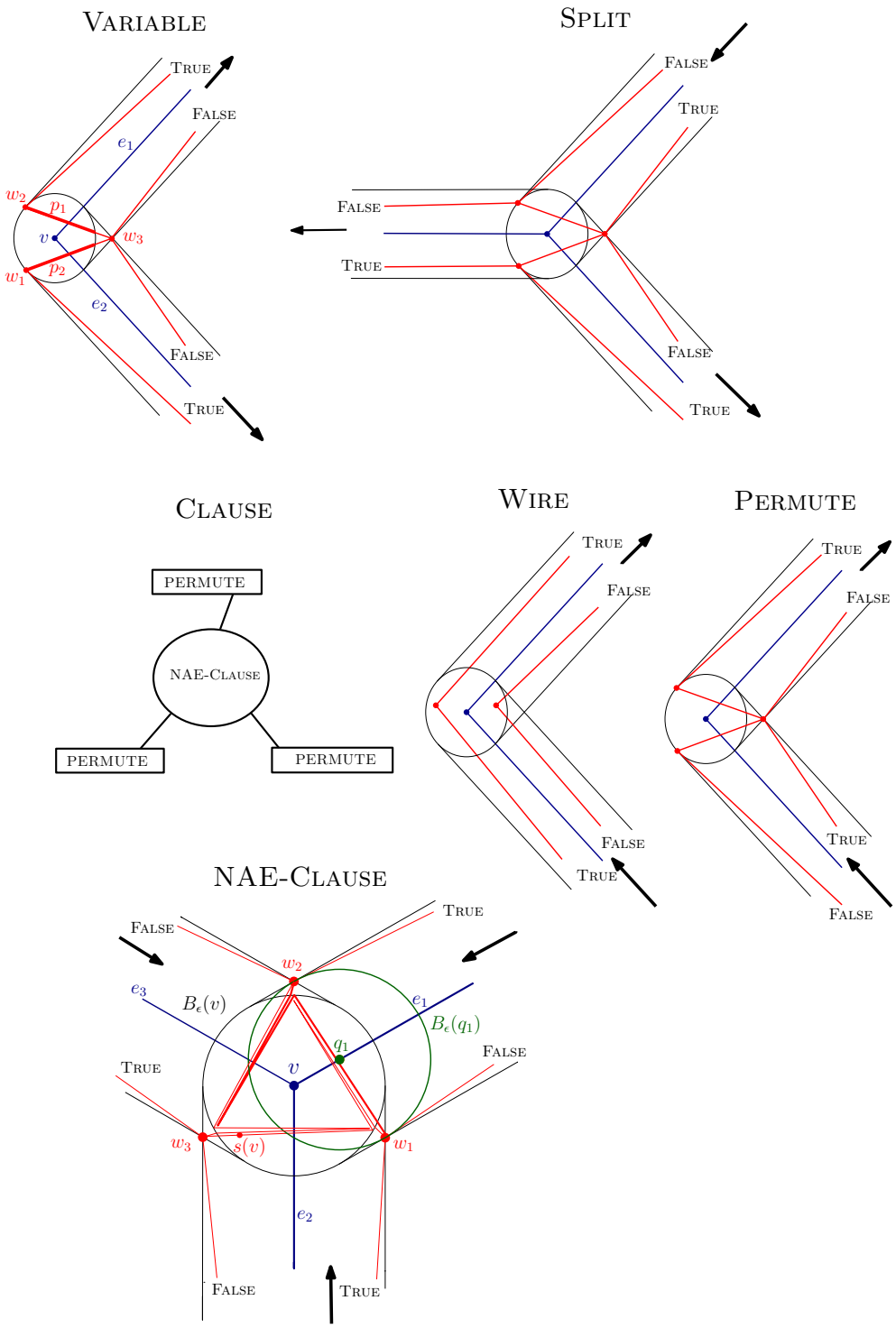


Figure 2 Gadgets to build a graph-similarity instance given a MONOTONE-PLANAR-3-SAT instance.

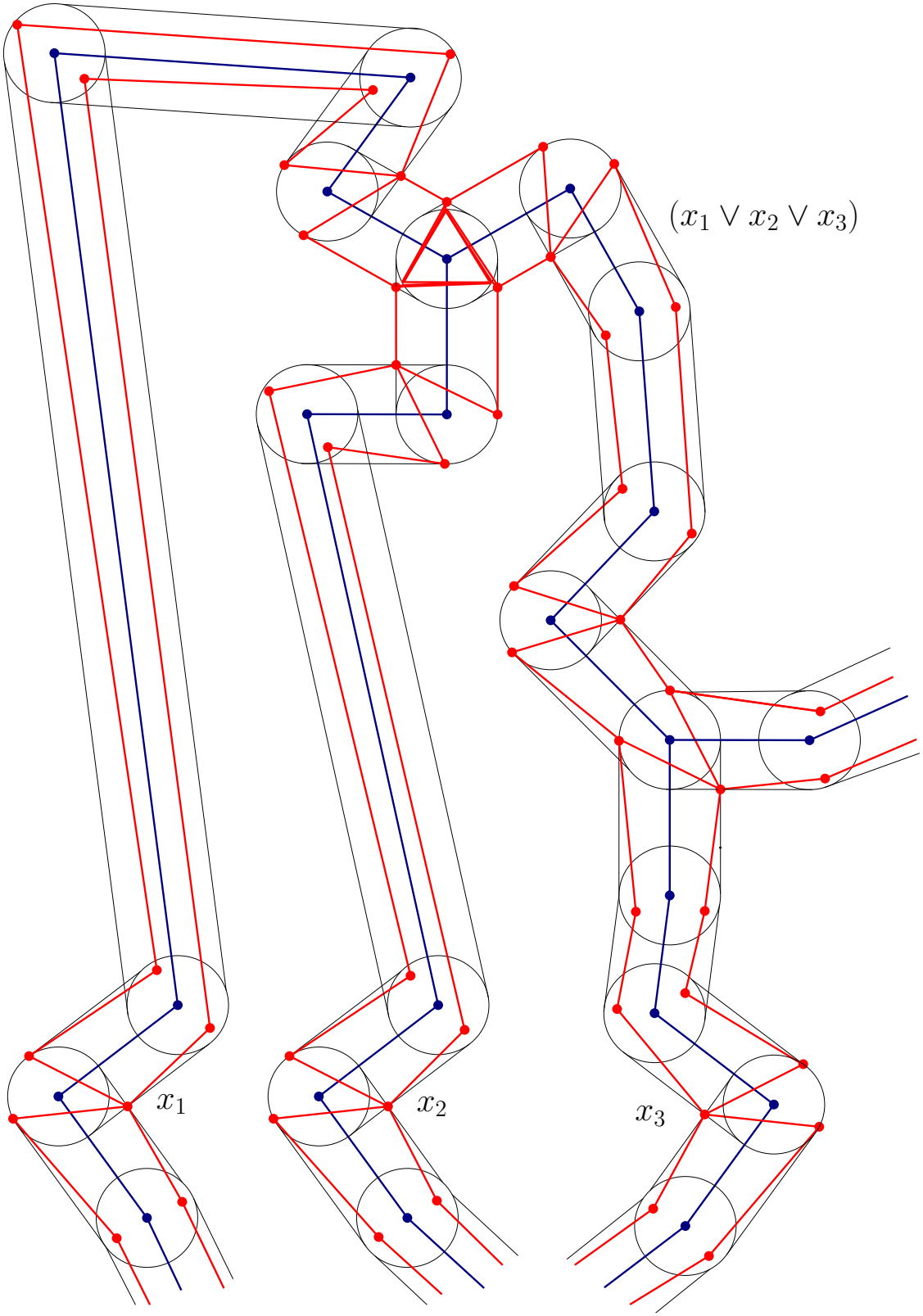


Figure 3 Construction of one CLAUSE gadget given the MP3S instance A with variables $V = \{x_1, x_2, \ldots, x_5\}$ and clauses $C = \{(x_1 \vee x_2 \vee x_3), (x_3 \vee x_4 \vee x_5), (\bar{x_1} \vee \bar{x_3} \vee \bar{x_5})\}$

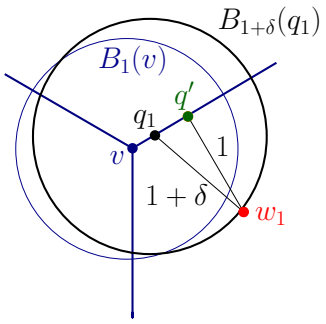


Figure 4 Illustration of the proof of Theorem 2.

For the Clause gadget, we first introduce a NAE-Clause gadget where it is required that not all three values in a clause are equal. For the construction, see Figure 2. Let q_1 be the point on e_1 with distance ε to w_1 and w_2. To force walking back and forth along e_1 for a combination of labels which we want to exclude, we have to ensure that a path from w_1 to w_2 leaves $B_\varepsilon(q_1)$ but stays, once entered, inside $B_\varepsilon(v)$. For the other pairs, (w_1, w_3) and (w_2, w_3) we do the same. A possible drawing of these paths maintaining the planarity of G_2 is shown in the lower sketch of Figure 2. Suppose we map v to $s(v)$ as shown in the Figure 2. Then, edges e_2 and e_3 can be mapped to paths through edges labeled True. But we cannot map e_1 to such a path P: When P reaches vertex w_1, any corresponding reparameterization of e_1 realizing $\delta_F(e_1, P) \leq \varepsilon$ must have reached q_1 as q_1 is the only point with distance at most ε to w_1 on e_2. As P leaves $B_\varepsilon(q_1)$ between w_1 and w_2 and any point on e_1 with distance at most ε to the part of P outside $B_\varepsilon(q_1)$ lies between v and q_1 it follows that $\delta_F(e_1, P) > \varepsilon$. For symmetric reasons it follows that any other all-equal labeling cannot be realized. However, there is a placement of v, such that all three edges e_1, e_2 and e_3 can be mapped to a path in G_2 with Fréchet distance at most ε, for every not-all-equal labeling. Note that Monotone-Planar-NAE-3-Sat is in P but, as shown in Figure 2, we can use the NAE-Clause gadget as the core of the Clause gadget.

Placing the other gadgets with no overlap (using the Wire gadget) and noting that all constructed subgraphs are plane, we can, given a MP3S instance A, construct plane graphs G_1 and G_2 such that a map from G_1 to G_2, which realizes $\vec{\delta}_G(G_1, G_2) < \varepsilon$, induces a solution of A: For each positive NAE-clause, at least one of the outgoing edges of G_1 must be mapped to a path through an edge labeled True and thus the corresponding variable v gets the value True. In this case, v cannot set any of the negative clauses True, because the other outgoing edge must be mapped to a path through the edge of G_2 labeled False and this signal can never be switched to True. The same holds for the case of negative NAE-clauses.

Conversely, given a solution S of the MP3S instance A, we can construct a placement of G_1 by choosing p_1 in the Variable gadget for each variable with a True label in S and p_2 for each variable with a False label in S. All edges of the other gadget can be mapped to G_2 in a signal preserving manner. Note that if there exists a clause C in A with three positive labeled variables in S, we change one signal in the Permute gadget from True to False. Thus, we have found a mapping realizing $\vec{\delta}_G(G_1, G_2) \leq \epsilon$. ◀

The characteristics of the gadget still hold for a slightly bigger value of ε which leads to:

▶ **Theorem 2.** *For plane graphs G_1, G_2 it is NP-hard to approximate the directed graph distance $\vec{\delta}_G(G_1, G_2)$ within a* 1.10566 *factor.*

Proof. We give a detailed proof for the NAE-Clause gadget and note that a similar argument holds for the other gadgets. See Figure 4 for an illustration of the proof.

Let us fix $\varepsilon = 1$. We draw the green spike in the NEA-Clause gadget such that its peak is arbitrarily close to the intersection of a straight line through the edge e_1 and the 1-circle around v. Now, we need to compute the smallest value δ_{min}, such that $B_1(v)$ is completely contained in $B_{1+\delta_{min}}(q_1)$. Then, for any value $\delta < \delta_{min}$, there exists a drawing of the spikes, such that the characteristics of the NAE-Clause gadget still hold, e.g., there is no placement of v allowing an all-equal-labeling.

Note that δ_{min} equals the distance from q_1 to v, when q_1 is at distance $1 + \delta_{min}$ to w_1. Let q' be the position of q_1 for $\delta = 0$ and let d be the distance between q' and q_1. Then we have $\tan(30°) = \frac{\delta_{min}+d}{1} = \delta_{min} + d$. Furthermore, we have $d = \sqrt{(1+\delta_{min})^2 - 1}$ and therefore $\delta_{min} = \tan(30°) - \sqrt{(1+\delta_{min})^2 - 1}$, which solves to $\delta_{min} = \frac{1}{4} - \frac{1}{4\sqrt{3}} \approx 0.10566$. The factor by which ε can be multiplied is greater than $1 + \delta_{min}$ for all other gadgets. Thus, δ_{min} is the critical value for the whole construction and the theorem follows. ◀

3 Algorithms for Plane Graphs

Our general algorithm consists of the following four steps. **1.** Compute ε-placements of vertices, **2.** Compute reachability information, **3.** Prune invalid ε-placements, **4.** Based on the remaining ε-placements, decide if there exists a mapping from G_1 to G_2 realizing $\vec{\delta}_{(w)G}(G_1, G_2) \leq \varepsilon$. See [6, 10] for a detailed presentation.

Deciding the Graph Distance in Exponential Time A brute-force method to decide the directed graph distance is to iterate over all possible combinations of valid vertex placements. For each such combination, we iterate over all edges of G_1 to determine whether the vertex placements allow to map each edge to a path with Fréchet distance smaller than ε. This can be done in constant time per edge using the previously computed reachability information. Thus, the runtime is $O(m_1 \cdot m_2^{n_1})$, where $n_i = |V_i|$ and $m_i = |E_i|$.

An alternative approach is the following, which in essence is an extension of step 3 of the general algorithm. First, we remove all tree-like substructures of G_1 and place these as described in [10]. Next, we decompose the remainder of G_1 into chordless cycles, where a chord is a maximal path in G_1 incident to two faces (see Figure 5). We place the parts of G_1 from bottom up, deciding in each step if we can place two adjacent cycles and all the nested substructures of the cycles simultaneously. The time and space complexity of this approach are summarized in the following Theorem. For more details and a proof, see [6].

▶ **Theorem 3.** *For plane graphs, the graph distance can be decided in $O(Fm_2^{2F-1})$ time and $O(m_2^{2F-1})$ space, where F is the number of faces of G_1.*

Note that this method is superior to the brute-force method if $2F - 1 < n_1$.

Polynomial Time Approximation Algorithm The general algorithmic approach yields a good approximation for deciding the graph distance for plane graphs with some geometric restrictions. Again, the decision is based on the existence of valid ε-placements. Thus, the runtime is the same as for the case where G_1 is a tree ($O(n_1 \cdot m_2^2)$ time and space).

▶ **Theorem 4.** *Let $G_1 := (V_1, E_1)$ and $G_2 := (V_2, E_2)$ be plane graphs. Assume that each edge of G_1 has length greater than 2ε. Let α_v be the smallest angle between two edges of G_1 incident to vertex v with $deg(v) \geq 3$, and let $\alpha := \frac{1}{2}\min_{v \in V_1}(\alpha_v)$. If there exists at least one valid ε-placement for each vertex of G_1, then $\vec{\delta}_G(G_1, G_2) \leq \frac{1}{\sin(\alpha)}\varepsilon$.*

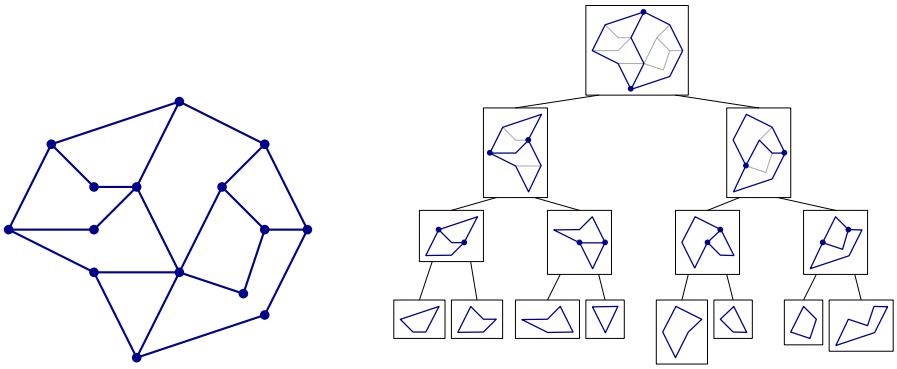


Figure 5 A plane graph is recursively decomposed into chordless cycles.

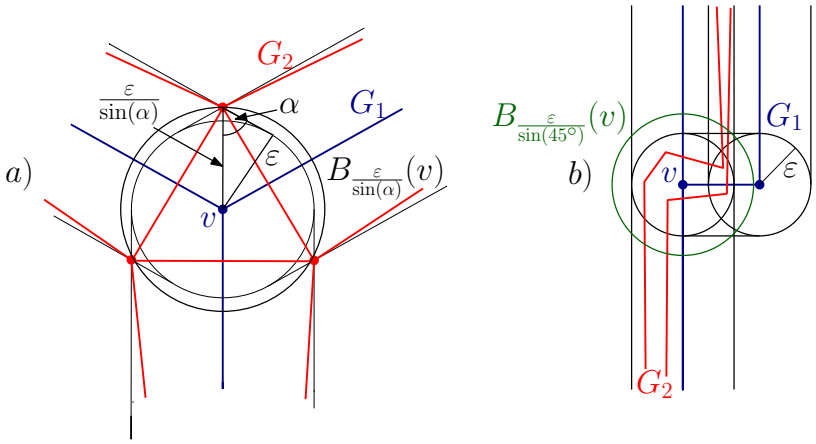


Figure 6 Illustration of the proof of Theorem 4

.

Proof. Let α be the smallest angle between two edges incident to a vertex v with degree at least three and let $C_1, C_2, \ldots, C_j$ be the valid placements of v for a given distance value ε. Furthermore, let V_{C_i} be the set of vertices of C_i. It can be easily shown that for a larger distance value of $\varepsilon_1 \geq \frac{1}{\sin(\alpha)}\varepsilon$ there exist vertices $v_1, v_2, \ldots, v_k$, embedded inside B_{ε_1}, such that the subgraph $C = (V', E')$, where $V' = \bigcup_{i=1}^{j} V_{C_i} \cup \{v_1\} \cup \{v_2\} \cup \cdots \cup \{v_k\}$ and $E' = \{e = \{uw\} \in E_2 | u \in V', w \in V'\}$ is connected (see Figure 6a)). Note that this property is not true if we allow edges with length smaller than 2ε (see Figure 6b)). However, with the condition of a minimal edge length of 2ε, there is only one valid $\frac{1}{\sin(\alpha)}\varepsilon$-placement C for each vertex with degree at least three. Furthermore, every valid ε placement is a valid $\frac{1}{\sin(\alpha)}\varepsilon$-placement. Now, for two paths which start and/or end at a common vertex v, v is mapped to a point on C. This ensures that each edge of G_1 is mapped correctly. ◀

References

1 Mahmuda Ahmed, Brittany T. 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 Mahmuda Ahmed and Carola Wenk. Constructing street networks from gps trajectories. In *Proceedings of the 20th Annual European Conference on Algorithms*, ESA'12, pages 60–71, Berlin, Heidelberg, 2012. Springer-Verlag. URL: `http://dx.doi.org/10.1007/978-3-642-33090-2_7`, `doi:10.1007/978-3-642-33090-2_7`.

6 Hugo A. Akitaya, Maike Buchin, Bernhard Kilgus, Stef Sijben, and Carola Wenk. Distance measures for embedded graphs. arXiv e-print, 2018. `https://arxiv.org/abs/1812.09095`.

7 Helmut Alt, Alon Efrat, Günter Rote, and Carola Wenk. Matching planar maps. *Journal of Algorithms*, 49(2):262 – 283, 2003.

8 Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5(1&2):75–91, 1995.

9 James Biagioni and Jakob Eriksson. Inferring road maps from global positioning system traces: Survey and comparative evaluation. *Transportation Research Record: Journal of the Transportation Research Board*, 2291:61–71, 2012.

10 Maike Buchin, Stef Sijben, and Carola Wenk. Distance measures for embedded graphs. In *Proc. 33rd European Workshop on Computational Geometry (EuroCG)*, pages 37–40, 2017.

11 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.

12 Jonathan J. Davies, Alastair R. Beresford, and Andy Hopper. Scalable, distributed, real-time map generation. *IEEE Pervasive Computing*, 5(4):47–54, 2006.

13 Sophia Karagiorgou and Dieter Pfoser. On vehicle tracking data-based road network generation. In *20th ACM SIGSPATIAL GIS*, pages 89–98, 2012.

Approximate strong edge-colouring of unit disk graphs

Nicolas Grelier[1], Rémi de Joannis de Verclos[2], Ross J. Kang[3], and François Pirot[4]

1 **Department of Computer Science, ETH Zürich**
nicolas.grelier@inf.ethz.ch

2 **Department of Mathematics, Radboud University Nijmegen**
r.deverclos@math.ru.nl

3 **Department of Mathematics, Radboud University Nijmegen**
ross.kang@gmail.com

4 **LORIA, Université de Lorraine and Department of Mathematics, Radboud University Nijmegen**
francois.pirot@loria.fr

Abstract

We show that the strong chromatic index of unit disk graphs is efficiently 6-approximable. This improves on 8-approximability as shown by Barrett, Istrate, Kumar, Marathe, Thite, and Thulasidasan (2006). We also show that strong edge-6-colourability is NP-complete for the class of unit disk graphs. Thus there is no polynomial-time $(7/6 - \varepsilon)$-approximation unless P=NP.

1 Introduction

A *strong edge-k-colouring* is a partition of the edges of a graph into k parts so that each part induces a matching. The *strong chromatic index* is the least k for which the graph admits a strong edge-k-colouring. If the vertices of the graph represent communicating nodes, say, in a wireless network, then an optimal strong edge-colouring may represent an optimal discrete assignment of frequencies to transmissions in the network so as to avoid both primary and secondary interference [20, 18, 1]. It is then relevant to model the network geometrically, i.e. as a *unit disk graph* [10]. Our interest is in approximative algorithmic aspects of strong edge-colouring in this model. This was considered by Barrett, Istrate, Kumar, Marathe, Thite, and Thulasidasan [1] who showed that the strong chromatic index of unit disk graphs is efficiently 8-approximable. We revisit the problem and make some further advances.

- We prove efficient 6-approximability.
- We prove efficient online 8-approximability.
- We show impossibility of efficient $(7/6 - \varepsilon)$-approximation unless P=NP.

It is $\exists\mathbb{R}$-complete to decide if a given graph has an embedding as a unit disk graph [12], but both of the approximation algorithms we use are *robust* in the sense that they efficiently output a valid strong edge-colouring upon the input of any abstract graph. Our contribution is to prove that they are guaranteed to output a colouring with good approximation ratio upon the input of a unit disk graph (regardless of any embedding).

Our work parallels and contrasts with work on the chromatic number of unit disk graphs, for which the best approximation ratio known has remained 3 since 1991 [19]. Before stating our main results in detail in Section 3 below, we first review some background material.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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

In this section, we highlight some graph theoretic notation, concepts and observations that are relevant to our study. For other standard background, consult e.g. [7]. Given a graph $G = (V, E)$, the *minimum degree*, *clique number*, *chromatic number* and *maximum degree* of G are denoted by $\delta(G)$, $\omega(G)$, $\chi(G)$ and $\Delta(G)$, respectively. The *degeneracy* of G is defined as $\delta^*(G) = \max\{\delta(H) \mid H \subseteq G\}$ and G is called *k-degenerate* if $\delta^*(G) \leq k$. A simple but useful set of inequalities for graph colouring is as follows. For any graph G,

$$\omega(G) \leq \chi(G) \leq \delta^*(G) + 1 \leq \Delta(G) + 1. \tag{1}$$

Note that the second inequality in (1) is algorithmic, in the sense that it follows from the use of an efficient greedy algorithm that always assigns the least available colour, provided we consider the vertices one by one in a suitable order, namely, according to degeneracy. Moreover, a greedy algorithm taking any ordering uses at most $\Delta(G) + 1$ colours.

The *line graph* of G is denoted $L(G)$. The *square* G^2 of G is the graph formed from G by adding all edges between pairs of vertices that are connected by a 2-edge path in G. The *strong chromatic index* of G (as defined above) is denoted $\chi_2'(G)$. Note that $\chi_2'(G) = \chi(L(G)^2)$. The *strong clique number* $\omega_2'(G)$ of G is $\omega(L(G)^2)$. Obviously, (1) implies that

$$\omega_2'(G) \leq \chi_2'(G) \leq \delta^*(L(G)^2) + 1 \leq \Delta(L(G)^2) + 1. \tag{2}$$

It is worth reiterating that the following greedy algorithm efficiently generates a strong edge-$(\delta^*(L(G)^2) + 1)$-colouring: order the edges of G by repeatedly removing from G an edge e for which $\deg_{L(G)^2}(e)$ is lowest, and then colour the edges sequentially according to the *reverse* of this ordering, at each step assigning as a colour the least positive integer that does not conflict with previously coloured edges. Again similarly, with an arbitrary ordering of the edges the greedy algorithm produces a strong edge-$(\Delta(L(G)^2)+1)$-colouring. Our main results will then follow from (2) by suitable bounds on $\delta^*(L(G)^2)$ and $\Delta(L(G)^2)$.

The strong chromatic index is a well-studied parameter in graph theory. Most notably, Erdős and Nešetřil conjectured in the 1980s that $\chi_2'(G) \leq 1.25\Delta(G)^2$ for all graphs G [8]. About a decade later, Molloy and Reed [17] proved the existence of some minuscule but fixed $\varepsilon > 0$ such that $\chi_2'(G) \leq (2 - \varepsilon)\Delta(G)^2$ for all graphs G. Recently there have been improvements [3, 2] and extensions [11, 6], but all rely on Molloy and Reed's original approach, a reduction to a Ramsey-type colouring result. The conjecture remains wide open.

A graph $G = (V, E)$ is said to be a *unit disk graph* if there exists a mapping $p : V \to \mathbb{R}^2$ from its vertices to the plane such that $uv \in E$ if and only if the Euclidean distance between $p(u)$ and $p(v)$ is at most 1. Any explicit mapping p that certifies that G is a unit disk graph is called an *embedding*. When we have an embedding p, we often make no distinction between a vertex u and its corresponding point $p(u)$ in the plane.

The class of unit disk graphs is popular due to its elegance and its versatility in capturing real-world optimisation problems [5]. For example, an embedded unit disk graph may represent placement of transceivers so that circles of radius $1/2$ centred at the points represent transmission areas. Indeed, the class was originally introduced in 1980 to model frequency assignment [10], with chromatic number one of the first studied parameters. Clark, Colbourn and Johnson [5] published a proof that it is NP-hard to compute the chromatic number of

unit disk graphs. They also showed the clique number of unit disk graphs is polynomial-time computable. Therefore, any upper bound C on the extremal ratio

$$r := \sup\{\chi(G)/\omega(G) \mid G \text{ is a unit disk graph}\}$$

(algorithmic or not) implies an efficient C-approximation of the chromatic number: simply output $C \cdot \omega(G)$. In 1991, Peeters [19] noted a simple 3-approximation: after lexicographically ordering the vertices of G according to any fixed embedding, a basic geometric argument proves that G is $3(\omega(G) - 1)$-degenerate (and then apply (1)). Since 3-colourability of unit disk graphs is NP-complete, there is no efficient $(4/3 - \varepsilon)$-approximation unless P=NP. Moreover, Malesińska, Piskorz and Weißenfels [16] exhibited some unit circular-arc graphs that certify $r \geq 3/2$. To date, the best approximation ratio known is 3.

Mahdian [14, 15] showed in 2000 that it is NP-hard to compute the strong chromatic index, even restricted to bipartite graphs of large fixed girth. More recently, Chalermsook, Laekhanukit and Nanongkai [4] showed that in general there is no polynomial-time $(n^{1/3-\varepsilon})$-approximation algorithm (where n is the number of vertices in the input) unless NP=ZPP.

To the best of our knowledge, no previous work has shown NP-hardness upon restriction to the class of unit disk graphs. Nevertheless, Barrett *et al.* [1] have initiated the study of approximate strong edge-colouring for unit disk graphs. With an argument similar to Peeters' [19] for chromatic number, they showed that $\delta^*(L(G)^2) \leq 8\omega_2'(G)$ for any unit disk graph G, which by (2) certifies that the greedy algorithm is an efficient 8-approximation for the strong chromatic index. Kanj, Wiese and Zhang [13] noted an efficient online 10-approximation for the strong chromatic index with essentially the same analysis as in [1].

3 Main results

Our work improves on [1] in several ways.

▶ **Theorem 3.1.** *For any unit disk graph G, $\delta^*(L(G)^2) \leq 6(\omega_2'(G) - 1)$.*

▶ **Corollary 3.2.** *The greedy algorithm under a reverse degeneracy ordering of the edges is an efficient 6-approximation for the strong chromatic index of unit disk graphs.*

The proof of Theorem 3.1 is rather involved. It shows that, for any embedded unit disk graph, some well-chosen edge-ordering certifies the required degeneracy bound. It would be very interesting to improve on the approximation ratio of 6.

▶ **Theorem 3.3.** *For any unit disk graph G, $\Delta(L(G)^2) \leq 8(\chi_2'(G) - 1)$.*

▶ **Corollary 3.4.** *The greedy algorithm is an efficient online 8-approximation[1] for the strong chromatic index of unit disk graphs.*

To prove Theorem 3.3, it suffices to solve the following kissing number-type problem. Given two intersecting unit disks C_1 and C_2 in $\mathbb{R}^2$, what is the size of a largest collection of pairwise non-intersecting unit disks such that each one intersects $C_1 \cup C_2$? The corresponding problem in $\mathbb{R}^3$ seems quite natural.

[1] To avoid any ambiguity, in the online setting *vertices* are revealed one at a time and all edges between a newly revealed vertex and previous vertices must be immediately and irrevocably assigned a colour.

▶ **Theorem 3.5.** *Strong edge-k-colourability of unit disk graphs is NP-complete, where $k = 6$ or $k = \binom{\ell}{2} + 4\ell + 6$ for some fixed $\ell \geq 5$.*

▶ **Corollary 3.6.** *It is NP-hard to compute the strong chromatic index of unit disk graphs. Moreover, it cannot be efficiently $(7/6 - \varepsilon)$-approximated unless P=NP.*

For $k \leq 3$, strong edge-k-colouring is polynomially solvable. It remains open to determine the complexity when k is in $\{4, 5\}$. The proof of Theorem 3.5 borrows upon ideas in the work of Gräf, Stumpf and Weißenfels [9], but with extra non-trivial difficulties for strong edge-colouring.

4 Further discussion

We can state slightly more general versions of our approximation results that give a more geometric flavour. We call a graph $G = (V, E)$ a *twin unit disk graph* if there exists a mapping $p : V \to \mathbb{R}^2 \times \mathbb{R}^2$ from its vertices to pairs of points in the plane such that

- the Euclidean distance between $p(u)_1$ and $p(u)_2$ is at most 1 for every $u \in V$; and
- $uv \in E$ if and only if the Euclidean distance between $p(u)_1$ and $p(v)_1$, between $p(u)_1$ and $p(v)_2$, between $p(u)_2$ and $p(v)_1$, or between $p(u)_2$ and $p(v)_2$ is at most 1.

Equivalently, this is the intersection class over unions of pairs of intersecting unit disks in $\mathbb{R}^2$. Note that, for any unit disk graph G, both G and $L(G)^2$ are twin unit disk graphs. Indeed for any edge $e = (p_1, p_2)$ in a unit disk graph, e can be represented by a vertex u in a twin unit disk with $p(u)_1 = p_1$ and $p(u)_2 = p_2$. So it is NP-hard to determine the chromatic number of twin unit disk graphs. We actually found the following stronger versions of Theorems 3.1 and 3.3, which imply efficient 6-approximation and online 8-approximation for the chromatic number of twin unit disk graphs (by (1)).

▶ **Theorem 4.1.** *For any twin unit disk graph G, $\delta^*(G) \leq 6(\omega(G) - 1)$.*

▶ **Theorem 4.2.** *For any twin unit disk graph G, $\Delta(G) \leq 8(\chi(G) - 1)$.*

If we were able to efficiently compute or well approximate the clique number of twin unit disk graphs or, in particular, the strong clique number of unit disk graphs, then we would have a strong incentive to bound $r'_2 := \sup\{\chi'_2(G)/\omega'_2(G) \mid G \text{ is a unit disk graph}\}$. This is a natural optimisation problem regardless. We only know $r'_2 \leq 6$ by Theorem 3.1, and $r'_2 \geq 4/3$ by considering the cycle C_7 on seven vertices (since $\chi'_2(C_7) = 4$ while $\omega'_2(C_7) = 3$). Relatedly, we believe that the following problem is worth investigating.

▶ Conjecture 4.3. It is NP-hard to compute the clique number of twin unit disk graphs.

5 A short proof for a 7-approximation

For expository purposes, we present a much shorter argument for a weaker approximation than Theorem 4.1. The proof is nearly the same as what Barrett *et al.* [1] used for an upper bound on the approximation ratio of 8, but with a small twist.

▶ Proposition 5.1. For any twin unit disk graph G, $\delta^*(G) \leq 7(\omega(G) - 1)$.

Proof. Let $G = (V, E)$ be a twin unit disk graph. Fix any embedding $p : V \to \mathbb{R}^2 \times \mathbb{R}^2$ of G in the plane. Equipped with such an embedding, we first define an ordering of V and then use it to certify the promised degeneracy property.

The ordering we use for this result, a lexicographic ordering, is the same used in [1]. Let $(x_1, y_1), (x_2, y_2), \ldots$ be a sequence of points in $\mathbb{R}^2$ defined by listing the elements of

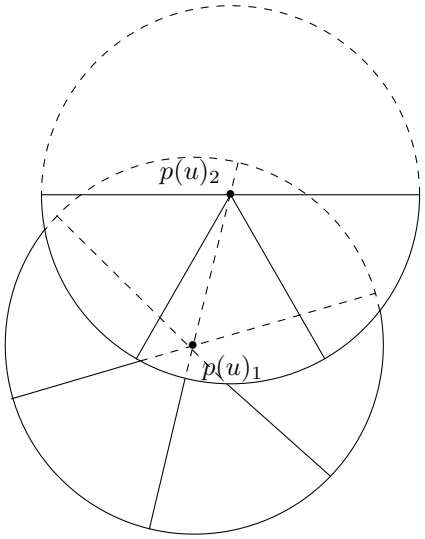


Figure 1 The seven sectors one of which must contain $p(v)_1$ or $p(v)_2$.

$\bigcup_{u\in V}\{p(u)_1, p(u)_2\}$ according to the lexicographic order on $\mathbb{R}^2$ (i.e. (a, b) is before (c, d) if and only if $a < c$ or ($a = c$ and $b \leq d$)). We consider the points of this sequence in order and add vertices at the end of our current ordering of V as follows. When considering point (x_j, y_j) for some $j \geq 1$, we add all vertices $u \in V$ for which there is some $i \leq j$ such that $\{p(u)_1, p(u)_2\} = \{(x_i, y_i), (x_j, y_j)\}$, and we do so according to the lexicographic order on $\mathbb{R}^2$.

It suffices to show that each vertex $u \in V$ has at most $7(\omega(G) - 1)$ neighbours that precede it in the lexicographic ordering. To do so, we show that every such neighbour v of u satisfies that either $p(v)_1$ or $p(v)_2$ is contained in one of seven unit $(\pi/3)$-sectors (each of which is centred around either $p(u)_1$ or $p(u)_2$). This is enough, since the set of vertices that map one of their twin points into one such sector induces a clique in G that includes u.

Let $u \in V$ and suppose without loss of generality that $p(u)_1$ is before $p(u)_2$ in lexicographic order. First observe that, if $v \in V$ is before u in the lexicographic order, then both $p(v)_1$ and $p(v)_2$ must be in the region of $\mathbb{R}^2$ that has smaller or equal y-coordinate compared to $p(u)_2$. If, moreover $uv \in E$, then $p(v)_1$ or $p(v)_2$ must lie in either a unit half-disk centred at $p(u)_2$ or in the unit disk centred at $p(u)_1$. We partition the unit disk centred at $p(u)_1$ into six unit $(\pi/3)$-sectors such that the line segment $[p(u)_1, p(u)_2]$ lies along the boundary between two of the sectors. Note that any of the points in the two sectors incident to $[p(u)_1, p(u)_2]$ also lies in the unit disk centred at $p(u)_2$. See Figure 1. Therefore, the four other sectors together with the three sectors that partition the unit half-disk centred at $p(u)_2$ are the seven unit $(\pi/3)$-sectors that we desire. ◀

It turns out that for Theorem 4.1 we can take the same approach as in Proposition 5.1, except with an ordering that is more subtle and an analysis that is substantially longer and more difficult. Full proof details will be made available in a journal version of the manuscript (also on arXiv).

Acknowledgments. This work was supported by a Vidi grant (639.032.614) of the Netherlands Organisation for Scientific Research (NWO).

References

1 Christopher L Barrett, Gabriel Istrate, VS Anil Kumar, Madhav V Marathe, Shripad Thite, and Sunil Thulasidasan. Strong edge coloring for channel assignment in wireless radio networks. In *Pervasive Computing and Communications Workshops, 2006*, page 5. IEEE, 2006.

2 M. Bonamy, T. Perrett, and L. Postle. Colouring Graphs with Sparse Neighbourhoods: Bounds and Applications. *ArXiv e-prints*, October 2018. arXiv:1810.06704.

3 Henning Bruhn and Felix Joos. A stronger bound for the strong chromatic index. *Combin. Probab. Comput.*, 27(1):21–43, 2018. URL: https://doi.org/10.1017/S0963548317000244, doi:10.1017/S0963548317000244.

4 Parinya Chalermsook, Bundit Laekhanukit, and Danupon Nanongkai. Coloring graph powers: graph product bounds and hardness of approximation. In *LATIN 2014*, volume 8392 of *Lecture Notes in Comput. Sci.*, pages 409–420. Springer, Heidelberg, 2014. URL: https://doi.org/10.1007/978-3-642-54423-1_36, doi:10.1007/978-3-642-54423-1_36.

5 Brent N Clark, Charles J Colbourn, and David S Johnson. Unit disk graphs. *Discrete Mathematics*, 86(1-3):165–177, 1990.

6 Rémi de Joannis de Verclos, Ross J. Kang, and Lucas Pastor. Colouring squares of claw-free graphs. To appear in *Canadian Journal of Mathematics*, 2018.

7 Reinhard Diestel. *Graph theory*, volume 173 of *Graduate Texts in Mathematics*. Springer, Berlin, fifth edition, 2017. URL: https://doi.org/10.1007/978-3-662-53622-3, doi:10.1007/978-3-662-53622-3.

8 Paul Erdős. Problems and results in combinatorial analysis and graph theory. In *Proceedings of the First Japan Conference on Graph Theory and Applications (Hakone, 1986)*, volume 72, pages 81–92, 1988. URL: https://doi.org/10.1016/0012-365X(88)90196-3, doi:10.1016/0012-365X(88)90196-3.

9 Albert Gräf, Martin Stumpf, and Gerhard Weißenfels. On coloring unit disk graphs. *Algorithmica*, 20(3):277–293, 1998.

10 W. K. Hale. Frequency assignment: Theory and applications. *Proceedings of the IEEE*, 68(12):1497–1514, Dec 1980. doi:10.1109/PROC.1980.11899.

11 Tomáš Kaiser and Ross J. Kang. The distance-t chromatic index of graphs. *Combin. Probab. Comput.*, 23(1):90–101, 2014. URL: https://doi.org/10.1017/S0963548313000473, doi:10.1017/S0963548313000473.

12 Ross J. Kang and Tobias Müller. Sphere and dot product representations of graphs. *Discrete Comput. Geom.*, 47(3):548–568, 2012. URL: https://doi.org/10.1007/s00454-012-9394-8, doi:10.1007/s00454-012-9394-8.

13 Iyad A. Kanj, Andreas Wiese, and Fenghui Zhang. Local algorithms for edge colorings in UDGs. *Theoret. Comput. Sci.*, 412(35):4704–4714, 2011. URL: https://doi.org/10.1016/j.tcs.2011.05.005, doi:10.1016/j.tcs.2011.05.005.

14 Mohammad Mahdian. The strong chromatic index of graphs. Master's thesis, University of Toronto, 2000.

15 Mohammad Mahdian. On the computational complexity of strong edge coloring. *Discrete Appl. Math.*, 118(3):239–248, 2002. URL: https://doi.org/10.1016/S0166-218X(01)00237-2, doi:10.1016/S0166-218X(01)00237-2.

16 Ewa Malesińska, Steffen Piskorz, and Gerhard Weißenfels. On the chromatic number of disk graphs. *Networks*, 32(1):13–22, 1998.

17 Michael Molloy and Bruce Reed. A bound on the strong chromatic index of a graph. *J. Combin. Theory Ser. B*, 69(2):103–109, 1997. URL: https://doi.org/10.1006/jctb.1997.1724, doi:10.1006/jctb.1997.1724.

18 Thyagarajan Nandagopal, Tae-Eun Kim, Xia Gao, and Vaduvur Bharghavan. Achieving MAC layer fairness in wireless packet networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, pages 87–98. ACM, 2000.

19 René Peeters. On coloring j-unit sphere graphs. Technical report, Tilburg University, 1991.

20 Subramanian Ramanathan and Errol L Lloyd. Scheduling algorithms for multihop radio networks. *IEEE/ACM Transactions on Networking (TON)*, 1(2):166–177, 1993.

Recognizing embedded caterpillars with weak unit disk contact representations is NP-hard*

Man-Kwun Chiu[†1], Jonas Cleve[‡2], and Martin Nöllenburg[§3]

1 **Institut für Informatik, Freie Universität Berlin**
chiumk@zedat.fu-berlin.de
2 **Institut für Informatik, Freie Universität Berlin**
jonascleve@inf.fu-berlin.de
3 **Institute of Logic and Computation, Technische Universität Wien**
noellenburg@ac.tuwien.ac.at

Abstract

Weak unit disk contact graphs are graphs that admit a representation of the nodes as a collection of internally disjoint unit disks whose boundaries touch if there is an edge between the corresponding nodes. We provide a gadget-based reduction to show that recognizing embedded caterpillars that admit a weak unit disk contact representation is NP-hard.

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. It is well known that the disk contact graphs are exactly the planar graphs [7]. If, however, all disks must be of the same size, the recognition problem is NP-hard [3]. Investigating the precise boundary between hardness and tractability for recognizing unit and weighted disk contact graphs has been the subject of some recent work [1, 2, 4, 5]. For instance, recognizing embedded trees admitting a unit disk contact representation (UDCR) is NP-hard [2], while the problem is trivial for paths or stars. In this paper we study the open problem of recognizing embedded caterpillars that have an embedding-preserving UDCR.

A *caterpillar* $C = (V, E)$ is a tree whose internal nodes form a path, i.e., after removing all leaves from C a *backbone path* remains. Accordingly we introduce the notions of *leaf* and *backbone* nodes and disks of C. Klemz et al. [5] showed that for caterpillars without a given embedding it can be decided in linear time whether a UDCR exists. Yet, if the cyclic order of the neighbors of each node $v \in V$, i.e., the *embedding* of C, is specified and must be preserved, we show that the decision problem is NP-hard, at least in the following weaker sense. In a *weak* UDCR of a caterpillar we still require that the disks of any two adjacent nodes in C must touch, yet we also allow that non-adjacent disks touch. According to this definition we can obtain dense circle packings on a hexagonal grid with a maximum node degree of 6, while according to the original definition of (*strong*) UDCRs proper gaps must exist between any pair of non-adjacent nodes and hence if the graph is a tree all nodes have degree at most 5. Generalizing the NP-hardness to strong UDCRs remains an open question.

* Research partly supported by the German Research Foundation within the collaborative DACH project *Arrangements and Drawings* as DFG Project MU 3501/3-1.
† Supported by ERC StG 757609.
‡ Supported by ERC StG 757609.
§ Supported by FWF grant AJS 399.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

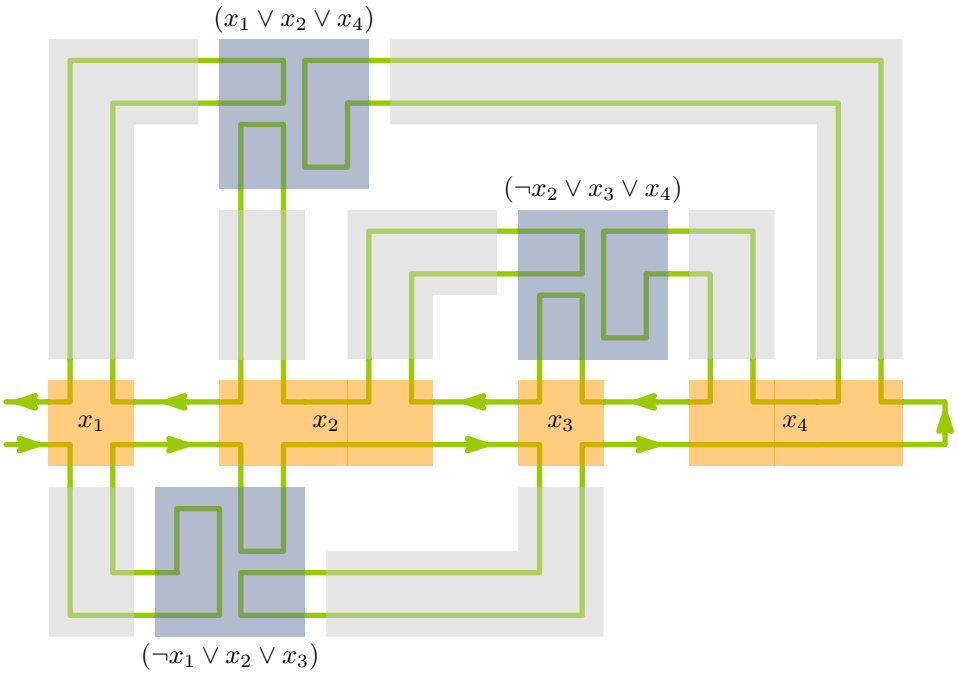


Figure 1 A rectilinear drawing of $(\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4) \wedge (\neg x_2 \vee x_3 \vee x_4)$. Variables (orange) are connected to their involved clauses (blue). The caterpillar will follow the green path.

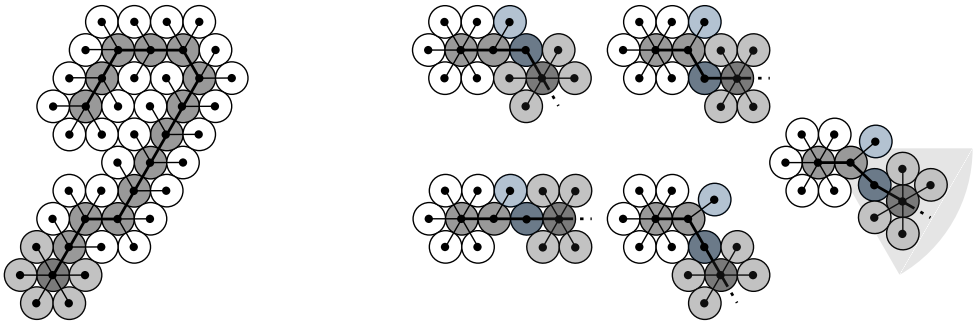
2 NP-hardness Reduction

To prove our NP-hardness result, we reduce from the NP-complete problem Planar 3-SAT. We first give an overview and then describe the gadgets in detail. Given a Planar 3-SAT instance ϕ with n variables and m clauses and its planar variable-clause graph $G(\phi)$, we construct an embedded caterpillar of size $O(m^2 + nm)$ that admits a weak UDCR if and only if ϕ is satisfiable. First, we will design a caterpillar C with a unique high-level realization that mimics the planar drawing of the variable-clause graph $G(\phi)$ of the Planar 3-SAT formula ϕ (see Figure 1). The unique realization can be obtained by locally optimal packings of leaf disks to enforce the required grid positioning of the backbone disks (see Figure 2a). We also call this a rigid construction. We then modify the subgraph of C near each variable in $G(\phi)$ such that we now have two possible local realizations corresponding to the true/false assignments in each variable gadget. The position of the realization will be propagated through the rigid components to the involved clause gadgets such that each clause gadget can be realized if and only if at least one of its literals is true. We obtain

▶ **Theorem 1.** *The problem of deciding whether a caterpillar with a given embedding admits a weak unit disk contact representation in the plane is* NP*-hard.*

2.1 Planar 3-SAT

Given a Boolean formula ϕ in 3-CNF with n variables, its corresponding variable-clause graph $G(\phi)$ has nodes for each variable x_i and each clause c_j of ϕ and there are edges between a variable x_i and a clause c_j iff either x_i or $\neg x_i$ appear in c_j. Furthermore there is an edge $\{x_i, x_{i+1}\}$ for all $1 \leq i < n$ plus $\{x_n, x_1\}$, i.e., a cycle through all variable nodes. The Planar 3-SAT problem is to decide, given a formula ϕ for which $G(\phi)$ is planar, whether ϕ



(a) By starting with an interior node with 5 leaves this is the graph's only realization (up to rotation).

(b) Five (of infinitely many) different realizations when allowing freedom after the second interior node. Nodes which can move around are marked. The third and fourth interior nodes together regain rigidity. The fourth disk stays in the marked $\pi/3$ sector.

Figure 2 An interior node with six neighbors enforces rigidity, even after allowing some freedom.

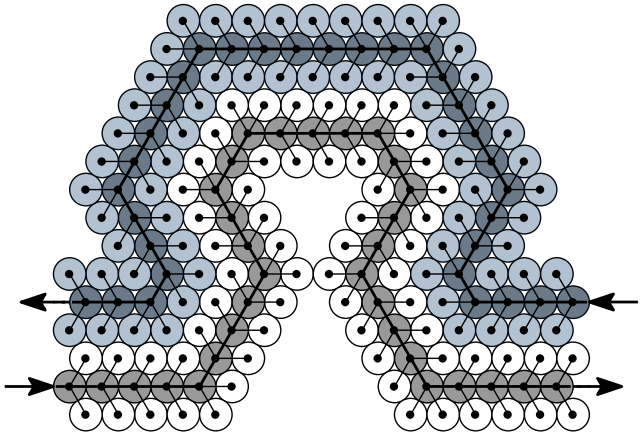
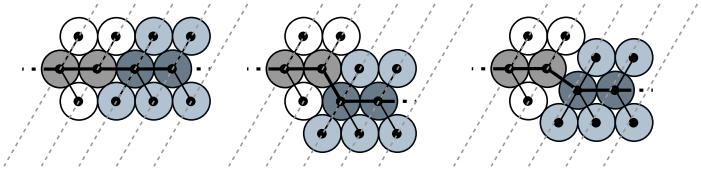


Figure 3 If the lower and upper part travel in the annotated direction and cannot move vertically apart by more than six disks, this is the only possible realization.

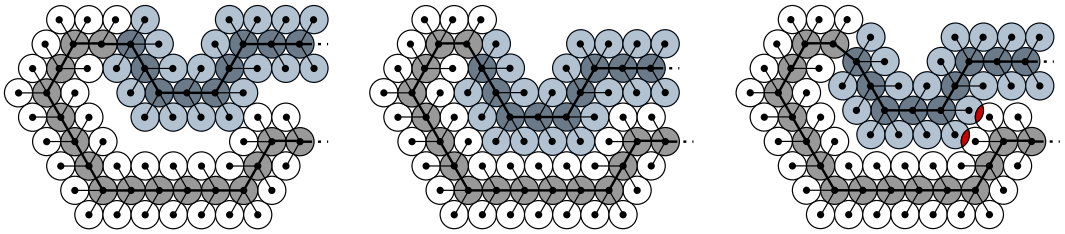
is satisfiable. Lichtenstein [8] showed that Planar 3-SAT is NP-complete. It is possible to arrange all variables on a horizontal line and to use only rectilinear connectors to connect the variables with the respective clauses in a comb-like fashion [6]. An example is shown in Figure 1 where we added a directed path indicating how the caterpillar traverses $G(\phi)$.

2.2 Rigidity – Allowing Exactly One Realization (up to Rotation)

We first observe that we can use a locally optimal packing of unit disks to enforce the direction in which the caterpillar continues. As observed in Figure 2a, starting with a node with five leaves fixes the position of the next backbone-disk. Since the next backbone-disk can have up to 3 more neighbors, adding two leaves to this node in a particular cyclic order again fixes the next backbone-disk's position. By repeatedly applying this construction, we can build a rigid 3-disk-wide path on a hexagonal grid. Furthermore, as shown in Figure 2b, even after allowing some freedom of movement, rigidity can be regained by an interior node with four leaves and thus six neighbors. Hence, it is possible to have two rigid components joined by a non-rigid part. Sometimes we would like to make sure that two parts of the



(a) Reparenting one leaf to the next interior node gives a restrained freedom of movement. All positions between the upper (left) and lower (center) position are possible (right).



(b) Adding a rigid structure which is aligned with the hexagonal grid removes the possibility to realize the intermediate positions (right). Only the two extremal positions (left and center) remain.

Figure 4 A construction which allows for exactly two different realizations.

caterpillar have a certain position relative to each other. This can be achieved by introducing a locking structure which is shown in Figure 3.

2.3 Variable Gadgets – Allowing Exactly Two Different Realizations

For the reduction we design a caterpillar and its embedding in such a way that there are exactly two local realizations to simulate truth values in each variable gadget. As shown in Figure 4a, flipping the connection of one leaf to the next interior node along the rigid path allows the latter path to shift between two positions where the line passing through the two positions forms an angle of 60 degrees relative to the direction of the backbone. However, intermediate positions are also possible which we want to prevent. Since the movement happens along a circular arc all intermediate positions might cause an intersection in other grid-aligned paths. By deliberately introducing such a path, as shown in Figure 4b, we can restrict this part of the graph to be realized in only two possible ways.

In Figure 5 we show the basic idea of the variable gadget. We assume to have a fixed inner structure (represented by the uncolored inner hexagon) to which six different paths are connected. Those paths are colored in alternating colors to distinguish them easily. The outer paths can be pushed in a counter-clockwise or clockwise fashion (which can be interpreted as $x = \text{true}$ or $x = \text{false}$) which moves exactly one disk in each of the six main directions. However, as before, intermediate positions are possible but they have to be avoided. By making the hexagon bigger, we can use a similar construction as before to only allow the two extremal positions in any realization of the caterpillar. The solution to this can be found in Figure 6 which focuses on just one corner with two adjacent paths (a sketch of the full hexagon can be seen in Figure 7). The gray part to the right is a corner of the inner hexagon and considered fixed. If the green path is pushed to one extremal position, the blue path has to follow so that no overlapping occurs. If the green path is in an intermediate position, the blue hook cannot align itself with the green path without intersection such that it is still touching the gray disk following the caterpillar.

All these ideas are combined to form a full variable gadget (see Figure 7). The caterpillar

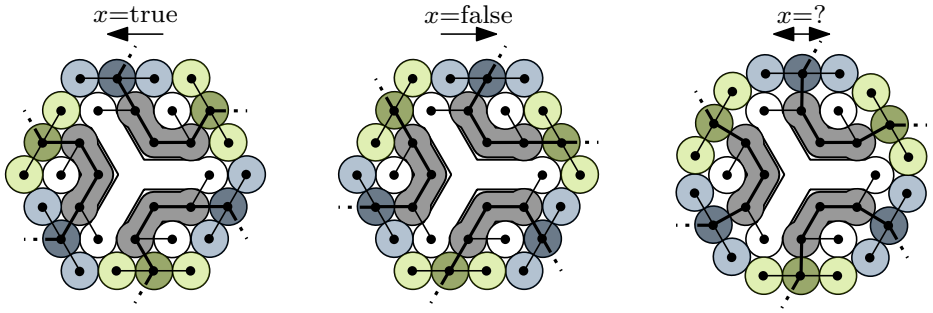


Figure 5 The variable gadget idea. Assume that the white-gray hexagon in the center is somehow fixed. Then, moving one of the colored parts in one direction forces the movement of all five others. Again, we have two extremal positions (left and center) but also all intermediate positions (right).

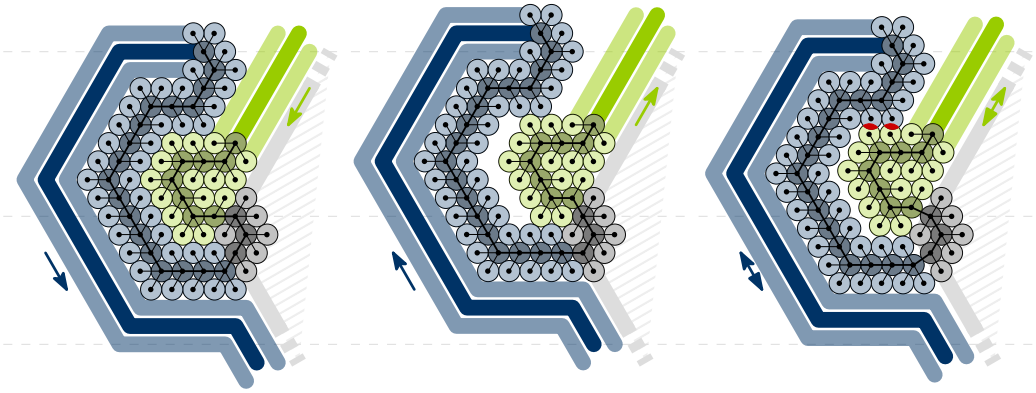


Figure 6 Introducing an interlocking structure at the corner of the variable gadget prohibits all but the two extremal positions (left and center). It also prevents any movement of the gray part.

path is assumed to be rigid when entering the gadget from the left. Each part with the same color is completely rigid and the transitions between two colors are as in Figure 4 so that we only have two possible local realizations. The path first traces the gray part on the lower left which is to prevent movement of the hexagons in the up-down-direction. Afterwards the construction from Figure 4b is used to allow exactly two positions for the following part. The lock from Figure 3 will make sure that the corresponding part on its way back will be together with the current part. The path moves counter-clockwise around the hexagon while using the construction from Figure 6 for the corners and simultaneously some interlocking path for the inner hexagon to make the interior completely rigid. When reaching the bottom part of the outer hexagon we extend to the left to align the vertical position of the variable with the outer structure. Then the path goes towards a clause and comes back to the same place—if no clause is connected here, we just connect the two parts directly.

We continue on the lower side of the construction into the next hexagon. If the first hexagon is pushed clockwise the second one is pushed counter-clockwise and vice-versa. This means, that every second clause connector is pushed left while every other second is pushed right. We finally finish the lower part of the construction of one variable gadget by reaching the gray part on the lower right. Here the path enters another variable gadget and eventually comes back to trace the upper part of the construction in the same fashion as the lower part. To have more than six clause connections shown here, we can just repeat the first and second hexagons arbitrarily often. Different variables are just chained to the right (cf. Figure 1).

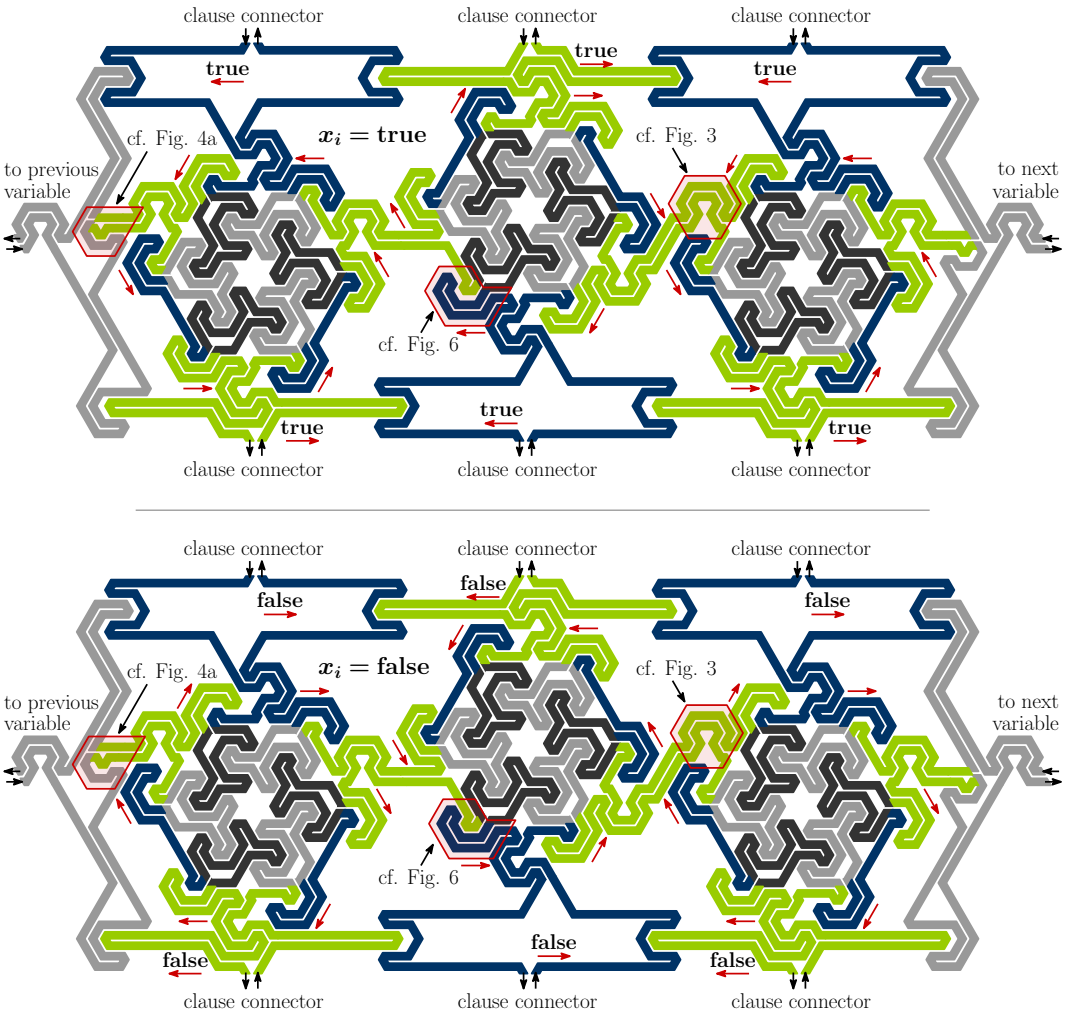


Figure 7 A simplified variable gadget depiction for x_i = true (top) and x_i = false (bottom). All six clause connectors are shifted by one disk to the left or right compared to the other state, indicated by the red arrows. Repeating the last two hexagons adds more connectors. Chaining the whole gadget gives arbitrarily many variables. Some appearances of previous figures are highlighted.

2.4 Clause Gadgets

We now have a variable gadget which moves a rigid sub-caterpillar between exactly two possible positions on the hexagonal grid, namely left and right. With this we want to construct a clause gadget which should be realizable if and only if at least one literal is set to true. The idea for the clause gadget is shown in Figure 8: We have one larger part coming from the right which has exactly one leaf missing and two smaller parts coming from the left, each of which has one leaf protruding to the right. The three parts should be connected to the corresponding variable gadgets such that a true value for the corresponding literal pulls them away from the center and a false value pushes them towards the center. As we can observe, if the right part is set to false there is room for at most one leaf of a left part but not for both. Thus, setting all literals to false makes it impossible to realize this caterpillar, while in all other cases a realization of the clause gadget exists.

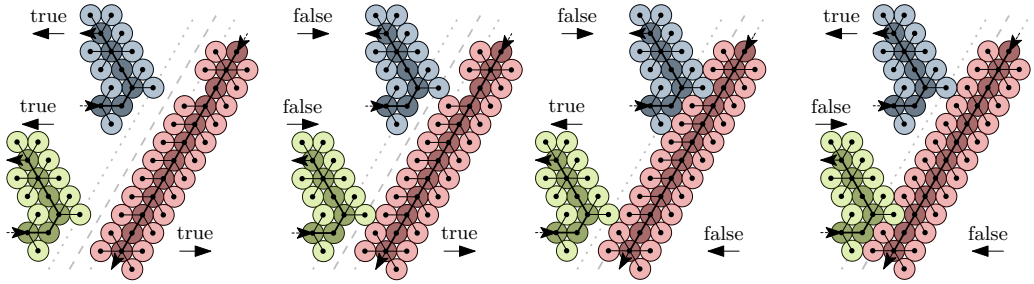


Figure 8 The idea of the clause gadget: The two literals on the left have one bulge each whereas the literal on the right has one notch which can accomodate either but not both bulges.

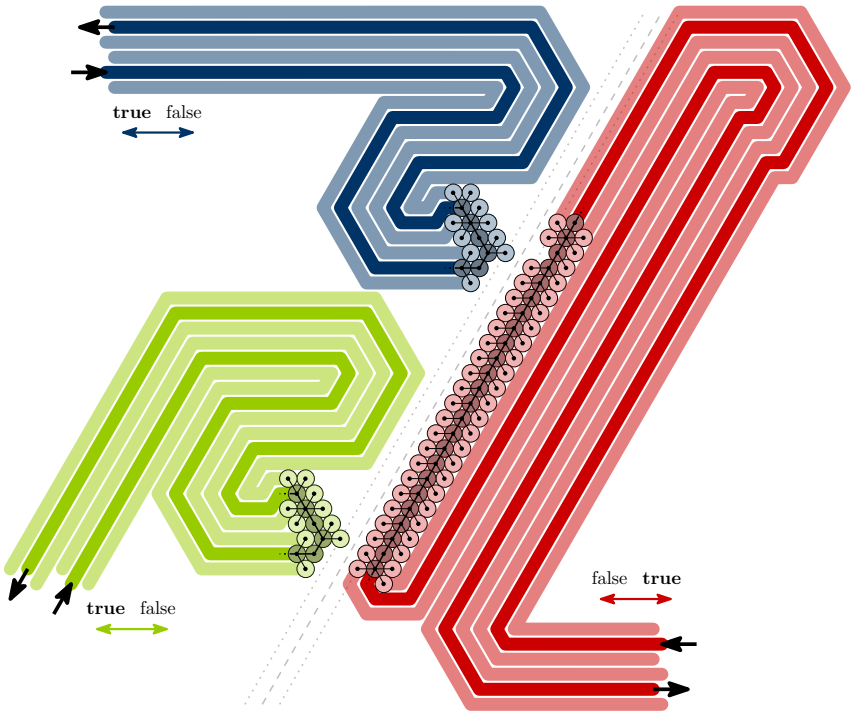


Figure 9 The full clause gadget.

Each of the three parts has some missing leaves and thus causes some freedom to move around. We need to make sure that, despite possible movement, they can be only realized the way we intend. The result is shown in Figure 9. The long right side of the clause will be realized as the first part—because of the one missing leaf it could be rotated by at most 60 degrees. By going all the way back with a small interlocking on the top this would lead to self-intersection and thus the shown realization is the only one (ignoring the leaves which could move up and down). The two paths on the left can only be realized as shown because they would otherwise intersect with the right side or with themselves. The clause gadget is connected like this on the upper side of the whole construction and rotated by 180 degrees on the lower side of the construction. We finally show a full picture of one possible realization of the abstract drawing of Figure 1 in Figure 10.

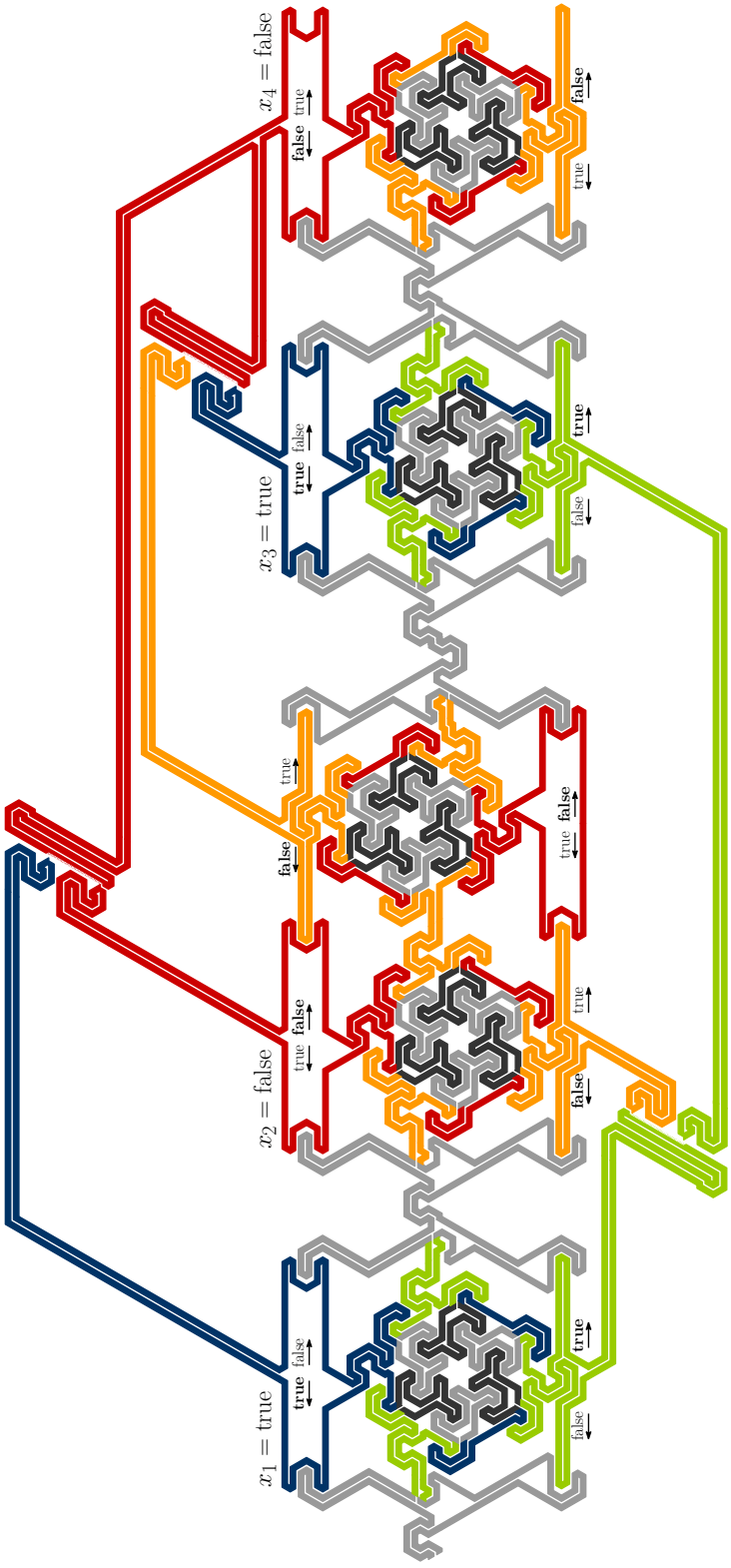


Figure 10 One possible realization of the formula from Figure 1 with x_1 = true, x_2 = false, x_3 = true, and x_4 = false. Due to space constraints the first hexagon of x_4 behaves different from the other variables. Otherwise a second hexagon would be needed.

2.5 Summary of the Reduction

Each variable gadget starts and end with a rigid part with constant size. Furthermore, each literal needs at most two hexagons (of constant size) in its corresponding variable gadget to have the correct connector. We have exactly $3m$ literals in ϕ and hence we need $O(n+m)$ many nodes for the variable gadgets. Each clause gadget has constant size and sits on its individual level. We can have at most m levels and each of the three connectors per clause has height and width of at most $O(m)$ and $O(n+m)$ respectively. Thus the clause gadgets with the connectors need $O(mn+m^2)$ many nodes which is also the size of the full construction.

Since the variable gadgets always start the same way, a variable is set to true if and only if the first hexagon is rotated counter-clockwise, false otherwise. Hence, by design of the gadgets above, a formula ϕ is satisfiable if and only if the corresponding polynomial-size caterpillar $C(\phi)$ can be recognized as a weak UDCR. This concludes the proof of Theorem 1.

Acknowledgments. This work was initiated during the *Japan-Austria Bilateral Seminar: Computational Geometry Seminar with Applications to Sensor Networks* in Zao Onsen, Japan in November 2018. We thank the organizers for providing a productive environment and the other participants, especially Oswin Aichholzer, André van Renssen, and Birgit Vogtenhuber, for the initial discussions.

References

1 Md. Jawaherul Alam, David Eppstein, Michael T. Goodrich, Stephen G. Kobourov, and Sergey Pupyrev. Balanced circle packings for planar graphs. In Christian Duncan and Antonios Symvonis, editors, *Graph Drawing (GD'14)*, volume 8871 of *LNCS*, pages 125–136. Springer Berlin Heidelberg. URL: `https://arxiv.org/abs/1408.4902`, `doi:10/gfvkfr`.

2 Clinton Bowen, Stephane Durocher, Maarten Löffler, Anika Rounds, André Schulz, and Csaba D. Tóth. Realization of simply connected polygonal linkages and recognition of unit disk contact trees. In Emilio Di Giacomo and Anna Lubiw, editors, *Graph Drawing and Network Visualization (GD'15)*, volume 9411 of *LNCS*, pages 447–459. Springer International Publishing. `doi:10/gfvkfp`.

3 Heinz Breu and David G. Kirkpatrick. Unit disk graph recognition is NP-hard. 9(1–2):3–24. `doi:10/dcr9m5`.

4 Man-Kwun Chiu, Maarten Löffler, Marcel Roeloffzen, and Ryuhei Uehara. A hexagon-shaped stable kissing unit disk tree. In Yifan Hu and Martin Nöllenburg, editors, *Graph Drawing and Network Visualization (GD'16)*, volume 9801 of *LNCS*, pages 628–630. Springer International Publishing.

5 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 (GD'15)*, volume 9411 of *LNCS*, pages 433–446. Springer International Publishing. URL: `http://arxiv.org/abs/1509.00720`, `doi:10/gfvkfq`.

6 Donald E. Knuth and Arvind Raghunathan. The problem of compatible representatives. 5(3):422–427. URL: `http://epubs.siam.org/doi/10.1137/0405033`, `doi:10/fqq93q`.

7 Paul Koebe. Kontaktprobleme der konformen Abbildung. 88:141–164.

8 David Lichtenstein. Planar formulae and their uses. 11(2):329–343. URL: `http://epubs.siam.org/doi/10.1137/0211025`, `doi:10/cgbttx`.

Simultaneous Representation of Proper and Unit Interval Graphs

Ignaz Rutter[1], Darren Strash[2], Peter Stumpf[1], and Michael Vollmer[3]

1 **Faculty of Computer Science and Mathematics, University of Passau, Germany**
{rutter,stumpf}@fim.uni-passau.de
2 **Department of Computer Science, Hamilton College, USA**
dstrash@hamilton.edu
3 **Department of Informatics, Karlsruhe Institute of Technology (KIT), Germany**
michael.vollmer@kit.edu

Abstract

A simultaneous representation of graphs $G_1, \ldots, G_k$ consists of a (geometric) intersection representation R_i for each graph G_i such that for each pair of graphs G_i and G_j the representations R_i and R_j are *compatible* in the sense that vertices shared by G_i and G_j are represented by the same geometric object in R_i and in R_j. An important special case is the *sunflower case*, where we require that $G_i \cap G_j$ yields the same *shared graph* S for each $i \neq j$. While the existence of simultaneous interval representations for $k = 2$ can be tested efficiently, testing it for non-sunflower graphs with k not fixed is NP-complete. We give efficient algorithms for testing the existence of simultaneous proper and unit interval representations for sunflower graphs with k not fixed.

1 Introduction

A fundamental problem in the area of intersection graphs is the *recognition* problem, where the task is to decide whether a given graph G admits a particular type of (geometric) intersection representation. The simultaneous representation problem is a generalization of the recognition problem which asks for a *simultaneous graph* $\mathcal{G} = (G_1, \ldots, G_k)$ whether it admits a simultaneous geometric representation $\mathcal{R} = (R_1, \ldots, R_k)$.

Simultaneous representations have first been studied in the context of graph embeddings where the goal is to embed each simultaneous graph without edge crossings while any shared vertices have the same coordinates in all embeddings; see [1] for a survey. The notion of simultaneous representation of general intersection graph classes was introduced by Jampani and Lubiw [9]. They gave an $O(n^2 \log n)$ recognition algorithm for simultaneous interval graphs with $k = 2$ [8]. Bläsius and Rutter later improved the running time to linear [2]. Bok and Jedličková very recently showed that recognizing simultaneous non-sunflower interval graphs with k not fixed is NP-complete [3]. The problem is open in the sunflower case.

Contribution. We settle these problems with k not fixed for simultaneous proper and unit interval graphs – those graphs with an interval representation where no interval properly contains another and where all intervals have unit length, respectively. For the sunflower case, we provide efficient recognition algorithms. The running time for proper interval graphs is linear, while for the unit case it is $\mathcal{O}(|V| \cdot |E|)$ where V and E are the set of vertices and edges in the union of the sunflower graphs, respectively. For the non-sunflower case, we prove NP-completeness. The reductions are similar to the simultaneous independent work of Bok and Jedličková for simultaneous interval graphs [3].

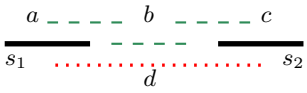


Figure 1 A simultaneous proper interval representation of a sunflower graph $\mathcal{G} = (P_5, P_3)$ without simultaneous unit interval representation (P_5 green dashed, P_3 red dotted, $P_5 \cap P_3$ black bold).

2 Preliminaries

All graphs in this paper are undirected. An *interval representation* $R = \{I_v \mid v \in V\}$ of a graph $G = (V, E)$ associates with each vertex $v \in V$ an interval $I_v = [x, y] \subset \mathbb{R}$ such that for each pair of vertices $u, v \in V$ we have $I_u \cap I_v \neq \emptyset \Leftrightarrow uv \in E$. An interval representation R is *proper* if no interval properly contains another one, and it is *unit* if all intervals have length 1. A graph is a *(proper/unit) interval graph* if and only if it admits a (proper/unit) interval representation. It is well-known that proper and unit interval graphs are the same graph class. However, the simultaneous unit interval graphs are a strict subclass of the simultaneous proper interval graphs; see Figure 1.

We use the well-known characterization of proper interval graphs using *straight enumerations* [6]. Two adjacent vertices $u, v \in V$ are *indistinguishable* if we have $N[u] = N[v]$ where $N[u] = \{v \colon uv \in E(H)\} \cup \{u\}$ is the closed neighborhood. Being indistinguishable is an equivalence relation and we call the equivalence classes *blocks* of G. Two blocks B, B' are *adjacent* if and only if $uv \in E$ for (any) $u \in B$ and $v \in B'$. A linear ordering σ of the blocks of G is a *straight enumeration* of G if for every block, the block and its adjacent blocks are consecutive in σ. A proper interval representation R defines a straight enumeration $\sigma(R)$ by ordering the intervals by their starting points and grouping together the blocks. Conversely, for each straight enumeration σ, there exists a corresponding representation R with $\sigma = \sigma(R)$ [6]. A *fine enumeration* of a graph H is a linear ordering η of $V(H)$ such that for $u \in V(H)$ the closed neighborhood $N[u]$ is consecutive in η.

▶ **Proposition 2.1** ([11, 6, 7]). *For a graph G the following statements are equivalent: (i) G is a proper interval graph, (ii) G has a straight enumeration, (iii) G has a fine enumeration. Also, for a connected proper interval graph its straight enumeration is unique up to reversal.*

In the following we only consider sunflower graphs $\mathcal{G} = (G_1, \ldots, G_k)$ with shared graph S. Note that it is necessary that S is an induced subgraph of each input graph G_i. Also note that $\mathcal{G}$ admits a simultaneous (proper/unit) interval representation if and only if each component of its union graph $\bigcup_{i=1}^{k} G_i$ does. We hence restrict our attention to sunflower graphs that are *connected* in the sense that their union graph is connected.

3 Sunflower Proper Interval Graphs

Let $\mathcal{G} = (G_1, \ldots, G_k)$ be a sunflower graph with shared graph $S = (V_S, E_S)$. By Proposition 2.1 each G_i has at least one fine enumeration. If there are fine enumerations $\sigma_1, \ldots, \sigma_k$ of $G_1, \ldots, G_k$ that coincide on V_S, then they induce a fine enumeration σ_S of S. We can then find a proper interval representation of S corresponding to σ_S that can be extended to proper interval representations of $G_1, \ldots, G_k$ in linear time [10]. Otherwise there is no simultaneous proper interval representation. Using PQ-trees [5, 4], the existence of such an ordering σ_S can be tested in linear time.

▶ **Theorem 3.1.** *Given a sunflower graph $\mathcal{G} = (G_1, \ldots, G_k)$, it can be tested in linear time whether $\mathcal{G}$ admits a simultaneous proper interval representation.*

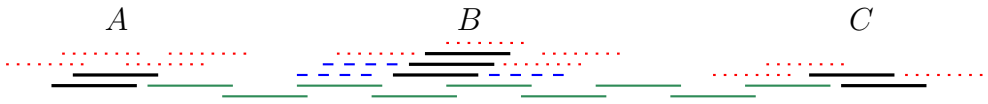


Figure 2 Simultaneous proper interval representation of G_1 (green solid), G_2 (red dotted), G_3 (blue dashed) with shared graph S (black bold). S has three blocks A, B, C. We denote the component of G_i containing a block D by C^i_D. C^2_A, C^2_B, C^3_B, C^2_C are loose. C^2_A is independent. (C^2_B, C^3_B) is a reversible part. (C^2_C) is not a reversible part, since C^1_C is aligned at C and not loose.

Next we characterize all simultaneous proper interval representations of a sunflower graph. Let $\mathcal{G} = (G_1, \ldots, G_k)$ be a sunflower graph with shared graph $S = (V_S, E_S)$ and for each $G_i \in \mathcal{G}$ let σ_i be a straight enumeration of G_i. We call the tuple $(\sigma_1, \ldots, \sigma_k)$ a *simultaneous enumeration* if for any $i, j \in \{1, \ldots, k\}$ and $u, v \in V_S$ the blocks $B_i(u)$, $B_i(v)$ and $B_j(u)$, $B_j(v)$ of G_i and G_j containing u, v are not ordered differently by σ_i and σ_j, i.e., we do not have $(B_i(u), B_i(v)) \in \sigma_i$ and $(B_j(v), B_j(u)) \in \sigma_j$ or vice versa.

▶ **Theorem 3.2.** *Let $\mathcal{G} = (G_1, \ldots, G_k)$ be a sunflower graph. There exists a simultaneous proper interval representation $\mathcal{R} = (R_1, \ldots, R_k)$ of $\mathcal{G}$ if and only if there is a simultaneous enumeration $(\sigma_1, \ldots, \sigma_k)$ of $\mathcal{G}$. If $(\sigma_1, \ldots, \sigma_k)$ exists, there also exists a simultaneous proper interval representation $\mathcal{R} = (R_1, \ldots, R_k)$ with $(\sigma(R_1), \ldots, \sigma(R_k)) = (\sigma_1, \ldots, \sigma_k)$.*

It turns out there is a unique straight enumeration of S induced by all simultaneous proper interval representations of $\mathcal{G}$ (up to reversal) if $\mathcal{G}$ is connected. For the following definitions see Figure 2. Let C be a component of a graph G in $\mathcal{G}$. We call C *loose* if all shared vertices in C are in the same block of S. Reversal of loose components is the only "degree of freedom" among simultaneous enumerations, besides full reversal. We say two vertices $u, v \in V_S$ *align* C if they are in different blocks of C. We call C *independent* if it is loose and not aligned by any two vertices of S.

We say C *is aligned at* a block B of S if it is aligned by two vertices u, v in B. Any two components aligned at the same block can not be reversed independently. For each block B of S, let $\mathcal{C}(B)$ be the connected components among graphs in $\mathcal{G}$ aligned at B. If all components in $\mathcal{C}(B)$ are loose, we call it a *reversible part*. Note that a reversible part contains at most one component of each graph G_i. Let $(\sigma_1, \ldots, \sigma_k)$ and $(\sigma'_1, \ldots, \sigma'_k)$ be tuples of straight enumerations of $G_1, \ldots, G_k$. We say $(\sigma'_1, \ldots, \sigma'_k)$ is *obtained from* $(\sigma_1, \ldots, \sigma_k)$ *by reversing* reversible part $\mathcal{C}(B)$ if $\sigma'_1, \ldots, \sigma'_k$ are obtained by reversal of all components in $\mathcal{C}(B)$. We characterize the simultaneous enumerations of $\mathcal{G}$ as follows.

▶ **Theorem 3.3.** *Let $\mathcal{G} = (G_1, \ldots, G_k)$ be a connected sunflower graph with simultaneous enumeration ρ. Then ρ' is a simultaneous enumeration of $\mathcal{G}$ if and only if ρ' can be obtained from ρ or its reversal ρ^r by reversing independent components and reversible parts.*

4 Sunflower Unit Interval Graphs

We now characterize for a sunflower graph $\mathcal{G} = (G_1, \ldots, G_k)$ with shared graph S the simultaneous enumerations $(\zeta_1, \ldots, \zeta_k)$ that can be *realized* by a simultaneous unit interval representation $(R_1, \ldots, R_k)$, in the sense that $\sigma(R_i) = \zeta_i$ for $i \in \{1, \ldots, k\}$. For $i \in \{1, \ldots, k\}$ let $(V_i, E_i) = G_i$. Let further $V = V_1 \cup \cdots \cup V_k$. For a straight enumeration η of some graph H we say for $u, v \in V(H)$ that $u <_\eta v$ if u is in a block before v, and we say $u \leq_\eta v$ if $u = v$ or $u <_\eta v$. We call $\leq_\eta$ the *partial order on $V(H)$ corresponding to η*. Note that for distinct u, v in the same block we have neither $u >_\eta v$ nor $u \leq_\eta v$. For convenience, we write $u \leq_i v$ and $u <_i v$ instead of $u \leq_{\zeta_i} v$ and $u <_{\zeta_i} v$, respectively.

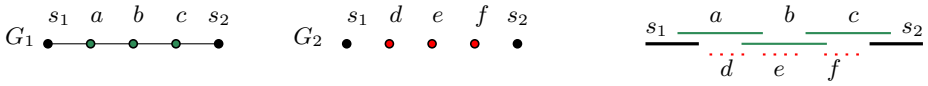


Figure 3 A sunflower graph $\mathcal{G} = (G_1, G_2)$ with shared vertices s_1, s_2. In the corresponding simultaneous enumeration ζ we have the (s_1, s_2)-chain $C = (s_1, a, b, c, s_2)$ and the (s_1, s_2)-bar $B = (s_1, d, e, f, s_2)$, both of size 5. Hence, $\mathcal{G}$ has conflict (C, B) for ζ.

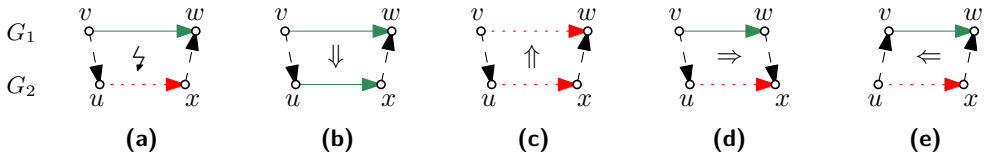


Figure 4 Two graphs G_1, G_2 with $V_1 = \{v, w\}$, $V_2 = \{u, x\}$, $u \leq_\eta x$, and $v \leq_\eta w$. In Figure 4a we have a forbidden configuration with (i) $vw \in E_1$, (ii) $ux \notin E_2$, (iii) $v \leq_\eta u$, and (iv) $x \leq_\eta w$. If three of these four conditions are met, we can conclude that the remaining one is false. Namely, in Figure 4b, 4c, 4d and 4e, we conclude $ux \in E_2$, $vw \notin E_1$, $w <_\eta x$, and $u <_\eta v$, respectively. We use arrows to represent a partial order between two vertices. We draw them green solid if they are adjacent, red dotted if they are non-adjacent in some graph G_i, and black dashed otherwise.

Let $u, v \in V_S$ with $u \neq v$. A *(u, v)-chain* of *size* m in (G_i, ζ_i) is a sequence $(u = c_1, \ldots, c_m = v)$ of vertices in V_i with $c_1 <_i \cdots <_i c_m$ that corresponds to a path in G_i. A *(u, v)-bar* between u and v of *size* m in (G_i, ζ_i) is a sequence $(u = b_1, \ldots, b_m = v)$ of vertices in V_i with $b_1 <_i \cdots <_i b_m$ that corresponds to an independent set in G_i; see Figure 3.

If there is a (u, v)-chain C in G_i of size $\ell \geq 2$ and a (u, v)-bar B in (G_j, ζ_j) of size at least ℓ, then we say that (C, B) is a *(chain-bar-)conflict* and that $\mathcal{G}$ *has conflict* (C, B) *for* ζ. Note that one can reduce the size of a (u, v)-bar by removing intervals between u, v. Thus, we can always assume that in a conflict, we have a bar and a chain of the same size $\ell \geq 2$.

Assume $\mathcal{G}$ has a simultaneous unit interval representation realizing ζ. If a graph $G \in \mathcal{G}$ has a (u, v)-chain of size $\ell \geq 2$, then I_u, I_v have a distance smaller than $\ell - 2$. On the other hand, if a graph $G \in \mathcal{G}$ has a (u, v)-bar of size ℓ, then I_u, I_v have a distance greater than $\ell - 2$. Hence, sunflower graph $\mathcal{G}$ has no conflict. The absence of conflicts is not only necessary, but also sufficient.

▶ **Theorem 4.1.** *A sunflower graph $\mathcal{G}$ with simultaneous enumeration ζ has a simultaneous unit interval representation that realizes ζ if and only if it has no conflict for ζ.*

Proof Sketch. Let $\alpha^\star$ be the union of the partial orders on $V_1, \ldots, V_k$ corresponding to $\zeta_1, \ldots, \zeta_k$. We set α to be the transitive closure of $\alpha^\star$, meaning α is the partial order on V induced by ζ. After identifying certain "indistinguishable" vertices of V, we can assume that α is a linear ordering on $V_1, \ldots, V_k$. Assuming there is no conflict, we then construct a simultaneous unit interval representation R. To this end, we first extend α to a linear ordering on V and thus of the interval starting points. Afterwards, we decide for every pair u, v of vertices from different graphs whether I_u, I_v intersect to obtain an order of the interval end points. Note that each partial order $\alpha_{|V_i}$ already is a fine enumeration of G_i.

All necessary extensions of α and decisions for adjacencies between vertices of different graphs arise from one *forbidden configuration*; see Figure 4a. We first go from right to left and extend α according to Figure 4e. In that run only necessary extensions are made. The key idea in that run is that the extensions of α correspond to extensions of pairs of chains and bars of equal size with a shared end to the right. If the forbidden configuration is obtained, then such a chain-bar pair also shares the second end and therefore yields a conflict. With

this preparation, we can then go from left to right and greedily extend α to a linear ordering τ that respects the implication of Figure 4e. As such τ avoids the forbidden configuration. We finally use τ to decide adjacency for every pair of vertices according to Figure 4b and thereby still avoiding the forbidden configuration. We obtain a graph H that has $G_1, \ldots, G_k$ as induced subgraphs and for which τ is a fine enumeration. By Proposition 2.1 H is a proper and thus a unit interval graph. A unit interval representation of H induces a simultaneous unit interval representation of $\mathcal{G} = (G_1, \ldots, G_k)$. ◀

We now give a recognition algorithm for sunflower unit interval graphs. By Theorem 3.1 we obtain a simultaneous enumeration ζ of $\mathcal{G}$, unless $\mathcal{G}$ is not even a simultaneous proper interval graph. By Theorem 4.1 we need to decide if $\mathcal{G}$ has a simultaneous enumeration η without conflicts. By Theorem 3.3, if it exists, η results from ζ by reversing reversible parts and independent components. We formulate this as a 2-SAT formula with a variable for each reversible part and for each independent component that encodes its orientation.

For each pair of shared vertices u, v we formulate clauses that exclude conflicts for u,v. The minimal (u, v)-chains for G_i are independent of reversals. The size of a largest (u, v)-bar in G_i only depends on the orientations of the connected components C and D containing u and v, respectively, while components in-between always contribute their maximum independent set regardless of whether they are reversed. For each of the at most four relevant combinations of orientations we check whether it produces a conflict. In that case we add a clause that forbids that combination (note that the orientations of C and D are determined by one reversible part or independent component each, if they are loose at all). The 2-SAT formula $\mathcal{F}$ contains these clauses for all shared vertex pairs and all graphs G_i. By construction $\mathcal{F}$ has a solution if and only if $\mathcal{G}$ is a simultaneous unit interval graph.

▶ **Theorem 4.2.** *Given a sunflower graph $\mathcal{G} = (G_1, \ldots, G_k)$, we can decide in $O(|V| \cdot |E|)$ time, whether $\mathcal{G}$ is a simultaneous unit interval graph, where $(V, E) = G_1 \cup \cdots \cup G_k$.*

References

1. T. Bläsius, S. G. Kobourov, and I. Rutter. Simultaneous embedding of planar graphs. *CoRR*, abs/1204.5853, 2012.
2. T. Bläsius and I. Rutter. Simultaneous PQ-ordering with applications to constrained embedding problems. *ACM Trans. Algorithms*, 12(2):16:1–16:46, 2015.
3. J. Bok and N. Jedličková. A note on simultaneous representation problem for interval and circular-arc graphs. *arXiv preprint arXiv:1811.04062*, 2018.
4. K. S. Booth. *PQ Tree Algorithms*. PhD thesis, University of California, Berkeley, 1975.
5. K. S. Booth and G. S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *Journal of Computer and System Sciences*, 13(3):335–379, 1976.
6. X. Deng, P. Hell, and J. Huang. Linear-time representation algorithms for proper circular-arc graphs and proper interval graphs. *SIAM J. Comput.*, 25(2):390–403, 1996.
7. P. Hell, R. Shamir, and R. Sharan. A fully dynamic algorithm for recognizing and representing proper interval graphs. *SIAM J. Comput.*, 31(1):289–305, 2002.
8. K. R. Jampani and A. Lubiw. Simultaneous interval graphs. In O. Cheong, K.-Y. Chwa, and K. Park, editors, *Algorithms and Computation: 21st International Symposium, ISAAC 2010, Jeju Island, Proceedings, Part I*, pages 206–217. Springer, 2010.
9. K. R. Jampani and A. Lubiw. The simultaneous representation problem for chordal, comparability and permutation graphs. *Journal of Graph Algorithms and Applications*, 16(2):283–315, 2012.

10 P. Klavík, J. Kratochvíl, Y. Otachi, I. Rutter, T. Saitoh, M. Saumell, and T. Vyskočil. Extending partial representations of proper and unit interval graphs. *Algorithmica*, 77(4):1071–1104, Apr 2017.

11 F. S. Roberts. *Representations of indifference relations*. PhD thesis, Department of Mathematics, Stanford University, 1968.

Packing Disks into Disks with Optimal Worst-Case Density

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

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

Abstract

We provide a tight result for a fundamental problem arising from packing disks into a circular container: The *critical density* of packing disks in a disk is 1/2. This implies that any set of (not necessarily equal) disks of total area $\delta \leq 1/2$ can always be packed into a disk of area 1; on the other hand, for any $\varepsilon > 0$ there are sets of disks of area $1/2 + \varepsilon$ that cannot be packed. 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.

An longer version will appear in the 35th Symposium on Computational Geometry [3].

1 Introduction

Deciding whether a set of disks can be packed into a given container is a fundamental geometric optimization problem that has attracted considerable attention; see below for references. Disk packing also has numerous applications in engineering, science, operational research and everyday life, e.g., for the design of digital modulation schemes [19], packaging cylinders [1, 8], bundling tubes or cables [24, 22], the cutting industry [23], or the layout of control panels [1], or radio tower placement [23]. Further applications stem from chemistry [25], foresting [23], and origami design [13].

Like many other packing problems, disk packing is typically quite difficult; what is more, the combinatorial hardness is compounded by the geometric complications of dealing with irrational coordinates that arise when packing circular objects. This is reflected by the limitations of provably optimal results for the optimal value for the smallest sufficient disk container (and hence, the densest such disk packing in a disk container), a problem that was discussed by Kraviz [12] in 1967: Even when the input consists of just 13 unit disks, the optimal value for the densest disk-in-disk packing was only established in 2003 [7], while the optimal value for 14 unit disks is still unproven. The enormous challenges of establishing densest disk packings are also illustrated by a long-standing open conjecture by Erdős and Oler from 1961 [18] regarding optimal packings of n unit disks into an equilateral triangle, which has only been proven up to $n = 15$. For other examples of mathematical work on densely packing relatively small numbers of identical disks, see [9, 15, 5, 6], and [20, 14, 10] for related experimental work. Many authors have considered heuristics for circle packing problems, see [23, 11] for overviews of numerous heuristics and optimization methods. The best known solutions for packing equal disks into squares, triangles and other shapes are continuously published on Specht's website `http://packomania.com` [21].

For deciding whether a set of not necessarily equal disks can be packed into a square container, Demaine, Fekete, and Lang in 2010 [2] gave a proof of NP-hardness by using a reduction from 3-Partition, so we cannot expect that there is a deterministic polynomial-time algorithm for this problem.

The related problem of packing square objects has also been studied for a long time. Already in 1967, Moon and Moser [16] proved that it is possible to pack a set of squares

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 19–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

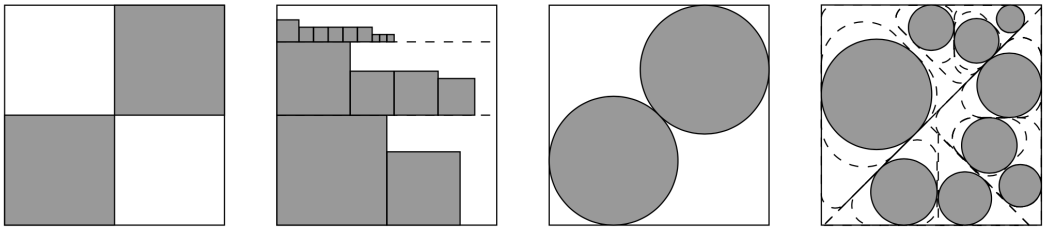


Figure 1 (1) An instance of critical density for packing squares into a square. (2) An example packing produced by Moon and Moser's shelf-packing. (3) An instance of critical density for packing disks into a square. (4) An example packing produced by Morr's Split Packing.

into the unit square in a shelf-like manner if their combined area, the sum of all squares' areas, does not exceed $\frac{1}{2}$. At the same time, $\frac{1}{2}$ is the *largest upper area bound* one can hope for, because two squares larger than the quarter-squares shown in Fig. 1 cannot be packed. We call the ratio between the largest combined object area that can always be packed and the area of the container the problem's *critical density*, or *worst-case density*. The equivalent problem of establishing the critical packing density for disks in a square was posed by Demaine, Fekete, and Lang [2] and resolved by Morr, Fekete and Scheffer [17, 4]. Making use of a recursive procedure for cutting the container into triangular pieces, they proved that the critical packing density of disks in a square is $\frac{\pi}{3+2\sqrt{2}} \approx 0.539$. It is quite natural to consider the analogous question of establishing the critical packing density for disks in a disk. However, the shelf-packing approach of Moon and Moser [16] uses the fact that rectangular shapes of the packed objects fit well into parallel shelves, which is not the case for disks; on the other hand, the split packing method of Morr et al. [17, 4] relies on recursively splitting triangular containers, so it does not work for a circular container that cannot be partitioned into smaller circular pieces.

1.1 Results

We prove that the critical density for packing disks into a disk is 1/2: Any set of not necessarily equal disks with a combined area of not more than half the area of a circular container can be packed; this is best possibly, as for any $\varepsilon > 0$ there are instances of total area $1/2 + \varepsilon$ that cannot be packed. See Fig. 2 for the critical configuration.

Our proofs are constructive, so they can also be used as a constant-factor approximation algorithm for the smallest-area container of a given shape in which a given set of disks can be packed. Due to the higher geometric difficulty of fitting together circular objects, the involved methods are considerably more complex than those for square containers. We make up for this difficulty by developing more intricate recursive arguments, including appropriate and powerful tools based on *interval arithmetic*.

2 A Worst-Case Optimal Algorithm

▶ **Theorem 1.** *Every set of disks with total area $\frac{\pi}{2}$ can be packed into the unit disk O with radius 1. For any $\varepsilon > 0$, there is a set of disks with total area $\frac{\pi}{2} + \varepsilon$ that cannot be packed into O. In other words, the* worst-case packing density *for packing disks into a disk is $\frac{1}{2}$.*

The worst case consists of two disks D_1, D_2 with radius $\frac{1}{2}$, see Fig. 2. Increasing the area of D_1 by ε yields a set of disks which cannot be packed. The total area of these two disks is

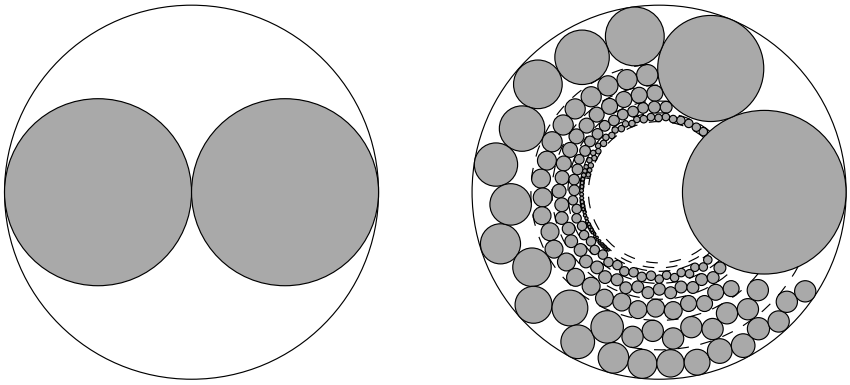


Figure 2 (1) A critical instance that allows a packing density no better than $\frac{1}{2}$. (2) An example packing produced by our algorithm.

$\frac{\pi}{4} + \frac{\pi}{4} = \frac{\pi}{2}$.

In the remainder of Section 2, we give a constructive proof for Theorem 1. Before we proceed to describe our algorithm in Section 2.4, we give some definitions and describe *Disk Packing* and *Ring Packing* as two subroutines of our algorithm.

2.1 Preliminaries for the Algorithm

We make use of the following definitions, see Fig. 3.

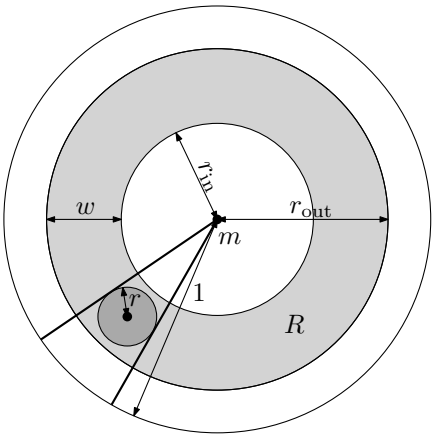


Figure 3 A ring $R \subset O$ with width w and a disk with its corresponding tangents.

For $r_{\text{out}} > r_{\text{in}} \geq 0$ and a container disk $\mathcal{C}$ such that $r_{\text{out}} \leq 2r_{\text{in}}$, we define a *ring* $R := R[r_{\text{out}}, r_{\text{in}}]$ of $\mathcal{C}$ as the closure of $r_{\text{out}} \setminus r_{\text{in}}$, see Fig. 3. If $r_{\text{in}} > 0$, the boundary of R consists of two connected components. The *inner boundary* is the component that lies closer to the center m of $\mathcal{C}$ and the *outer boundary* is the other component. The *inner radius* and the *outer radius* of R are the radius of the inner boundary and the radius of outer boundary. Each ring considered by our algorithm has one of three states $\{\text{OPEN}, \text{CLOSED}, \text{FULL}\}$. Initially, after its construction by the algorithm, each ring is OPEN.

Let r be a disk inside a container disk $\mathcal{C}$. The two *tangents* of r are the two rays starting

in the center of $\mathcal{C}$ and touching the boundary of r. We say that a disk lies *adjacent* to r_{out} when the disk is touching the boundary of r_{out} from the inside of r_{out}.

2.2 Disk Packing: A Subroutine

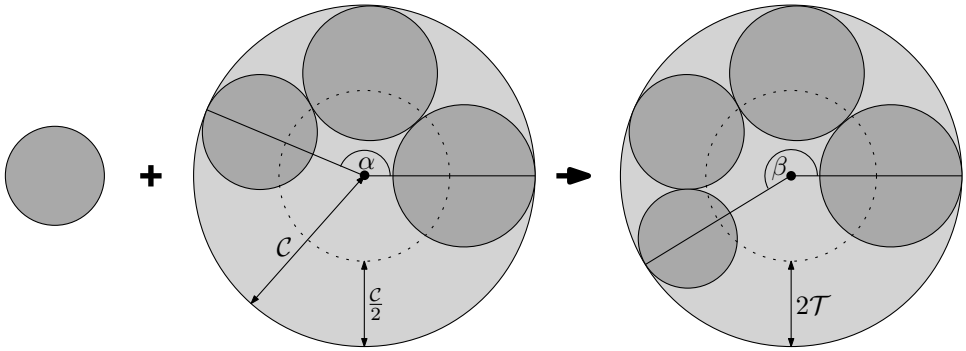


Figure 4 Disk Packing places disks in decreasing order of radius into a container $\mathcal{C}$ adjacent to the boundary of $\mathcal{C}$.

Consider a container disk $\mathcal{C}$, a set S of already packed disks that overlap with $\mathcal{C}$, but are not necessarily contained in it, and another disk r_i to be packed; see Fig. 4. We *pack* r_i into $\mathcal{C}$ adjacently to the boundary of $\mathcal{C}$ as follows: Let α be the maximal polar angle realized by the center of any disk from S. We choose the center of r_i such that it realizes the smallest possible polar angle $\beta \geq \alpha$ such that r_i touches the outer boundary of $\mathcal{C}$ from the interior of $\mathcal{C}$ without overlapping another disk from S, see Fig. 4. If r_i cannot be packed into $\mathcal{C}$, we say that r_i *does not fit into* R.

Let $0 < \mathcal{T} \leq \frac{1}{4}$, called *threshold*. *Disk Packing* considers the disks in decreasing order of radius and packs each disk r_i adjacent to the previous disk r_{i-1} and the boundary of $\mathcal{C}$ until r_i does not fit into $\mathcal{C}$ or $r_i < \mathcal{T}$.

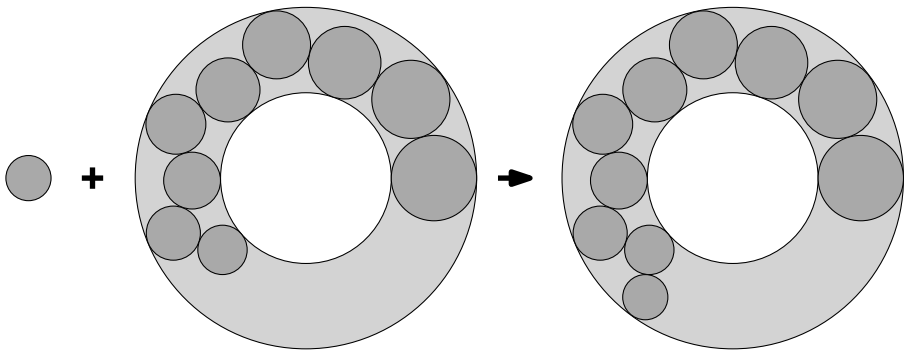


Figure 5 Ring Packing packs disks into a ring $R[r_{\text{out}}, r_{\text{in}}]$, alternating adjacent to the outer and to the inner boundary of R.

2.3 Ring Packing: A Subroutine

Consider a ring $R := R[r_{\text{out}}, r_{\text{in}}]$ with inner radius r_{in} and outer radius r_{out}, a (possibly empty) set S of already packed disks that overlap with R, and another disk r_i to be packed, see Fig. 5. We *pack* r_i into R adjacent to the outer (inner) boundary of R as follows: Let α be the maximal polar angle realized by a midpoint of a disk from S. We choose the midpoint of r_i realizing a smallest possible polar angle $\beta \geq \alpha$ such that r_i touches the outer (inner) boundary of R from the interior of R without overlapping another disk from S. If r_i cannot be packed into R, we say that r_i *does not fit into* R *(adjacent to the outer (inner) boundary)*.

Ring Packing iteratively packs disks into R alternating adjacent to the inner and outer boundary. If the current disk r_i does not fit into R, Ring Packing stops and we declare R to be FULL. If r_{i-1} and r_i could pass each other in R, i.e., the sum of the diameters of r_{i-1} and r_i are smaller than the width of R, Ring Packing stops and we declare R to be CLOSED.

2.4 Description of the Algorithm

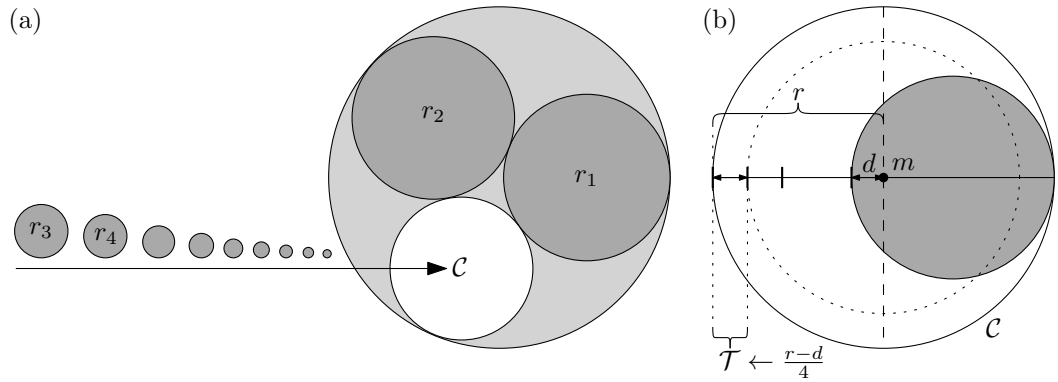


Figure 6 (a): If $r_1, r_2 \geq 0.495\mathcal{C}$, Disk Packing packs r_1, r_2 into $\mathcal{C}$. We update the current container disk $\mathcal{C}$ as the largest disk that fits into $\mathcal{C}$ and recurse on $\mathcal{C}$ with $r_3, \ldots, r_n$. (b): Determining the threshold $\mathcal{T}$ for disks packed by Disk Packing.

Our algorithm *creates* rings. A ring only exists after it is created. We stop packing at any point in time when all disks are packed. Furthermore, we store the current threshold $\mathcal{T}$ for Disk Packing and the smallest inner radius $r_{\min}$ of a ring created during the entire run of our algorithm. Initially, we set $\mathcal{T} \leftarrow 1, r_{\min} \leftarrow 1$. Our algorithm works in five phases:

- **Phase 1 - Recursion:** If $r_1, r_2 \geq 0.495\mathcal{C}$, apply Disk Packing to r_1, r_2, update $\mathcal{C}$ as the largest disk that fits into $\mathcal{C}$ and $\mathcal{T}$ as the radius of $\mathcal{C}$, and recurse on $\mathcal{C}$, see Fig. 6(a).
- **Phase 2 - Disk Packing:** Let r be the radius of $\mathcal{C}$. If the midpoint m of $\mathcal{C}$ lies inside a packed disk r_i, let d be the minimal distance of m to the boundary of r_i, see Fig. 6(b). Otherwise, we set $d = 0$.
  We apply Disk Packing to the container disk $\mathcal{C}$ with the threshold $\mathcal{T} \leftarrow \frac{r-d}{4}$.
- **Phase 3 - Ring Packing:** We apply Ring Packing to the ring $R := R[r_{\text{out}}, r_{\text{in}}]$ determined as follows: Let r_i be the largest disk not yet packed. If there is no open ring inside $\mathcal{C}$, we create a new open ring $R[r_{\text{out}}, r_{\text{in}}] \leftarrow R[r_{\min}, r_{\min} - 2r_i]$. Else, let $R[r_{\text{out}}, r_{\text{in}}]$ be the open ring with the largest inner radius r_{in}.
- **Phase 4 - Managing Rings:** Let $R[r_{\text{out}}, r_{\text{in}}]$ be the ring filled in Phase 3. We declare $R[r_{\text{out}}, r_{\text{in}}]$ to be closed and proceed as follows: Let r_i be the largest disk not yet packed.

If r_i and r_{i+1} can pass one another inside $R[r_{\text{out}}, r_{\text{in}}]$, i.e., if $2r_i + 2r_{i+1} \leq r_{\text{out}} - r_{\text{in}}$, we create two new open rings $R[r_{\text{out}}, r_{\text{out}} - 2r_i]$ and $R[r_{\text{out}} - 2r_i, r_{\text{in}}]$.

- **Phase 5 - Continue:** If there is an open ring, we go to Phase 3. Otherwise, we set $\mathcal{C}$ as the largest disk not covered by created rings, set $\mathcal{T}$ as the radius of $\mathcal{C}$, and go to Phase 2.

3 Analysis of the Algorithm

The analysis uses an intricate combination of manual analysis and an automated analysis based on interval arithmetic. For lack of space, details are omitted. See the appendix for full details.

4 Hardness

It is straightforward to see that the hardness proof for packing disks into a square can be adapted to packing disks into a disk, as follows.

▶ **Theorem 2.** *It is NP-hard to decide whether a given set of disks fits into a circular container.*

The proof is completely analogous to the one by Demaine, Fekete, and Lang in 2010 [2], who used a reduction from 3-Partition. Their proof constructs a disk instance which first forces some symmetrical free "pockets" in the resulting disk packing. The instance's remaining disks can then be packed into these pockets if and only if the related 3-Partition instance has a solution. Similar to their construction, we construct a symmetric triangular pocket by using a set of three identical disks of radius $\frac{\sqrt{3}}{2+\sqrt{3}}$ that can only be packed into a unit disk by touching each other. Analogous to [2], this is further subdivided into a sufficiently large set of identical pockets. The remaining disks encode a 3-Partition instance that can be solved if and only if the disks can be partitioned into triples of disks that fit into these pockets.

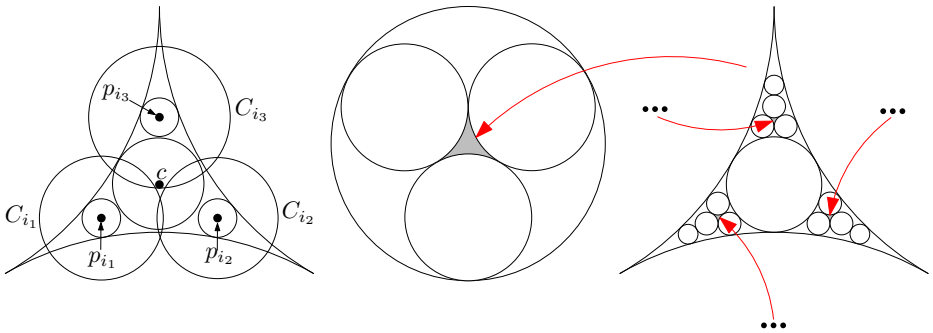


Figure 7 Elements of the hardness proof: (1) A symmetric triangular pocket from [2], allowing three disks with centers p_{i_1}, p_{i_2}, p_{i_3} to be packed if and only if the sum of the three corresponding numbers from the 3-Partition instance is small enough. (2) Creating a symmetric triangular pocket in the center by packing three disks of radius $\frac{\sqrt{3}}{2+\sqrt{3}}$ and the adapted argument from [2] for creating a sufficiently large set of symmetric triangular pockets.

5 Conclusions

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

References

1 I. Castillo, F. J. Kampas, and J. D. Pintér. Solving circle packing problems by global optimization: numerical results and industrial applications. *European Journal of Operational Research*, 191(3):786–802, 2008.

2 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.

3 S. P. Fekete, P. Keldenich, and C. Scheffer. Packing disks into disks with optimal worst-case density. In *Proceedings of the 35th Symposium on Computational Geometry*, 2019. To appear.

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

5 F. Fodor. The densest packing of 19 congruent circles in a circle. *Geometriae Dedicata*, 74:139–145, 1999.

6 F. Fodor. The densest packing of 12 congruent circles in a circle. *Beiträge zur Algebra und Geometrie (Contributions to Algebra and Geometry)*, 41:401–409, 2000.

7 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.

8 H. J. Fraser and J. A. George. Integrated container loading software for pulp and paper industry. *European Journal of Operational Research*, 77(3):466–474, 1994.

9 M. Goldberg. Packing of 14, 16, 17 and 20 circles in a circle. *Mathematics Magazine*, 44:134–139, 1971.

10 R. Graham, B. Lubachevsky, K. Nurmela, and P. Östergøard. Dense packings of congruent circles in a circle. *Discrete Mathematics*, 181:139–154, 1998.

11 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.

12 S. Kravitz. Packing cylinders into cylindrical containers. *Mathematics Magazine*, 40:65–71, 1967.

13 R. J. Lang. A computational algorithm for origami design. *Proceedings of the Twelfth Annual Symposium on Computational Geometry (SoCG)*, pages 98–105, 1996.

14 B. Lubachevsky and R. Graham. Curved hexagonal packings of equal disks in a circle. *Discrete & Computational Geometry*, 18:179–194, 1997.

15 H. Melissen. Densest packing of eleven congruent circles in a circle. *Geometriae Dedicata*, 50:15–25, 1994.

16 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.

17 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.

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

19 R. Peikert, D. Würtz, M. Monagan, and C. de Groot. Packing circles in a square: A review and new results. In *Proceedings of the 15th IFIP Conference*, pages 45–54, 1992.

20 G. Reis. Dense packing of equal circles within a circle. *Mathematics Magazine*, issue 48:33–37, 1975.

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

22 K. Sugihara, M. Sawai, H. Sano, D.-S. Kim, and D. Kim. Disk packing for the estimation of the size of a wire bundle. *Japan Journal of Industrial and Applied Mathematics*, 21(3):259–278, 2004.

23 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.

24 H. Wang, W. Huang, Q. Zhang, and D. Xu. An improved algorithm for the packing of unequal circles within a larger containing circle. *European Journal of Operational Research*, 141(2):440–453, sep 2002.

25 D. Würtz, M. Monagan, and R. Peikert. The history of packing circles in a square. *Maple Technical Newsletter*, page 35–42, 1994.

Dynamic Disk Connectivity*

Alexander Kauer[1] and Wolfgang Mulzer[1]

1 **Institut für Informatik, Freie Universität Berlin, Berlin, Germany**
[akauer, mulzer]@inf.fu-berlin.de

Abstract

Let $0 < \varphi \leq 1$ be a parameter. We present a data structure for maintaining the connected components of the intersection graph of a set of n disks with radii in $[\varphi, 1]$. The data structure allows inserting or deleting a disk in $\mathcal{O}(\frac{1}{\varphi} 2^{\alpha(n)} \log^{10} n)$ amortized expected time and querying two disks for connectivity in $\mathcal{O}(\log n)$ amortized time, where $\alpha(n)$ is the inverse Ackermann function. It requires $\mathcal{O}(\frac{1}{\varphi} n \log^3 n)$ expected space.

1 Introduction

Determining the connectivity in graphs is a fundamental algorithmic problem. The dynamic version where edges can be inserted or deleted is reasonably well understood [3–5, 9, 11], with data structures that support updates and queries for the connectivity of two vertices in polylogarithmic time. However, the case of vertex updates seems significantly harder, as a single update can have a larger impact. Chan et al. [2, Theorem 1] presented a data structure allowing vertex updates in $\widetilde{\mathcal{O}}(m^{2/3})$ amortized time and queries in $\widetilde{\mathcal{O}}(m^{1/3})$ time, where m is the number of possible edges of the graph that need to be known in advance.

A special case is dynamic connectivity of geometric intersection graphs. The vertices of such graphs are geometric objects of a certain type and two vertices share an edge if and only if the respective objects intersect. Connectivity queries now model reachability queries in sensor/IoT networks, road networks, and similar geometrically defined networks. Due to the restricted nature of the possible graphs, faster solutions may now be within reach. On the other hand, we must perform additional work to find the edges affected by an update.

Chan et al. [2, Theorem 5] also gave a general method for various geometric objects with slightly sub-linear update times and sub-linear query times. For disks with radii in $[\varphi, 1]$, $0 < \varphi \leq 1$, Kaplan et al. [7] described a faster data structure with $\mathcal{O}((\frac{1}{\varphi})^2 2^{\alpha(n)} \log^{10} n)$ amortized expected update time and $\mathcal{O}(\frac{\log n}{\log \log n})$ worst case query time; see Seiferth's thesis for details [10, Theorem 3.11]. Here, we show how to improve the dependence on $\frac{1}{\varphi}$ to linear.

2 Basic Composition of the Data Structure

The data structure by Kaplan et al. [7] relies on a dynamic graph connectivity structure for edge updates combined with appropriate rebuilding for changing vertex counts.

▶ **Theorem 2.1** (Holm et al. [5, Theorem 3]). *Let G be a graph with n vertices. There is a data structure such that inserting or deleting an edge in G take amortized time $\mathcal{O}(\log^2 n)$ and a connectivity query takes worst case time $\mathcal{O}(\frac{\log n}{\log \log n})$. This data structure requires $\mathcal{O}(m + n \log n)$ space, where m is the largest edge count at any given time.*

To avoid too many edge updates in the data structure of Holm et al. and to allow faster retrieval of intersecting disks, the intersection graph is not maintained explicitly, but is

* Partially supported by ERC STG 757609, GIF grant 1367/2015, and DFG grant MU 3501/2-2.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

represented by a grid with cell diameter φ. All disks with center in a single cell intersect, so we can merge them for connectivity. We use the cells as vertices and insert an edge between two of them if they contain intersecting disks. Every cell's disks can intersect only disks of $\mathcal{O}((\frac{1}{\varphi})^2)$ other cells, called the *neighborhood*. This limits the number of edge updates.

When inserting or deleting a disk, we must determine for which cells in the neighborhood we need to insert or delete edges. For this, we maintain for each pair of neighboring cells a *maximal bichromatic matching* (MBM) of intersecting disks between the cells using a data structure by Kaplan et al. requiring $\mathcal{O}(2^{\alpha(n)} \log^{10} n)$ expected amortized time for updates and $\mathcal{O}(n \log^3 n)$ expected space [7, Lemma 9.9 in the full version]. Altogether, this results in Kaplan et al.'s main result for dynamic disk connectivity:

▶ **Theorem 2.2** (Data Structure for Disk Graphs [10, Theorem 3.11]). *Let $0 < \varphi \leq 1$, and let P be a set of disks with radius in $[\varphi, 1]$. There is a dynamic connectivity structure for the intersection graph of P such that the insertion or deletion of a disk takes amortized expected time $\mathcal{O}((\frac{1}{\varphi})^2 2^{\alpha(n)} \log^{10} n)$ and a connectivity query takes worst case time $\mathcal{O}(\frac{\log n}{\log \log n})$, where n is the maximum number of inserted disks at any time and $\alpha(n)$ is the inverse Ackermann function. The data structure requires $\mathcal{O}((\frac{1}{\varphi})^2 n \log^3 n)$ expected space.*

A limitation of the approach behind Theorem 2.2 is that all disks are handled identically, irrespective of their actual size. To address this, we will use a hierarchy of grids $(\mathcal{G}_i)_{0 \leq i \leq \lceil \log \frac{1}{\varphi} \rceil}$ instead of a single grid, where $\mathcal{G}_i$ has cells of diameter $\frac{1}{2^i}$, for $i = 0, \ldots, \lceil \log \frac{1}{\varphi} \rceil$. Each grid with cells of diameter d then stores all disks with radius in $[d, 2d)$.

We represent the hierarchy of grids with quadtrees [1], where every cell $\mathcal{G}_0$ constitutes the root of one quadtree. Each disks s is stored in the quadtree cell that contains s's center and whose diameter is comparable to the radius of s, as explained above. The quadtrees have height at most $\lceil \log \frac{1}{\varphi} \rceil$ and their nodes are created upon the first access.

As before, two intersecting disks must be contained in two neighboring cells. Each neighboring cell σ of a given cell τ is either a child of a larger neighboring cell of τ, a child of τ itself, or a cell in $\mathcal{G}_0$. See Figure 1 for an example. A cell has of at most $13^2 \in \mathcal{O}(1)$ neighbors on its own level and on each level above, whereas in the levels below the number grows exponentially. Altogether, the size of a neighborhood is at most

$$\mathcal{O}(1) \cdot \sum_{i=0}^{\lceil \log \frac{1}{\varphi} \rceil} (2^i)^2 \in \mathcal{O}\Big(\big(\tfrac{1}{\varphi}\big)^2\Big). \tag{1}$$

▶ **Lemma 2.3.** *The data structure based on quadtrees has the same asymptotic update, query, and space bounds as the one from Theorem 2.2.*

Proof. When inserting or deleting a disk with radius r we need to obtain its cell $\sigma \in \mathcal{G}_i$ with $\frac{1}{2^i} \leq r < \frac{1}{2^{i-1}}$ and its neighborhood. We already observed that all neighboring cells are either a child of another neighboring cell, a child of σ, or in $\mathcal{G}_0$. Thus, these can be obtained via recursing down one or multiple quadtrees, requiring only constant time per neighbor. For each of the $\mathcal{O}((\frac{1}{\varphi})^2)$ neighboring cells we then need to update the MBM and may need to update the connectivity structure, as before. ◀

3 Different Handling of a Disk's Boundary and Inner Area

3.1 Limit the Insertions into MBMs

The quadratic dependency on $\frac{1}{\varphi}$ during updates in Lemma 2.3 can be approached using two observations: First, when a cell's disks are contained completely in an inserted disk, we do

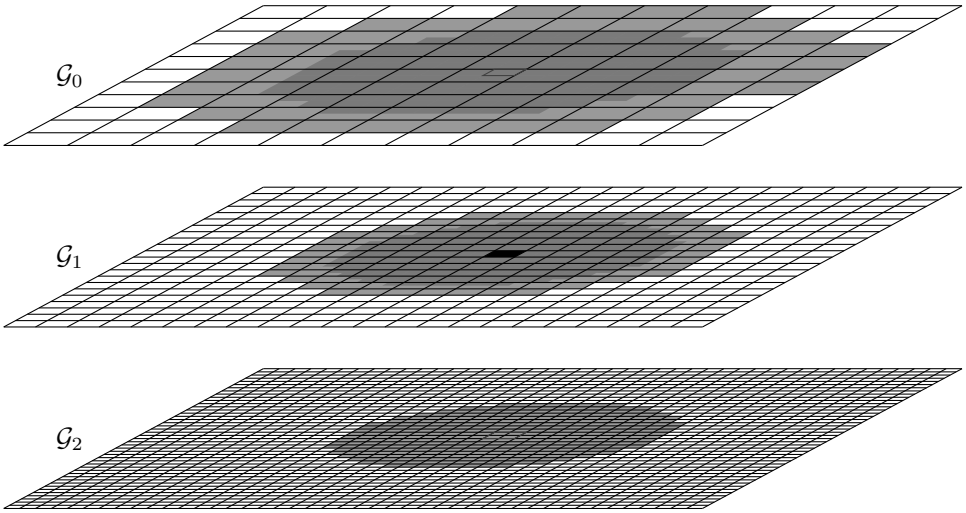


Figure 1 The neighborhood of the black colored cell in $\mathcal{G}_1$. The area of the neighboring cells in one level beneath is colored in a darker shade. When representing the grids via quadtrees every neighboring cell is either a child of another neighboring cell, a child of the black cell, or in $\mathcal{G}_0$.

not need to update the MBM. Instead, we can directly insert an edge into the underlying connectivity structure.

▶ **Definition 3.1.** Let s be a disk and $\sigma \in \mathcal{G}_i$ a cell, for some $i \geq 0$. Then, σ is *fully contained* in s if and only if σ intersects s and cannot contain disks intersecting the boundary of s.

Second, we bound the number of cells that still require an MBM after considering the fully contained cells.

▶ **Lemma 3.2.** *Inserting or deleting a disk s of radius r into the data structure of Lemma 2.3 requires checking $\mathcal{O}(\frac{r}{\varphi})$ cells that may contain disks intersecting the boundary of s. Those can be found in $\mathcal{O}(\frac{r}{\varphi})$ time.*

These are exactly all cells within some distance of the updated disk's boundary, where the distance depends on the disk's radius and the cell diameter. They must be either a cell of $\mathcal{G}_0$ or a child of a cell of that type, allowing the retrieval in $\mathcal{O}(\frac{r}{\varphi})$ time by a simple top-down traversal. See Figure 2. Another look at the cells during the recursive retrieval yields the following corollary.

▶ **Corollary 3.3.** *Inserting or deleting a disk s of radius r into the data structure of Lemma 2.3 requires checking $\mathcal{O}((\frac{r}{\varphi})^2)$ cells fully contained in s. Those can be found in $\mathcal{O}((\frac{r}{\varphi})^2)$ time.*

Among all paths in the quadtrees there are $\mathcal{O}(\frac{r}{\varphi})$ topmost cells fully contained in s. These can be found in $\mathcal{O}(\frac{r}{\varphi})$ time. Their interiors are pairwise disjoint and their union is exactly the union of all cells fully contained in s.

Using Corollary 3.3, we can save some time during updates: we do not update the MBM to inner cells, but insert an edge directly into the underlying edge connectivity structure.

▶ **Lemma 3.4.** *There is a data structure for dynamic disk connectivity with expected amortized update time $\mathcal{O}((\frac{1}{\varphi})^2 \log^2 n + \frac{1}{\varphi} 2^{\alpha(n)} \log^{10} n)$ and worst case time $\mathcal{O}(\frac{\log n}{\log \log n})$ for connectivity queries while requiring $\mathcal{O}((\frac{1}{\varphi}) n \log^3 n + (\frac{1}{\varphi})^2 n)$ expected space.*

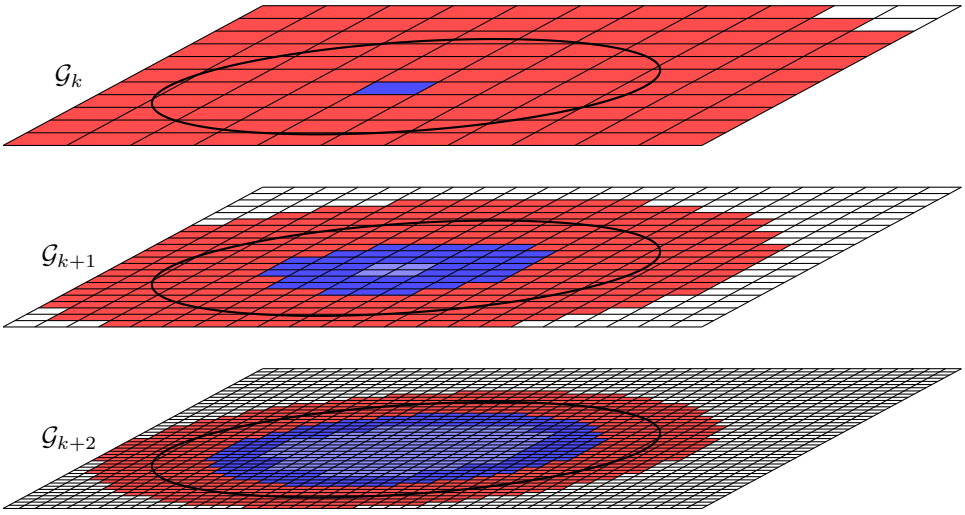


Figure 2 The types of cells which require checking when updating a disk in Lemma 2.3: fully contained (topmost) fully contained may contain disks intersecting the boundary

Proof. We augment the data structure of Lemma 2.3. In addition to an MBM, we save a counter for each neighboring pair of cells of different size. The counter describes how many of the disks of the larger cell fully contain the smaller cell.

When updating a pair during insertion or deletion of a disk s, we now update either the counter or the MBM. If s fully contains the other cell we update the counter, otherwise the MBM. Both cells' content intersect if and only if the counter is non-zero and the smaller cell is non-empty or the MBM contains an edge. Depending on whether this condition changes during the update, the edge connectivity data structure must be updated as well.

When updating a disk s, we encounter $\mathcal{O}(\log \frac{1}{\varphi})$ neighboring cells until we reach the level where s must be inserted or deleted. For each of these, an update to the MBM is required. Afterwards, we recurse down according to Corollary 3.3 to retrieve all $\mathcal{O}((\frac{1}{\varphi})^2)$ fully contained cells and the remaining $\mathcal{O}(\frac{1}{\varphi})$ cells described in Lemma 3.2 and update the MBMs, counters, and connectivity structure accordingly. ◀

3.2 Query for Replacements Instead

We were able to reduce the cost of the quadratic part, but didn't eliminate it. As we can have $\mathcal{O}((\frac{1}{\varphi})^2)$ edge changes during an update we cannot fully handle them in the update. Thus, we move parts of the handling of fully contained cells into the query.

▶ **Lemma 3.5.** *The result of a query into the data structure of Lemma 3.4 does not change when we omit all edges introduced solely through fully contained cells (i.e. non-zero counters), except those involving the cells containing at least one of the query disks.*

See Figure 3 (a). By Lemma 3.5, we can ignore fully contained cells during updates, except maintaining for each cell which disks fully contain them as topmost. Instead of inserting the required edges during a query, we query for suitable representatives.

Fix a query for two disks s_1, s_2. Consider two other disks t_1, t_2, which contain s_1 with the radius of t_1 not smaller than the radius of t_2. If there is a path in the intersection graph between s_1 and s_2, then there is a path from t_1 to s_2, even when s_1 is removed. Additionally, when s_2 is not contained in t_1, then the boundary of t_1 must intersect the boundary of

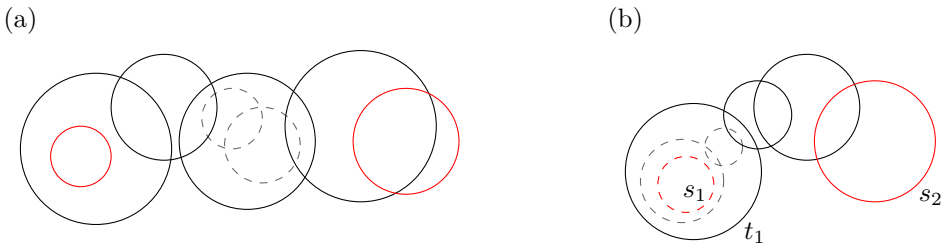


Figure 3 (a) A path between the two red disks uses the black disks as intermediates. Note that any non-red disk contained by a black disk is not required to form a path and can be safely ignored. (b) Removing the dashed disks and querying for t_1 instead of s_1 still leads to a valid path to s_2.

another disk of the path. Thus, all disks contained by t_1 (except s_2) can be removed without changing connectivity between t_1 and s_2, possibly including t_2; see Figure 3 (b). The removed disks include all those of Lemma 3.5 whose edges we exempted from the removal.

Still, for a representative this would require obtaining the largest disk fully containing a cell encountered on the path to a queried disk. Consider two different cells on the path down to s_1 and for each the largest disk which fully contains it but not its parent. The disk obtained at the larger cell must also fully contain the other chosen cell, but not the other way round. Thus, both disks intersect or the disk of the larger cell contains the other one. Therefore, we only need to retrieve the largest disk of the topmost cell of the path which was fully contained by some disk when replacing s_1 and s_2.

▶ **Lemma 3.6.** *Fix an instance of the data structure of Lemma 3.4 and two disks s_1, s_2. Let c_1, c_2 be the first cells on the paths to s_1, s_2 that are fully contained by a disk as topmost. Let s_1', s_2' be the largest such disks. A query for s_1, s_2 has the same result as a query for s_1', s_2' with all edges added solely through fully contained cells (i.e. non-zero counters) omitted.*

The dynamic nested rectangle intersection data structure by Kaplan et al. [6, Section 5] allows inserting or deleting nested or disjoint rectangles with a priority in amortized time $\mathcal{O}(\log^2 n)$ and retrieving the highest priority rectangle containing a query point in amortized time $\mathcal{O}(\log n)$, while requiring $\mathcal{O}(n \log n)$ space. We can use it to retrieve the representatives without a dependency on $\frac{1}{\varphi}$ in the query time by storing all fully contained topmost cells.

▶ **Theorem 3.7.** *Let $0 < \varphi \leq 1$, and let P be a set of disks with radius in $[\varphi, 1]$. There is a dynamic connectivity structure for the intersection graph of P such that the insertion or deletion of a disk takes amortized expected time $\mathcal{O}(\frac{1}{\varphi} 2^{\alpha(n)} \log^{10} n)$ and a connectivity query takes amortized time $\mathcal{O}(\log n)$, where n is the maximum number of sites at any time and $\alpha(n)$ is the inverse Ackermann function. It requires $\mathcal{O}(\frac{1}{\varphi} n \log^3 n)$ expected space.*

Proof. We extend the data structure of Lemma 2.3 similar to Lemma 3.4. Instead of a counter, we maintain for each cell an AVL tree [8] of all disks which fully contain this cell as topmost, ordered by radius. Also, we maintain in each node the disks count in its subtree.

In addition, we maintain a rectangle intersection data structure as by Kaplan et al., such that each cell with a non-empty AVL tree and a non-zero disk count in its subtree is inserted into the data structure with its size as priority. Due to this and Corollary 3.3, we have at most $\mathcal{O}(\min(n^2, \frac{1}{\varphi} n))$ cells in the rectangular intersection data structure simultaneously.

During a disk update we have $\mathcal{O}(\frac{1}{\varphi})$ updates to the rectangle structure via changes to the topmost fully contained cells and along the path in the quadtrees $\mathcal{O}(\log \frac{1}{\varphi})$ updates via

changes to counters. As each of these updates requires $\mathcal{O}(\log(n^2)) = \mathcal{O}(\log n)$ amortized time and we maintain fewer MBMs, the overall update time is reduced by a factor of $\frac{1}{\varphi}$.

Queries are done via finding the representatives described in Lemma 3.6 with the help of the rectangle data structure and the AVL trees in $\mathcal{O}(\log(n^2) + \log n) = \mathcal{O}(\log n)$ amortized time and then querying the connectivity structure as before.

We need $\mathcal{O}(\frac{1}{\varphi} n \log(n^2))$ space for the rectangular intersection data structure and $\mathcal{O}(\frac{1}{\varphi} n)$ for the trees and counters, but maintain fewer MBMs. Thus, less expected space is required. ◀

References

1 Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark H. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, third edition, 2008.

2 Timothy M. Chan, Mihai Pătraşcu, and Liam Roditty. Dynamic connectivity: Connecting to networks and geometry. *SIAM J. Comput.*, 40(2):333–349, 2011.

3 David Eppstein, Giuseppe F. Italiano, Roberto Tamassia, Robert Endre Tarjan, Jeffery Westbrook, and Moti Yung. Maintenance of a minimum spanning forest in a dynamic plane graph. *J. Algorithms*, 13(1):33–54, 1992.

4 Monika Rauch Henzinger and Valerie King. Randomized fully dynamic graph algorithms with polylogarithmic time per operation. *J. ACM*, 46:502–516, 1999.

5 Jacob Holm, Kristian de Lichtenberg, and Mikkel Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. *J. ACM*, 48(4):723–760, 2001.

6 Haim Kaplan, Eyal Molad, and Robert E. Tarjan. Dynamic rectangular intersection with priorities. In *Proc. 35th Annu. ACM Sympos. Theory Comput. (STOC)*, pages 639–648, 2003.

7 Haim Kaplan, Wolfgang Mulzer, Liam Roditty, Paul Seiferth, and Micha Sharir. Dynamic planar Voronoi diagrams for general distance functions and their algorithmic applications. In *Proc. 28th Annu. ACM-SIAM Sympos. Discrete Algorithms (SODA)*, pages 2495–2504, 2017. Full version at `arXiv:1604.03654`.

8 Donald E. Knuth. *The Art of Computer Programming. Vol. 3. Sorting and Searching*. Addison-Wesley, 2nd edition, 1998.

9 Mihai Pătraşcu and Mikkel Thorup. Planning for fast connectivity updates. In *Proc. 48th Annu. IEEE Sympos. Found. Comput. Sci. (FOCS)*, pages 263–271, 2007.

10 Paul Seiferth. *Disk Intersection Graphs: Models, Data Structures, and Algorithms*. PhD thesis, Freie Universität Berlin, Germany, 2016.

11 Mikkel Thorup. Near-optimal fully-dynamic graph connectivity. In *Proc. 32nd Annu. ACM Sympos. Theory Comput. (STOC)*, pages 343–350, 2000.

Terrain-Like and Non-Jumping Graphs*

Stav Ashur[1], Omrit Filtser[2], and Rachel Sababn[3]

1 **Department of Computer Science, Ben-Gurion University of the Negev**
`stavshe@post.bgu.ac.il`
2 **Department of Computer Science, Ben-Gurion University of the Negev**
`omritna@post.bgu.ac.il`
3 **Department of Computer Science, Ben-Gurion University of the Negev**
`rachelfr@post.bgu.ac.il`

Abstract

Let $G = (V, E)$ be a graph with n vertices. A *labeling* of (the vertices of) G is an injective function $\pi : V \to [n]$. We say that π is a *terrain-like labeling* of G if for any four vertices a, b, c, d such that $\pi[a] < \pi[b] < \pi[c] < \pi[d]$, if both $\{a, c\}$ and $\{b, d\}$ are in E, then so is $\{a, d\}$. The graph G is *terrain-like* if it has a terrain-like labeling. Similarly, π is a *non-jumping labeling* of G (Ahmed et al., 2017) if for any four vertices a, b, c, d such that $\pi[a] < \pi[b] < \pi[c] < \pi[d]$, if both $\{a, c\}$ and $\{b, d\}$ are in E, then so is $\{b, c\}$. The graph G is *non-jumping* if it has a non-jumping labeling (see Figure 1). In this paper we compare terrain-like graphs and non-jumping graphs, answering on the way a question raised by Ahmed et al. concerning the latter family.

1 Introduction

The family of terrain-like graphs was introduced by Ashur et al. [2], extending a manuscript of Katz [5]. Ashur et al. adapt the PTAS of Gibson et al. [4] for vertex guarding the vertices of x-monotone terrains, to obtain a PTAS for *minimum dominating set* (MDS) in terrain-like graphs. Then, by showing that the *visibility graphs* of *weakly-visible* polygons and terrains are terrain-like, they immediately obtain similar PTASs for guarding such polygons and terrains.

Ahmed et al. [1] defined the family of non-jumping graphs and proved that it is equivalent to the family of *monotone L-graphs* and thus admits a PTAS for MDS [3]. They showed that several well-known graph families, such as outerplanar graphs, convex bipartite graphs, and complete graphs, are subfamilies of non-jumping graphs and are therefore also monotone L-graphs. They also gave an example of a (non-planar) graph which is jumping (i.e. not non-jumping), providing a long and involved proof for it, and raised the question whether all planar graphs are non-jumping (and thus can be realized as monotone L-graphs).

Denote by $\mathcal{F}_{NJ}$ and $\mathcal{F}_{TL}$ the families of non-jumping and terrain-like graphs, respectively. The resemblance between the definitions of $\mathcal{F}_{NJ}$ and $\mathcal{F}_{TL}$, together with the fact that many of the graph families that were found to be non-jumping in [1] (including those mentioned above) are also terrain-like, raises the question what is the connection between them?

In this paper, we investigate the relation between these two graph families. First, we present a natural infinite family of graphs that are in $\mathcal{F}_{TL}$ but not in $\mathcal{F}_{NJ}$, and give a short and simple proof for it. Moreover, the smallest member of this family is a planar graph, implying that there exist planar graphs that cannot be realized as monotone L-graphs. Then, we present some basic properties of the terrain-like labeling function, and use them to prove that there exists an infinite family of graphs that are in $\mathcal{F}_{NJ}$ but not in $\mathcal{F}_{TL}$. Finally, we present a family of graphs which are not in $\mathcal{F}_{TL} \cup \mathcal{F}_{NJ}$.

* Work by the authors was partially supported by the Lynn and William Frankel Center for Computer Sciences.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 19–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

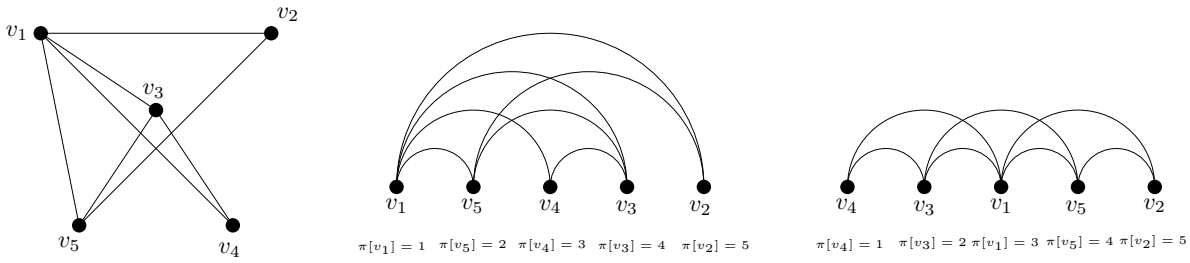


Figure 1 Left: The graph G. Center: A terrain-like (and jumping) labeling of G. Right: A non-jumping (and not terrain-like) labeling of G.

2 $\mathcal{F}_{TL}$ vs. $\mathcal{F}_{NJ}$

▶ **Theorem 2.1.** $\mathcal{F}_{TL} \not\subseteq \mathcal{F}_{NJ}$

Proof. Let $K_n = (V = \{v_1, v_2, \ldots, v_n\}, E)$ be the complete graph on n vertices. For $n \geq 6$, let $K_n^{-3} = (V, E \setminus \{e_1, e_2, e_3\})$, where e_1, e_2, e_3 are any three pairwise-disjoint edges in E. We show that for any $n \geq 6$, $K_n^{-3} \in \mathcal{F}_{TL} \setminus \mathcal{F}_{NJ}$. Assume w.l.o.g. that $e_1 = \{v_1, v_2\}$, $e_2 = \{v_3, v_4\}$, and $e_3 = \{v_5, v_6\}$.
$K_n^{-3} \in \mathcal{F}_{TL}$: Consider the labeling $\pi[v_i] = i$. For any 4 vertices $v_{i_1}, v_{i_2}, v_{i_3}, v_{i_4}$ such that $i_1 < i_2 < i_3 < i_4$, we have $\{v_{i_1}, v_{i_4}\} \in E$ since $i_4 - i_1 \geq 3$; thus π is a terrain-like labeling.
$K_n^{-3} \notin \mathcal{F}_{NJ}$: Assume by contradiction that $K_n^{-3} \in \mathcal{F}_{NJ}$, then there exists a non-jumping labeling π of K_n^{-3}. Assume w.l.o.g. that $\pi[v_1] < \pi[v_2]$. We claim that either $\pi[v_1] = 1$ or $\pi[v_2] = n$. Indeed, assume that $\pi[v_i] = 1$ for some $i \neq 1$ and $\pi[v_j] = n$ for some $j \neq 2$. Notice that $\{v_i, v_2\}$ and $\{v_1, v_j\}$ are edges of the graph, but $\{v_1, v_2\}$ is not an edge of the graph, so π is not a non-jumping labeling w.r.t. v_i, v_1, v_2, v_j — a contradiction. By symmetry, the above claim holds also for v_3, v_4 and for v_5, v_6, but then π is not an injective function. ◀

As a corollary, we get that not all planar graphs are non-jumping, thus answering the question raised by Ahmed et al. [1]. Indeed, it is easy to verify that K_6^{-3} is planar (see Figure 2).

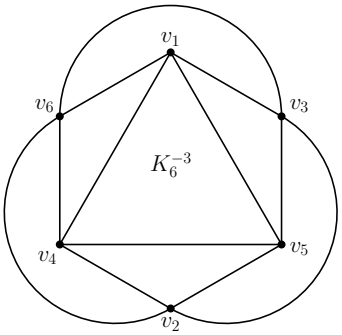


Figure 2 A planar embedding of K_6^{-3}.

2.1 Some properties of labeling functions

▶ **Observation 2.2.** *Let $G = (V, E)$ be a graph and let $H = (V', E')$ be an induced graph of G (i.e., $V' \subseteq V$ and $E' = \{\{u, v\} \mid u, v \in V', \{u, v\} \in E\}$). Let $\pi : V \to [|V|]$ be a terrain-like*

(resp., a non-jumping) labeling of G, and let $\pi' : V' \to [|V'|]$ be a labeling of H such that $\pi'[v_i] < \pi'[v_j]$ if and only if $\pi[v_i] < \pi[v_j]$. Then π' is a terrain-like (resp., a non-jumping) labeling of H.

Denote by $P_n = (V, E)$ the path graph with n vertices, such that $V = \{v_1, \ldots, v_n\}$ and $E = \{\{v_i, v_{i+1}\} \mid 1 \leq i \leq n-1\}$.

▶ **Lemma 2.3.** *Let π be a terrain-like labeling of P_n such that $\pi[v_1] = 1$ and $\pi[v_n] = n$, then $\pi[v_i] = i$ for $i = 1, \ldots, n$.*

Proof. Let j be the largest index such that $\pi[v_i] = i$ for $i = 1, \ldots, j$. If $j = n$ then we are done. Otherwise, $j \leq n-3$ and $\pi[v_{j+1}] = k$, for some $j+1 < k < n$. Let l be the largest index such that $\pi[v_j] < \pi[v_l] < \pi[v_{j+1}]$, then $\pi[v_{j+1}] < \pi[v_{l+1}]$. But now π is not a terrain-like labeling w.r.t. $v_j, v_l, v_{j+1}, v_{l+1}$, since $\{v_j, v_{l+1}\} \notin E$, so j must be n. ◀

Denote by $C_n = (V, E)$ the cycle graph with n vertices, such that $V = \{v_1, v_2, \ldots, v_n\}$ and $E = \{\{v_i, v_{i+1}\} \mid 1 \leq i \leq n-1\} \cup \{\{v_1, v_n\}\}$.

▶ **Lemma 2.4.** *Let π be a terrain-like (alternatively, a non-jumping) labeling of C_n such that $\pi[v_1] = 1$, then either $\pi[v_n] = n$ or $\pi[v_2] = n$.*

Proof. Assume that $\pi[v_2] < \pi[v_n]$. If $\pi[v_n] = n$ then we are done. Otherwise, let j be the smallest index such that $\pi[v_n] < \pi[v_j]$, and notice that $j \geq 3$. But now π is neither a terrain-like nor a non-jumping labeling w.r.t. v_1, v_{j-1}, v_n, v_j, since both $\{v_1, v_j\}$ and $\{v_{j-1}, v_n\}$ are not in E. The case $\pi[v_n] < \pi[v_2]$ is symmetric. ◀

▶ **Lemma 2.5.** *Let π be a terrain-like labeling of C_n. Assume w.l.o.g. that $\pi[v_1] = 1$ and $\pi[v_2] < \pi[v_n]$, then either:*

1. $\pi[v_1] < \pi[v_2] < \pi[v_3] < \cdots < \pi[v_{n-1}] < \pi[v_n]$, *or*
2. $\pi[v_1] < \pi[v_{n-1}] < \pi[v_{n-2}] < \cdots < \pi[v_2] < \pi[v_n]$.

Proof. By Lemma 2.4, $\pi[v_n] = n$, and thus for any $1 < i < n$ we have $\pi[v_1] < \pi[v_i] < \pi[v_n]$. First, we claim that if $\pi[v_2] < \pi[v_i]$ for some $3 \leq i \leq n-2$, then $\pi[v_2] < \pi[v_{i+1}]$. Indeed, if $\pi[v_1] < \pi[v_{i+1}] < \pi[v_2] < \pi[v_i]$ then we have $\{v_1, v_2\}, \{v_i, v_{i+1}\} \in E$ but $\{v_1, v_i\} \notin E$. Symmetrically, if $\pi[v_{n-1}] < \pi[v_i]$ for some $2 \leq i \leq n-3$, then $\pi[v_{n-1}] < \pi[v_{i+1}]$.

Secondly, we claim that if $\pi[v_i] < \pi[v_2]$ for some $3 \leq i \leq n-2$, then $\pi[v_{i+1}] < \pi[v_2]$. Indeed, if $\pi[v_1] < \pi[v_i] < \pi[v_2] < \pi[v_{i+1}]$ then we have $\{v_1, v_2\}, \{v_i, v_{i+1}\} \in E$ but $\{v_1, v_{i+1}\} \notin E$.

Therefore we can only have the following two cases:

1. If $\pi[v_2] < \pi[v_3] < \pi[v_n]$, then by the first claim we have $\pi[v_2] < \pi[v_j] < \pi[v_n]$ for $j = 3, \ldots, n-1$. By Lemma 2.3 on the induced path $v_2, v_3, \ldots, v_n$ we get that $\pi[v_1] < \pi[v_2] < \pi[v_3] < \cdots < \pi[v_{n-1}] < \pi[v_n]$.
2. If $\pi[v_1] < \pi[v_3] < \pi[v_2]$, then by the second claim we have $\pi[v_1] < \pi[v_j] < \pi[v_2]$ for $j = 3, \ldots, n-1$, and, since $\pi[v_{n-1}] < \pi[2]$, by the first claim we have $\pi[v_{n-1}] < \pi[v_j]$ for $j = 2, \ldots, n-2$. Again by Lemma 2.3 on the induced path $v_{n-1}, \ldots, v_3, v_2$ we get that $\pi[v_1] < \pi[v_{n-1}] < \pi[v_{n-2}] < \cdots < \pi[v_2] < \pi[v_n]$.

◀

▶ **Theorem 2.6.** $\mathcal{F}_{NJ} \not\subseteq \mathcal{F}_{TL}$

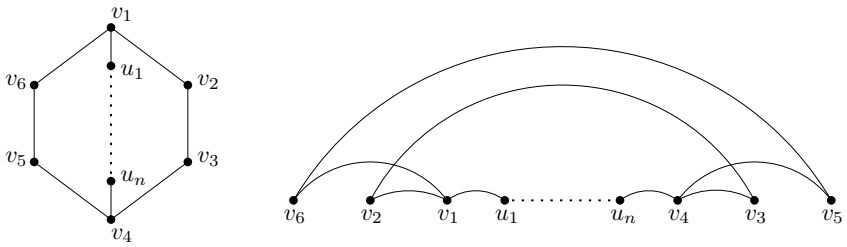


Figure 3 Left: The graph G. Right: A non-jumping labeling of G, i.e. $\pi[v_6] = 1, \pi[v_2] = 2, \pi[v_1] = 3, \pi[u_1] = 4, \ldots, \pi[u_n] = n+3, \pi[v_4] = n+4, \pi[v_3] = n+5, \pi[v_5] = n+6$.

Proof. Let C_6 be the cycle graph with vertex set $V = \{v_1, v_2, \ldots, v_6\}$, and P_n the path graph with vertex set $U = \{u_1, u_2, \ldots, u_n\}$, $n \geq 2$. Consider the graph $G = (V \cup U, E)$, where $E = E(C_6) \cup E(P_n) \cup \{\{v_1, u_1\}, \{v_4, u_n\}\}$. In other words, G contains an induced cycle on 6 vertices $v_1, v_2, \ldots, v_6$, and an induced path on $n+2$ vertices $v_1, u_1, u_2, \ldots, u_n, v_4$; see Figure 3 (left).

$G \in \mathcal{F}_{NJ}$

Figure 3 (right) shows a non-jumping labeling of G, so G is in $\mathcal{F}_{NJ}$.

$G \notin \mathcal{F}_{TL}$

Assume by contradiction that G is in $\mathcal{F}_{TL}$, then there exists a terrain-like labeling $\pi : V \cup U \to [n+6]$. Let $\pi_V : V \to [6]$ be a labeling such that $\pi_V[v_i] < \pi_V[v_j]$ if and only if $\pi[v_i] < \pi[v_j]$. Since C_6 is an induced cycle, Lemmas 2.4 and 2.5 can be applied to π_V. By Lemma 2.4, there must be an edge between the first and last vertex in the labeling π_V. Formally, if $\pi_V[v_i] = 1$ and $\pi_V[v_j] = 6$, then $\{v_i, v_j\} \in E$. There are 6 edges in C_6, so there are 6 possible choices of $e = \{v_i, v_j\}$, but we observe that the graph is symmetric for all the edges in $\{\{v_1, v_6\}, \{v_1, v_2\}, \{v_4, v_5\}, \{v_4, v_3\}\}$, and for all the edges in $\{\{v_5, v_6\}, \{v_2, v_3\}\}$. Thus, w.l.o.g. we only consider the following two cases: either $e = \{v_1, v_6\}$ or $e = \{v_5, v_6\}$. By Lemma 2.5 we have four cases for the labeling of V:

1. $\pi[v_1] < \pi[v_2] < \pi[v_3] < \pi[v_4] < \pi[v_5] < \pi[v_6]$
2. $\pi[v_1] < \pi[v_5] < \pi[v_4] < \pi[v_3] < \pi[v_2] < \pi[v_6]$
3. $\pi[v_6] < \pi[v_1] < \pi[v_2] < \pi[v_3] < \pi[v_4] < \pi[v_5]$
4. $\pi[v_6] < \pi[v_4] < \pi[v_3] < \pi[v_2] < \pi[v_1] < \pi[v_5]$

Cases 1 and 3: It is not hard to verify that either $\pi[v_3] < \pi[u_n] < \pi[v_4]$, or $\pi[v_4] < \pi[u_n] < \pi[v_5]$. Thus either $\pi[v_3] < \pi[u_i] < \pi[v_4]$ for all $1 \leq i \leq n$, or $\pi[v_4] < \pi[u_i] < \pi[v_5]$ for all $1 \leq i \leq n$. If $\pi[v_3] < \pi[u_1] < \pi[v_4]$, then the labeling $\pi[v_1] < \pi[v_3] < \pi[u_1] < \pi[v_4]$ contradicts the terrain-like property, and if $\pi[v_4] < \pi[u_1] < \pi[v_5]$, then the labeling $\pi[v_1] < \pi[v_4] < \pi[u_1] < \pi[v_5]$ is a contradiction.

Case 2: Again, we have either $\pi[v_4] < \pi[u_i] < \pi[v_3]$ for all $1 \leq i \leq n$, or $\pi[v_5] < \pi[u_i] < \pi[v_4]$ for all $1 \leq i \leq n$. If $\pi[v_4] < \pi[u_1] < \pi[v_3]$, then the labeling $\pi[v_1] < \pi[v_4] < \pi[u_1] < \pi[v_3]$ contradicts the terrain-like property, and if $\pi[v_5] < \pi[u_1] < \pi[v_4]$, then the labeling $\pi[v_1] < \pi[v_5] < \pi[u_1] < \pi[v_4]$ is a contradiction.

Case 4: Notice that either $\pi[v_6] < \pi[u_n] < \pi[v_4]$, or $\pi[v_4] < \pi[u_n] < \pi[v_3]$. Thus either $\pi[v_6] < \pi[u_i] < \pi[v_4]$ for all $1 \leq i \leq n$, or $\pi[v_4] < \pi[u_i] < \pi[v_3]$ for all $1 \leq i \leq n$. If

$\pi[v_6] < \pi[u_1] < \pi[v_4]$, then the labeling $\pi[u_1] < \pi[v_4] < \pi[v_1] < \pi[v_5]$ contradicts the terrain-like property, and if $\pi[v_4] < \pi[u_1] < \pi[v_3]$, then the labeling $\pi[v_4] < \pi[u_1] < \pi[v_3] < \pi[v_1]$ is a contradiction. ◀

Finally, does every graph belong either to $\mathcal{F}_{TL}$ or to $\mathcal{F}_{NJ}$? The answer is clearly no, since, in general, minimum dominating set is NP-hard to approximate within a factor of $\Omega(\log n)$ [6]. Nevertheless, it would be nice to see a concrete and simple example. Below, we present an infinite family of graphs which are neither in $\mathcal{F}_{TL}$ nor in $\mathcal{F}_{NJ}$.

The Harary graphs $H_{n,k}$ are k-connected graphs on n vertices, having the smallest possible number of edges. When n is even and k is odd, $H_{n,k}$ is defined as follows: $H_{n,k} = (V = \{v_0, ..., v_{n-1}\}, E_1 \cup E_2)$, where $E_1 = \{\{v_i, v_{i+j}\} | 1 \leq j \leq \lfloor \frac{k}{2} \rfloor, 0 \leq i \leq n-1\}$ and $E_2 = \{\{v_i, v_{i+\frac{n}{2}}\} | 0 \leq i \leq \frac{n}{2} - 1\}$ (where the addition is modulo n), see Figure 4.

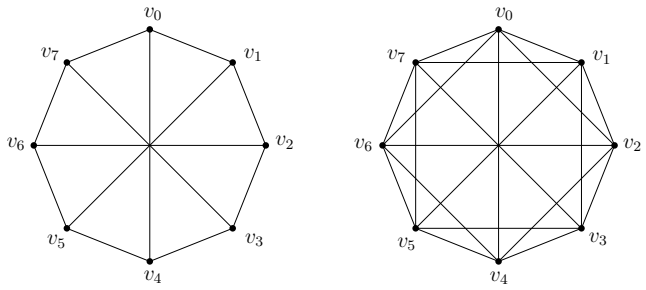


Figure 4 $H_{8,3}$ (left) and $H_{8,5}$ (right).

▶ **Theorem 2.7.** *For any $m \geq 4$, $H_{2m,3}$ is neither in $\mathcal{F}_{TL}$ nor in $\mathcal{F}_{NJ}$.*

Since we are interested in a simple example, we prove the theorem here only for $H_{8,3}$.

Proof. **(For $m = 4$)** Assume by contradiction that π is a non-jumping labeling of $H_{8,3}$, and assume w.l.o.g. that $\pi[v_0] = 1$. Since $C_1 = (v_0, v_1, v_2, v_3, v_4)$ and $C_2 = (v_0, v_4, v_5, v_6, v_7)$ are induced cycles, we can apply Lemma 2.4, and get 3 cases: (i) $\pi[v_4] = 8$, (ii) $\pi[v_1] = 8$, or (iii) $\pi[v_7] = 8$. Notice that (ii) and (iii) are symmetric cases, so we consider only cases (i) and (ii). We denote the labeling of C_1 by π_1 and the labeling of C_2 by π_2.

(i) Assume w.l.o.g. that $\pi[v_0] < \pi[v_1] < \pi[v_7] < \pi[v_4]$, then for any possible labeling of v_5 we get that π is not a non-jumping labeling: if $\pi[v_7] < \pi[v_5]$ then we have $\{v_0, v_7\}, \{v_1, v_5\} \in E$ but $\{v_1, v_7\} \notin E$, and if $\pi[v_5] < \pi[v_7]$ then we have $\{v_0, v_7\}, \{v_4, v_5\} \in E$ but $\{v_5, v_7\} \notin E$.

(ii) Notice that $\pi[v_0] < \pi[v_2] < \pi[v_4] < \pi[v_1]$ is not possible, so assume $\pi[v_0] < \pi[v_4] < \pi[v_2] < \pi[v_1]$. We notice that either $\pi_2[v_4] = 5$ or $\pi_2[v_7] = 5$. If $\pi_2[v_4] = 5$ then since $\pi_2[v_6] < 5$ we get that $\pi[v_6] < \pi[v_4]$, but then we have $\{v_0, v_4\}, \{v_6, v_2\} \in E$ but $\{v_4, v_6\} \notin E$. If $\pi_2[v_7] = 5$, then since $\pi_2[v_5] < 5$ we get that $\pi[v_5] < \pi[v_7]$, but then we have $\{v_0, v_7\}, \{v_5, v_1\} \in E$ but $\{v_5, v_7\} \notin E$.

Now assume by contradiction that π is a terrain-like labeling of $H_{8,3}$, and assume w.l.o.g. that $\pi[v_0] = 1$. Again by applying Lemma 2.4 we have four cases: (i) $1 = \pi[v_0] < \pi[v_1] < \pi[v_2] < \pi[v_3] < \pi[v_4] = 8$, (ii) $1 = \pi[v_0] < \pi[v_3] < \pi[v_2] < \pi[v_1] < \pi[v_4] = 8$, (iii) $1 = \pi[v_0] < \pi[v_4] < \pi[v_3] < \pi[v_2] < \pi[v_1] = 8$, (iv) $1 = \pi[v_0] < \pi[v_2] < \pi[v_3] < \pi[v_4] < \pi[v_1] = 8$.

(i) We first get that $\pi[v_0] < \pi[v_5] < \pi[v_1]$ since any other labeling results in a contradiction, and then any labeling of v_6 given $1 = \pi[v_0] < \pi[v_5] < \pi[v_1] < \pi[v_2] < \pi[v_3] < \pi[v_4] = 8$ is impossible.

(ii) There are 2 possibilities for labeling v_5: either $\pi[v_0] < \pi[v_5] < \pi[v_3]$ or $\pi[v_1] < \pi[v_5] < \pi[v_4]$. If $\pi[v_0] < \pi[v_5] < \pi[v_3]$ then we have $\pi[v_2] < \pi[v_6] < \pi[v_1]$ and no possible labeling for v_7. If $\pi[v_1] < \pi[v_5] < \pi[v_4]$ then there is no possible labeling for v_6.

(iii) We first get that $\pi[v_0] < \pi[v_5] < \pi[v_4]$ since any other labeling results in a contradiction, and then any labeling of v_6 given $1 = \pi[v_0] < \pi[v_5] < \pi[v_4] < \pi[v_3] < \pi[v_2] < \pi[v_1] = 8$ is impossible.

(iv) There are 2 possibilities for labeling v_5: either $\pi[v_0] < \pi[v_5] < \pi[v_2]$ or $\pi[v_4] < \pi[v_5] < \pi[v_1]$. If $\pi[v_0] < \pi[v_5] < \pi[v_2]$ then we have $\pi[v_5] < \pi[v_6] < \pi[v_2]$ and no possible labeling for v_7. If $\pi[v_4] < \pi[v_5] < \pi[v_1]$ then there is no possible labeling for v_6.

◀

References

1 Abu Reyan Ahmed, Felice De Luca, Sabin Devkota, Alon Efrat, Md Iqbal Hossain, Stephen Kobourov, Jixian Li, Sammi Abida Salma, and Eric Welch. L-graphs and monotone L-graphs. *arXiv:1703.01544*, 2017.

2 Stav Ashur, Omrit Filtser, Matthew J. Katz, and Rachel Saban. Terrain-like graphs: PTASs for guarding weakly-visible polygons and terrains. Manuscript, 2018.

3 Sayan Bandyapadhyay, Anil Maheshwari, Saeed Mehrabi, and Subhash Suri. Approximating dominating set on intersection graphs of rectangles and l-frames. In *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018, August 27-31, 2018, Liverpool, UK*, pages 37:1–37:15, 2018. URL: `https://doi.org/10.4230/LIPIcs.MFCS.2018.37`, `doi:10.4230/LIPIcs.MFCS.2018.37`.

4 Matt Gibson, Gaurav Kanade, Erik Krohn, and Kasturi Varadarajan. Guarding terrains via local search. *Journal of Computational Geometry*, 5(1):168–178, 2014.

5 Matthew J. Katz. A PTAS for vertex guarding weakly-visible polygons — an extended abstract. *arXiv:1803.02160*, 2018.

6 Ran Raz and Shmuel Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 475–484, 1997. URL: `https://doi.org/10.1145/258533.258641`, `doi:10.1145/258533.258641`.

Recognizing Planar Laman Graphs

Jonathan Rollin[1], Lena Schlipf[1], and André Schulz[1]

1 **FernUniversität in Hagen**
{jonathan.rollin | lena.schlipf | andre.schulz}@fernuni-hagen.de

Abstract

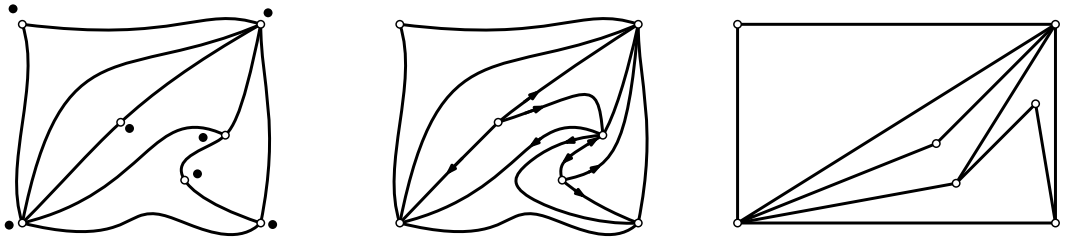
Laman graphs are the minimally rigid graphs in the plane. We present two algorithms for recognizing planar Laman graphs. A simple algorithm with running time $O(n^{3/2})$ and another one with running time $O(n \log^3 n)$ based on latest planar network flow algorithms. Both improve upon the previously fastest algorithm for general graphs by Gabow and Westermann [*Algorithmica*, 7(5-6):465–497, 1992] with running time $O(n\sqrt{n \log n})$.

1 Introduction

Let $G = (V, E)$ be a graph with n vertices. The graph G is called a *Laman graph* if it has $2n - 3$ edges and every subset $V' \subseteq V$ induces a subgraph with no more than $2|V'| - 3$ edges. A *bar-joint framework* is a physical structure made from fixed-length bars that are linked by universal joints (allowing 360° rotations) at their endpoints. A bar-joint framework is *flexible* if it has a motion other than a global rotation or translation. A nonflexible framework is called *rigid*. Moreover it is called *minimally rigid*, if it is rigid, but it becomes flexible after removing any bar. Interestingly, in 2d a bar-joint framework (in a generic configuration) is minimally rigid, if and only if its underlying graph is a Laman graph.

Various characterizations of Laman graphs are known [9, 14, 15]. The class of *plane* Laman graphs provides even more structure [8, 12]. Of particular interest for our result is the following geometric characterization: A geometric graph is a *pointed pseudotriangulation* (PPT) if each inner face contains exactly three angles less than π, called *small*, and every vertex is incident to an angle larger than π, called *big* [19]. Streinu [20] proved that PPTs are Laman graphs. Moreover, Haas et al. [8] showed that every planar Laman graph can be realized as a PPT.

The concept of pointed pseudotriangulations can be transferred to plane (abstract) graphs. In fact, this was an intermediate step in the proof by Haas et al. A *combinatorial pointed pseudotriangulation* (CPPT) is a plane graph (with $2n - 3$ edges) with an assignment of the labels "small"/"big" to the angles satisfying the properties of a PPT. Not every CPPT can be stretched to a PPT, but every plane Laman graph admits a CPPT assignment and



(a) (b) (c)

Figure 1 A CPPT with big angles drawn marked by solid circles (a), the derived directed graph $\vec{G}$ (b), the stretched pointed pseudotriangulation (c).

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

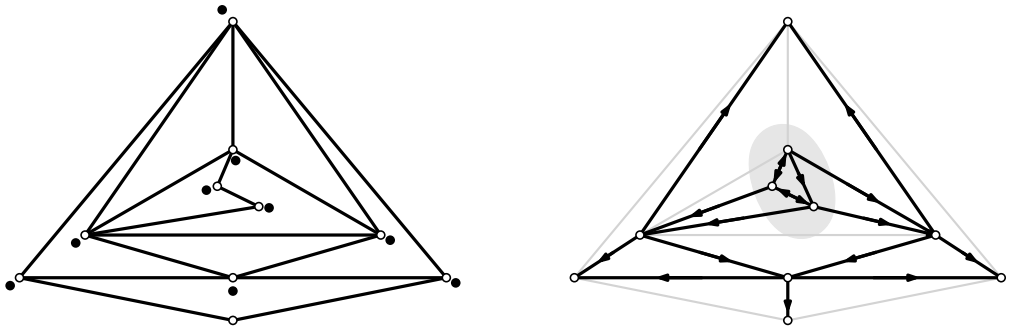


Figure 2 A CPPT that is not stretchable (a) and the derived directed graph $\vec{G}$ where the highlighted vertices do not have 3 disjoint paths to the outer face (b).

each CPPT which is a Laman graph is stretchable. Haas et al. [8] provide an algorithm with running time $O(n^{3/2})$ that finds a CPPT assignment for a planar Laman graph.

In order to recognize plane Laman graphs it is sufficient to check whether a given CPPT is stretchable. We provide such algorithms based on the following characterization of stretchability by Haas et al. by means of connectivity.

▶ **Lemma 1.1** ([8]). *For a CPPT G a directed plane graph $\vec{G}$, with $V(\vec{G}) = V(G)$, can be computed in linear time such that G is stretchable if and only if for each interior vertex $u \in \vec{G}$ there are 3 vertex disjoint directed paths from u to distinct vertices on the boundary of $\vec{G}$.*

The condition of Lemma 1.1 seems to be an interesting property on its own, since it can be understood as a form of "directed 3-connectivity".

1.1 Our contribution

Consider a positive integer k, a directed graph G, and disjoint sets $S, T \subseteq V(G)$. We call S *k-connected to T* if for each vertex $s \in S$ there are k directed paths from s to T pairwise having only vertex s in common.

▶ **Theorem 1.2.** *For each fixed $k \geq 1$ there is an algorithm deciding for a directed planar graph G and a partition $V(G) = S \cup T$ whether S is k-connected to T in $O(n^{3/2})$ time.*

We present the simple algorithm for Theorem 1.2 in Section 2. To check the Laman property for a plane graph G we use the algorithms of Haas et al. [8] to find a CPPT assignment and the directed plane graph $\vec{G}$ from Lemma 1.1, and then the algorithm from Theorem 1.2 to decide whether the set of interior vertices of $\vec{G}$ is 3-connected to the set of boundary vertices. This decides whether G is a plane Laman graph by Theorem 1.2 and Lemma 1.1 and has running time $O(n^{3/2})$.

A faster algorithm is obtained as follows. To search for a CPPT assignment we use an algorithm of Borradaile et al. [3]. This algorithm computes a maximum flow between multiple sources and sinks in $O(n \log^3 n)$ time. To check the connectivity condition we first use a construction similar to one of Kaplan and Nussbaum [11]. Then an algorithm due to Łącki et al. [13] is used that computes for a single source a maximum flow to each other vertex in $O(n \log^3 n)$ time. Details of this faster algorithm are presented in Section 3.

▶ **Theorem 1.3.** *The recognition problem for planar Laman graphs can be solved in $O(n \log^3 n)$.*

Checking the Laman condition for general graphs can be done in polynomial time. The fastest (but very complicated) algorithm is due to Gabow and Westermann [6] (see also [4]) and needs $O(n\sqrt{n \log n})$ time. Their algorithm is based on a characterization by means of matroid sums. There is also a very easy pebble-game algorithm that runs in $O(n^2)$ time [15]. For planar graphs Haas et al. [8] give an algorithm that computes a PPT from a Laman graph in time $O(n^{3/2})$. Their algorithm can be turned into a recognition algorithm by checking whether the derived realization is a PPT. If the original graph is not a Laman graph some parts of the drawing collapse. Such a check however requires computations with exponentially large numbers. So it depends on the model of computation if the overall algorithm runs in $O(n^{3/2})$ time. Our combinatorial algorithms avoid these subtleties.

2 Proof of Theorem 1.2

We first give some structural results. The following statement is similar to Menger's theorem.

▶ **Lemma 2.1.** *Let $k \geq 0$, G be a directed (not necessarily planar) graph, and S, $T \subseteq V(G)$ be disjoint with $|T| \geq k$. Then S is k-connected to T if and only if for each $s \in S$ and $A \subseteq V(G) \setminus \{s\}$ with $|A| = k-1$ there is a directed path from s to T not using the vertices in A.*

▶ **Lemma 2.2.** *Let G be a directed graph and let S, T, $T' \subseteq V(G)$ be disjoint. If S is k-connected to $T \cup T'$ and T' is k-connected to T, then S is k-connected to T.*

Proof. Let $s \in S$ and fix a set $A \subseteq V(G) \setminus \{s\}$ of size $k-1$. There is a directed path from s to some vertex $u \in T \cup T'$ not using vertices from A. If $u \in T$ we are done. If $u \in T'$, then there is a directed path from u to T not using vertices from A. In each case there is a directed path from s to T not using vertices from A. So S is k-connected to T by Lemma 2.1. ◀

For a single vertex we can decide in $O(kn)$ time whether it is k-connected to T as follows. A slight modification of a by now standard construction due to Ford and Fulkerson [5] gives a directed graph G' and vertices s', $t' \in V(G')$ such that $s \in V(G)$ is k-connected to T in G if and only if G' admits an s'-t'-flow with value at least k. To check whether G' admits such a flow we use at most k steps of augmentation in Ford–Fulkerson's algorithm. Since each augmentation step needs only linear time we have the following result.

▶ **Lemma 2.3.** *For each $k \in \mathbb{N}$ there is an algorithm deciding for any directed (not necessarily planar) graph G, $s \in V(G)$, and $T \subseteq V(G)$ whether s is k-connected to T in linear time.*

We call $A \subseteq V(G)$ a *separator* if removing A splits G into two (not necessarily connected) subgraphs G_1 and G_2, such that $|V(G_1)|, |V(G_2)| \leq \frac{2}{3}|V(G)|$. For every planar graph G a separator with size in $O(\sqrt{n})$ can be found in linear time [16].

Proof of Theorem 1.2. The algorithm works recursively as follows. Let A denote a separator of G of size $O(\sqrt{n})$. Use Lemma 2.3 to check for each $a \in A \cap S$ whether a is k-connected to T in G. Let G_1 and G_2 denote the two subgraphs of G separated by A (each including A). For $i = 1, 2$ let $S_i = (S \cap V(G_i)) \setminus A$ and let $T_i = (T \cap V(G_i)) \cup A$. Apply the algorithm recursively to check whether S_i is k-connected to T_i in G_i, for $i = 1, 2$.

The algorithm indeed checks whether S is k-connected to T in G since either it finds some vertex in $A \cap S$ that is not k-connected to T in G, or it is sufficient to check whether $S \setminus A$ is k-connected to $A \cup T$ by Lemma 2.2. Then it is sufficient to check G_1 and G_2 separately.

The separator can be found in linear time. Then the algorithm from Lemma 2.3 is called $O(\sqrt{n})$ times for each vertex in the separator, each call with time in $O(n)$. So the total time for each step of the recursion and hence for the whole algorithm is $O(n^{3/2})$. ◀

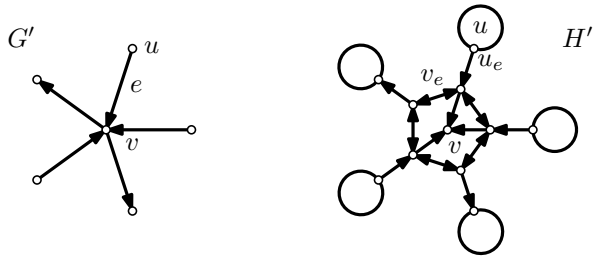


Figure 3 The modification applied to a planar directed graph G'. The modification of Kaplan and Nussbaum does not include the original vertices inside of the new cycles. We need this since all the original vertices except a fixed source are targets in our algorithm.

3 Proof of Theorem 1.3

Here we describe how to use flow algorithms for planar graphs to check whether a plane graph G admits a CPPT assignment and whether a CPPT is stretchable.

The vertex-face incidence graph of a plane graph $G = (V, E)$ with face set F is a planar bipartite graph $H = (V \cup F, E')$ where $(v, f) \in E'$ if and only if $v \in V$ is incident to $f \in F$ in G. A plane graph G admits a CPPT assignment if and only if its vertex-face incidence graph has a subgraph H' where each interior face of G has degree 3 in H', the outer face of G has degree 0 in H', and each vertex of G has degree in H' equal to its degree in G minus 1 [8]. This means that edges of H' correspond to small angles in the CPPT assignment. Via some standard techniques we can use the algorithm of Borradaile et al. [3] (which computes an integer flow) to find such an assignment in $O(n \log^3 n)$ time if it exists.

To check whether a CPPT is stretchable using the algorithm of Łącki et al. [13] for maximum flow we need the following result similar to Lemma 1.1.

▶ **Lemma 3.1.** *For each CPPT G a directed planar graph $\vec{G}$ with $|V(\vec{G})| \leq 7|V(G)|$ can be computed in linear time together with some $s \in V(\vec{G})$ and $T \subseteq V(\vec{G})$ such that G is stretchable if and only if in $\vec{G}$ for each $t \in T$ the value of a maximum s-t-flow is at least 3.*

Having this lemma our algorithm to recognize planar Laman graphs works as described in the introduction. It remains to prove Lemma 3.1. To this end we modify a construction of Kaplan and Nussbaum [11]. Consider a directed graph G and distinct $s, t \in V(G)$. Kaplan and Nussbaum construct a directed planar graph $G_{s,t}$ obtained from G by replacing each $u \in V(G) \setminus \{s, t\}$ by a cycle C_u with vertices $u_1, \ldots, u_d$, where d is the degree of u in G, such that arcs incident to u are replaced by arcs not sharing endpoints while keeping their orientation and the cyclic order around u. The edges of the cycle C_u are oriented in both directions and receive (flow) capacity 1/2. See Figure 3.

▶ **Lemma 3.2** ([11])**.** *There are k internally vertex disjoint s-t-paths in G if and only if in $G_{s,t}$ the maximum s-t-flow has value at least k.*

Proof of Lemma 3.1. Consider a CPPT G. Let G' denote a directed planar graph given by Lemma 1.1, that is, G is stretchable if and only if in G' the set of interior vertices is 3-connected to the set of boundary vertices. Let H denote the directed planar graph obtained by reversing the direction of each arc in G' and by adding a new vertex s in the outer face of G' connected by arcs sv to all boundary vertices v of G'. Then in G' the set of interior vertices T is 3-connected to the set of boundary vertices if and only if in H there are 3 internally vertex disjoint s-t-paths for each $t \in T$.

We obtain a directed planar graph H' by splitting each vertex in $V(H) \setminus \{s\}$ into a cycle as follows. Replace each arc e in H, directed from $u \neq s$ to v, by arcs $u_e v_e$ and $v_e v$, where u_e, v_e are new vertices (and distinct for all e). Replace each arc e in H, directed from s to v, by arcs sv_e, and $v_e v$, where v_e is a new vertex. Further for each vertex u in H connect the new vertices u_e by a cycle C_u in the cyclic order of the arcs e around u, where the edges are directed in both directions. Finally a (flow) capacity function is defined where all arcs on cycles C_u receive capacity $1/2$ and all other arcs capacity 1. See Figure 3.

This construction corresponds to the graph $H_{s,t}$ constructed by Kaplan and Nussbaum, except that there is not a specific target t and, additionally the original vertices from H are kept inside of the cycles together with their incoming arcs. In particular H' is planar and $V(H) \subseteq V(H')$. Consider some $t \in V(H)$. Note that each s-t-flow in H' does not use vertices from $V(H) \setminus \{s, t\}$, since these vertices do not have outgoing arcs. Hence for each $t \in V(H)$ any s-t-flow in H' corresponds to an s-t-flow in H_E (by contracting t and C_t to a single vertex). By Lemma 3.2 there are 3 internally vertex disjoint s-t-paths in H if and only if in H' the value of a maximum s-t-flow is at least 3.

Clearly $\vec{G}$ can be constructed in linear time and $|V(\vec{G})| \leq |V(G)| + 2|E(G)| + 1 \leq 7|V(G)|$. This shows that $\vec{G} = H'$ together with the set T satisfies the desired conditions. ◀

4 Conclusions and further directions

An obvious direction for future research is to search for faster or simpler algorithms recognizing (planar) Laman graphs.

Our algorithms do not provide any certificate for their correctness. This could be a Henneberg sequence [9] or a decomposition into two acyclic subgraphs [4]. We do not know how to compute either of these faster than using the algorithm of Gabow and Westermann [6].

Finally, it remains to improve the running time for nonplanar graphs. Notice that our approach heavily depends on planarity. However, it is of independent interest to see if the connectivity results for directed graphs can be extended. We can adapt the ideas presented in the first algorithm when the graph has a small separator. The running time becomes linear for graphs with separators of constant size and stays in $O(n^{3/2})$ as long as there are separators of size $O(\sqrt{n})$. Similar variants of connectivity were studied before, for instance the all-pairs reachability [7, 10], all pairs minimum cut [2], or the vertex disjoint path or Menger problem [18]. We are not aware of other related results.

Instead of asking if a graph has a representation as a pointed pseudotriangulation one can ask for other representations such as general pseudotriangulations [17, 19] or straight-line triangle representations [1]. For the latter no polynomial time algorithm is known.

References

1 Nieke Aerts and Stefan Felsner. Straight line triangle representations. *Discrete Comput. Geom.*, 57(2):257–280, 2017.

2 Glencora Borradaile, David Eppstein, Amir Nayyeri, and Christian Wulff-Nilsen. All-pairs minimum cuts in near-linear time for surface-embedded graphs. In *32nd International Symposium on Computational Geometry*, volume 51 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. 22, 16. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2016.

3 Glencora Borradaile, Philip N. Klein, Shay Mozes, Yahav Nussbaum, and Christian Wulff-Nilsen. Multiple-source multiple-sink maximum flow in directed planar graphs in near-linear time. *SIAM J. Comput.*, 46(4):1280–1303, 2017.

4 Ovidiu Daescu and Anastasia Kurdia. Towards an optimal algorithm for recognizing Laman graphs. *J. Graph Algorithms Appl.*, 13(2):219–232, 2009.

5 L. R. Ford, Jr. and D. R. Fulkerson. *Flows in networks*. Princeton University Press, Princeton, N.J., 1962.

6 Harold N. Gabow and Herbert H. Westermann. Forests, frames, and games: algorithms for matroid sums and applications. *Algorithmica*, 7(5-6):465–497, 1992.

7 Loukas Georgiadis, Daniel Graf, Giuseppe F. Italiano, Nikos Parotsidis, and Przemysław Uznański. All-pairs 2-reachability in $O(n^\omega \log n)$ time. In *44th International Colloquium on Automata, Languages, and Programming*, volume 80 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 74, 14. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2017.

8 Ruth Haas, David Orden, Günter Rote, Francisco Santos, Brigitte Servatius, Herman Servatius, Diane Souvaine, Ileana Streinu, and Walter Whiteley. Planar minimally rigid graphs and pseudo-triangulations. *Comput. Geom.*, 31(1-2):31–61, 2005.

9 Lebrecht Henneberg. Die Graphische Statik der Starren Körper. In Felix Klein and Conrad Müller, editors, *Encyklopädie der Mathematischen Wissenschaften mit Einschluss ihrer Anwendungen: Vierter Band: Mechanik*, pages 345–434. Vieweg+Teubner Verlag, Wiesbaden, 1908.

10 Jacob Holm, Eva Rotenberg, and Mikkel Thorup. Planar reachability in linear space and constant time. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science—FOCS 2015*, pages 370–389. IEEE Computer Soc., Los Alamitos, CA, 2015.

11 Haim Kaplan and Yahav Nussbaum. Maximum flow in directed planar graphs with vertex capacities. *Algorithmica*, 61(1):174–189, 2011.

12 Stephen Kobourov, Torsten Ueckerdt, and Kevin Verbeek. Combinatorial and geometric properties of planar Laman graphs. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1668–1678. SIAM, Philadelphia, PA, 2012.

13 Jakub Łącki, Yahav Nussbaum, Piotr Sankowski, and Christian Wulff-Nilsen. Single source – all sinks max flows in planar digraphs. *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 599–608, 2012.

14 Gerard Laman. On graphs and rigidity of plane skeletal structures. *J. Engrg. Math.*, 4:331–340, 1970.

15 Audrey Lee and Ileana Streinu. Pebble game algorithms and sparse graphs. *Discrete Math.*, 308(8):1425–1437, 2008.

16 Richard J. Lipton and Robert E. Tarjan. A separator theorem for planar graphs. *SIAM J. Appl. Math.*, 36(2):177–189, 1979.

17 David Orden, Francisco Santos, Brigitte Servatius, and Herman Servatius. Combinatorial pseudo-triangulations. *Discrete Math.*, 307(3-5):554–566, 2007.

18 Heike Ripphausen-Lipa, Dorothea Wagner, and Karsten Weihe. The vertex-disjoint Menger problem in planar graphs. *SIAM J. Comput.*, 26(2):331–349, 1997.

19 Günter Rote, Francisco Santos, and Ileana Streinu. Pseudo-triangulations—a survey. In *Surveys on discrete and computational geometry*, volume 453 of *Contemp. Math.*, pages 343–410. Amer. Math. Soc., Providence, RI, 2008.

20 Ileana Streinu. Pseudo-triangulations, rigidity and motion planning. *Discrete Comput. Geom.*, 34(4):587–635, 2005.

Maximum Matchings and Minimum Blocking Sets in Θ_6-Graphs

Therese Biedl[1], Ahmad Biniaz[1], Veronika Irvine[1], Kshitij Jain[2], Philipp Kindermann[3], and Anna Lubiw[1]

1 **David R. Cheriton School of Computer Science, University of Waterloo, Canada**
{biedl,virvine,alubiw}@uwaterloo.ca, ahmad.biniaz@gmail.com
2 **Borealis AI, Waterloo, Canada**
kshitij.jain.1@uwaterloo.ca
3 **Lehrstuhl für Informatik I, Universität Würzburg, Germany**
philipp.kindermann@uni-wuerzburg.de

Abstract

Θ_6-graphs are important geometric graphs that have many applications. They are equivalent to Delaunay graphs where empty equilateral triangles with a horizontal edge take the place of empty circles. We investigate lower bounds on the size of maximum matchings in these graphs. The best known lower bound is $(n-1)/3$, where n is the number of vertices of the graph. Babu et al. (2014) conjectured that any Θ_6-graph has a perfect matching (as is true for standard Delaunay graphs). Although this conjecture remains open, we improve the lower bound to $(3n-8)/7$.

We also relate the size of maximum matchings in Θ_6-graphs to the minimum size of a *blocking set*. Every edge of a Θ_6-graph on a point set P corresponds to an empty triangle that contains the endpoints of the edge but no other point of P. A *blocking set* has at least one point in each such triangle. We prove that the size of a maximum matching is at least $\beta(n)/2$ where $\beta(n)$ is the minimum, over all Θ_6-graphs with n vertices, of the minimum size of a blocking set. In the other direction, lower bounds on matchings allow us to show that $\beta(n) \geq 3n/4 - 2$.

1 Introduction

One of the many beautiful properties of Delaunay triangulations is that they always contain a perfect matching, as proved by Dillencourt [10]. This is one example of a structural property of a so-called *proximity graph*. A proximity graph is determined by a set $\mathcal{S}$ of geometric objects in the plane, such as all discs, or all axis-aligned squares. Given such a set $\mathcal{S}$ and a finite point set P, we construct a proximity graph with vertex set P and with an edge (p, q) if there is an object from $\mathcal{S}$ that contains p and q, and no other point of P in its interior. When $\mathcal{S}$ consists of all discs, then we get the Delaunay triangulation.

Our paper is about structural properties of Θ_6-graphs, which are the proximity graphs determined by equilateral triangles with a horizontal edge. More precisely, for any finite point set P, define $G^{\triangle}(P)$ to be the proximity graph of P with respect to *upward* equilateral triangles $\triangle$, define $G^{\triangledown}(P)$ to be the proximity graph of P with respect to *downward* equilateral triangles $\triangledown$, and define $G^{\hexagram}(P)$, the Θ_6-graph of P, to be their union. In particular, for two points p and q in the plane, we denote by $\triangle(p, q)$ (resp., by $\triangledown(p, q)$) the smallest upward (resp., downward) equilateral triangle that has p and q on its boundary. We say that a triangle is *empty* if it has no points of P in its interior. With these definitions, $G^{\hexagram}(P)$ has an edge between p and q if and only if $\triangle(p, q)$ is empty or $\triangledown(p, q)$ is empty, in which case we say that the edge (p, q) is *introduced* by $\triangle(p, q)$ or by $\triangledown(p, q)$.

Θ_6-graphs were first introduced by Clarkson [9] and Keil [11] as follows. Place 6 rays emanating from every point $p \in P$ at angles that are multiples of $\pi/3$ from the positive x-

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

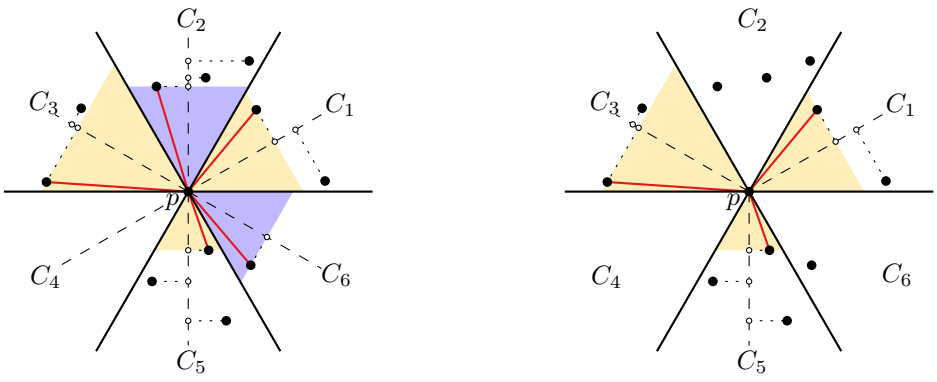


Figure 1 The construction of (a) the Θ_6-graph, and (b) the odd half-Θ_6-graph.

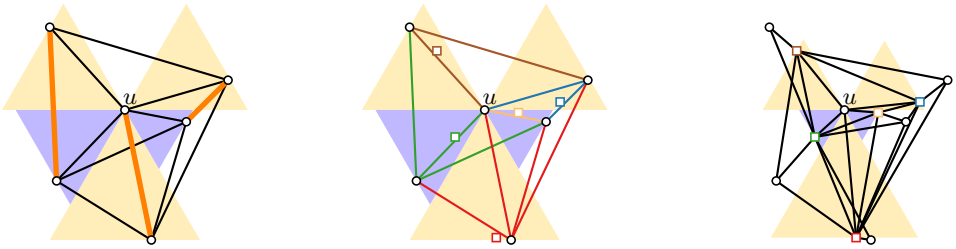


Figure 2 A Θ_6-graph on $n = 6$ points with a perfect matching and a blocking set of size 5. (a) A perfect matching. (b) A blocking set B of size $n - 1$. Edges have the same color as their blocking point. (c) $G^{\hexagram}(P \cup B)$. All edges are incident to a point of B.

axis. These rays partition the plane into 6 cones with apex p, which we label $C_1, \ldots, C_6$ in counterclockwise order starting from the positive x-axis; see Figure 1a. Add an edge from p to the *closest* point in each cone C_i, where the distance between the apex p and a point q in C_i is measured by the Euclidean distance from p to the projection of q on the bisector of C_i as depicted in Figure 1a. It is straight-forward to show both definitions of Θ_6-graphs are equivalent. The edges of $G^{\triangle}(P)$ come from the odd cones, and the edges of $G^{\triangledown}(P)$ come from the even cones, so the graphs $G^{\triangle}(P)$ and $G^{\triangledown}(P)$ are known as "half-Θ_6" graphs.

We explore two conjectures about Θ_6-graphs; see Figure 2 for an example.

▶ **Conjecture 1** (Babu et al. [3]). *Every Θ_6-graph has a perfect matching.*

The best known bound is that every Θ_6-graph on n points has a matching of size at least $\lceil (n-1)/3 \rceil$; see Babu et al. [3]. Our main result is an improvement of this lower bound:

▶ **Theorem 1.** *Every Θ_6-graph has a matching of size at least $(3n-8)/7$.*

We prove Theorem 1 using a technique that has been used for matchings in planar proximity graphs, namely the Tutte-Berge theorem, which relates the size of a maximum matching in a graph to the number of components of odd cardinality after removing some vertices. In our case, this approach is more complicated because Θ_6-graphs are not necessarily planar.

Our second main result relates the size of matchings to the size of *blocking* sets of proximity graphs, which were introduced by Aronov et al. [1]. For a proximity graph $G(P)$

defined in terms of a set of objects $\mathcal{S}$, a set B of points *blocks* $G(P)$ if B has a point inside any object from $\mathcal{S}$ that contains exactly two points of P, i.e., the set B destroys all the edges of $G(P)$, or equivalently, $G(P \cup B)$ has no edges between vertices in P. Refer to Figure 2.

For a set of points P, let $\beta(G^{\hexagram}(P))$ be the minimum size of a blocking set of $G^{\hexagram}(P)$. Let $\beta(n)$ be the minimum, over all point sets P of size n, of $\beta(G^{\hexagram}(P))$. It is known that $\beta(n) \geq \lceil (n-1)/2 \rceil$ since that already holds for $G^{\triangle}$-graphs [8]. Let $\mu(n)$ be the minimum, over all point sets P of size n, of the size of a maximum matching in $G^{\hexagram}(P)$. Conjecture 1 can hence be restated as $\mu(n) \geq \lceil (n-1)/2 \rceil$. We relate the parameters μ and β as follows:

▶ **Theorem 2.** *(a) Every Θ_6-graph has a matching of size $\beta(n)/2$, i.e., $\mu(n) \geq \beta(n)/2$.*
(b) If $\mu(n) \geq cn + d$ for some constants c, d, then $\beta(n) \geq (cn+d)/(1-c)$.

Theorem 2 has two consequences. The first is that Theorem 1 implies the following.

▶ **Corollary 3.** $\beta(n) \geq 3n/4 - 2$.

The second consequence is that Conjecture 1 is equivalent to the following conjecture.

▶ **Conjecture 2.** $\beta(n) \geq n - 1$.

Some proofs are sketched, the full proofs can be found in the full version of the paper [6].

2 Preliminaries

We assume that points are in general position and that no line passing through two points of P makes an angle of 0°, 60°, or 120° with the horizontal.

▶ **Lemma 4** (Babu et al. [3])**.** *Let P be a set of points in the plane, and let p and q be any two points in P. There is a path between p and q in $G^{\triangle}(P)$ that lies entirely in $\triangle(p,q)$. Moreover, the triangles that introduce the edges of this path also lie entirely in $\triangle(p,q)$. Analogous statements hold for $G^{\triangledown}(P)$ and $\triangledown(p,q)$.*

The next lemma has been proved in the general setting of convex-distance Delaunay graphs. We state the result for our special case. For two points p and q in the plane, define the weight function $w^{\triangle}(p,q)$ to be the scaling factor, relative to the unit triangle, of $\triangle(p,q)$.

▶ **Lemma 5** (Aurenhammer and Paulini [2], Theorem 4)**.** *The minimum spanning tree of points P with respect to the weight function $w^{\triangle}(p,q)$ is contained in $G^{\triangle}(P)$.*

A consequence of Lemma 5 (as noted by Aurenhammer and Paulini in their more general setting) is that the minimum spanning tree of points P with respect to the weight function $w^{\triangle}(p,q)$ is contained in both $G^{\triangle}(P)$ and $G^{\triangledown}(P)$, because $w^{\triangle}(p,q) = w^{\triangledown}(p,q)$.

The Tutte-Berge Matching Theorem. Let G be a graph and let S be an arbitrary subset of vertices of G. Removing S will split G into a number $\mathrm{comp}(G \setminus S)$ of connected components. Let $\mathrm{odd}(G \setminus S)$ be the number of odd components of $G \setminus S$. In 1947, Tutte [12] characterized graphs that have a perfect matching as exactly those graphs that have at most $|S|$ odd components for any subset S. In 1957, Berge [5] extended this result to a formula (today known as Tutte-Berge formula) for the size of maximum matchings in graphs. The following is an alternate way of stating this formula in terms of the number of unmatched vertices, i.e., vertices that are not matched by the matching.

▶ **Theorem 6** (Tutte-Berge formula; Berge [5])**.** *The number of unmatched vertices of a maximum matching in G is equal to the maximum over subsets $S \subseteq V$ of* $\mathrm{odd}(G \setminus S) - |S|$.

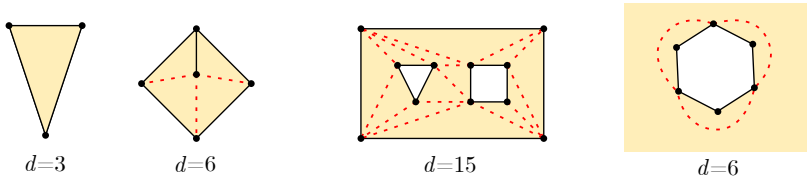


Figure 3 The notion of degree of a face.

We will use this formula in our proofs of Theorems 1 and 2. In fact, as in Dillencourt's proof [10] that Delaunay graphs have perfect matchings, we will find an upper bound on $\mathrm{comp}(G \setminus S) - |S|$, i.e., we establish a bound on the *toughness* of the graph [4]. We define the *degree* of a face as the number of triangles in a triangulation of it plus 2; see Figure 3.

3 Bounding the Size of a Matching

Let P be a set of n points in the plane. We will prove Theorem 1—that $G^{\hexagram}(P)$ contains a matching of size at least $(3n-8)/7$. It is known that all interior faces of $G^{\triangle}(P)$ and $G^{\triangledown}(P)$ are triangles [3], but their outer face might be non-convex, so we add a set $A = \{a_1, \ldots, a_6\}$ of *surrounding points* near the corners of the smallest upward and downward equilateral triangle $T^{\triangle}$ and $T^{\triangledown}$, containing all points of P; see Figure 4.

Fix a set S for which we want to bound $\mathrm{comp}(G^{\hexagram}(P) \setminus S) - |S|$, and define $S_A = S \cup A$. Pick an arbitrary representative point from every connected component of $G^{\hexagram}(P) \setminus S$, and let Q be the set of these points, so $|Q| = \mathrm{comp}(G^{\hexagram}(P) \setminus S)$.

Define $G^{\triangle}_A = G^{\triangle}(P \cup A)$ and consider its subgraph $G^{\triangle}_A[S_A]$. Note that the outer face of both $G^{\triangle}_A$ and $G^{\triangle}_A[S_A]$ is the hexagon formed by A; we add three graph edges to triangulate the outer face, so every face of $G^{\triangle}_A$ is a triangle.

Let $f^{\triangle}_d$ be the number of faces of degree d in $G^{\triangle}_A[S_A]$ that contain a point of Q. Let $f^{\triangle}_{4+} = \sum_{d \geq 4} f^{\triangle}_d$. As all faces of $G^{\triangle}_A$ are triangles, we know from Dillencourt [10, Lemma 3.4] that every face of $G^{\triangle}_A[S_A]$ contains at most one component, so at most one point of Q. Thus,

$$|Q| = f^{\triangle}_3 + f^{\triangle}_{4+} \qquad \text{and similarly} \qquad |Q| = f^{\triangledown}_3 + f^{\triangledown}_{4+}, \tag{1}$$

where $f^{\triangledown}_d$ is defined in a symmetric manner on graph $G^{\triangledown}_A[S_A]$.

Let $\mathcal{F}_d$ be the set of faces of degree d in $G^{\triangle}_A$ and observe that, since no point of Q appears in the four triangles outside the hexagon of A, we have $f^{\triangle}_3 \leq |\mathcal{F}_3| - 4$. An easy counting

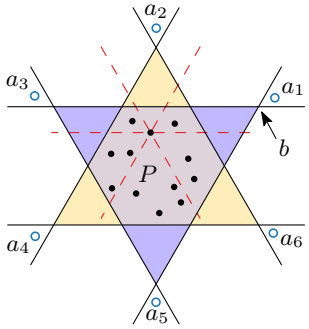


Figure 4 Augmentation of P: the shaded regions are $T^{\triangle}$ and $T^{\triangledown}$, and $A = \{a_1, \ldots, a_6\}$.

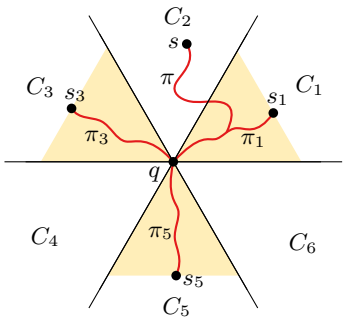


Figure 5 Illustration for the proof of Lemma 7.

argument (also used by Biedl et al. [7]) shows that $\sum_{d\geq 3}(d-2)|\mathcal{F}_d(G)| = 2|V| - 4$. Thus,

$$f_3^{\triangle} + 2f_{4+}^{\triangle} \leq \sum_{d\geq 3}(d-2)f_d^{\triangle} \leq \sum_{d\geq 3}(d-2)|\mathcal{F}_d| - 4$$
$$\leq 2|V(G_A^{\triangle})| - 4 - 4 = 2|S| + 2|A| - 8 = 2|S| + 4, \qquad (2)$$

and similarly $f_3^{\triangledown} + 2f_{4+}^{\triangledown} \leq 2|S| + 4$. The crucial insight for getting an improved matching bound is that no component can reside inside a face of degree 3 in both $G^{\triangle}$ and $G^{\triangledown}$.

▶ **Lemma 7.** *We have $f_3^{\triangle} \leq f_{4+}^{\triangledown}$ and $f_3^{\triangledown} \leq f_{4+}^{\triangle}$.*

Proof sketch. Take any point $q \in Q$. Find the shortest path π in the minimum-weight spanning tree T of $P \cup A$ that connects q to some point $s \in S_A$. Assume w.l.o.g. that s is in cone C_2; see Figure 5. Let π_1, π_3, π_5 be the paths from q to the points s_1, s_3, s_5 of S_A that are closest to q in cones C_1, C_3, C_5 that lie fully in $\triangle(q, s_i)$, respectively (exists by Lemma 4). Then, no interior vertex of π, π_1, π_3, π_5 is in S_A. Hence, s, s_1, s_3, s_5 belong to the boundary of the same face $F^{\triangle}$ of $G^{\triangle}[S_A]$ that contains q, so $F^{\triangle}$ has degree at least 4. ◀

Now we have the tools to prove an upper bound on the toughness of a Θ_6-graph.

▶ **Lemma 8.** *For any $S \subseteq P$, we have* $\mathrm{comp}(G^{\hexagram}(P) \setminus S) - |S| \leq (n+16)/7$.

Proof. Recall that we fixed a set Q of points in $P \setminus S$ with $|Q| = \mathrm{comp}(G^{\hexagram}(P) \setminus S)$. So $n = |P| \geq |S| + |Q|$. Combining this with the above inequalities, we get

$$\begin{aligned}
7\left(\mathrm{comp}(G^{\hexagram}(P) \setminus S) - |S|\right) &\leq 7|Q| - 7|S| + (n - |Q| - |S|) = n + 3|Q| + 3|Q| - 8|S| \\
&= n + 3\left(f_3^{\triangle} + f_{4+}^{\triangle}\right) + 3\left(f_3^{\triangledown} + f_{4+}^{\triangledown}\right) - 8|S| \qquad \text{(by (1))} \\
&\leq n + 2f_3^{\triangle} + 4f_{4+}^{\triangle} + 2f_3^{\triangledown} + 4f_{4+}^{\triangledown} - 8|S| \qquad \text{(by Lemma 7)} \\
&\leq n + (4|S| + 8) + (4|S| + 8) - 8|S| = n + 16. \qquad \text{(by (2))}
\end{aligned}$$ ◀

Therefore, $\mathrm{odd}(G^{\hexagram}(P) \setminus S) - |S| \leq \mathrm{comp}(G^{\hexagram}(P) \setminus S) - |S| \leq (n+16)/7$. In consequence of the Tutte-Berge formula, therefore any maximum matching M of $G^{\hexagram}(P)$ has at least $(6n-16)/7$ matched vertices and $|M| \geq (3n-8)/7$. This completes the proof of Theorem 1.

4 The Relationship Between Blocking Sets and Matchings

In this section, we prove Theorem 2—that a lower bound on the blocking size function $\beta(n)$ implies a lower bound on the size $\mu(n)$ of a maximum matching, and vice versa.

▶ **Lemma 9.** *For any $n \geq 1$, we have $\beta(n+1) \leq \beta(n) + 1$.*

Proof sketch. Consider a set P with n points such that $\beta(n) = \beta(G^{\hexagram}(P))$. Consider the points a_1 and b depicted in Figure 4. We can block $G^{\hexagram}(P \cup \{a_1\})$ by using a minimum blocking set B of $G^{\hexagram}(P)$ and adding b to it, so $\beta(n+1) \leq \beta(G^{\hexagram}(P \cup \{a_1\})) \leq \beta(n)+1$. ◀

Since $\beta(1) = 0$, this lemma also shows that $\beta(n) \leq n-1$, i.e., that Conjecture 2 is tight.

▶ **Theorem 2.** *(a) Every Θ_6-graph has a matching of size $\beta(n)/2$, i.e., $\mu(n) \geq \beta(n)/2$.*

Proof. Fix a point set P, an arbitrary set $S \subseteq P$, one representative point in each connected component of $G^{\hexagram}(P) \setminus S$, and let Q be the set of these points. Let (q_1, q_2) be an edge in $G^{\hexagram}(Q)$ introduced by $\triangle(q_1, q_2)$. By Lemma 4, there is a path π between q_1 and q_2 in $G^{\triangle}(P)$ that is fully contained in $\triangle(q_1, q_2)$. Since q_1 and q_2 belong to different components of $G^{\hexagram}(P) \setminus S$, at least one point of π belongs to S. Thus, S blocks $G^{\hexagram}(Q)$, and $|S| \geq \beta(|Q|)$. Further, $\beta(n) \leq \beta(|Q|) + n - |Q|$ by Lemma 9 since $|Q| \leq n$. By Theorem 6, it follows that

$$\mu(G^{\hexagram}(P)) \geq \frac{n - (|Q| - |S|)}{2} \geq \frac{n - (|Q| - \beta(|Q|))}{2} \geq \frac{n - (n - \beta(n))}{2} = \frac{\beta(n)}{2}.$$ ◀

In particular, if $\beta(n) \geq n-1$, then $\mu(n) \geq \beta(n)/2 \geq (n-1)/2$, so by integrality $\mu(n) \geq \lceil (n-1)/2 \rceil$. In other words, Conjecture 2 implies Conjecture 1.

▶ **Theorem 2.** *(b) If $\mu(n) \geq cn + d$ for some constants c, d, then $\beta(n) \geq (cn+d)/(1-c)$.*

Proof. Let P be a set of n points such that $\beta(G^{\hexagram}(P)) = \beta(n) = b$, and let B be a minimum blocking set of $G^{\hexagram}(P)$ of size b. Let M be a matching of size at least $\mu(b+n) \geq cb + cn + d$ in $G^{\hexagram}(P \cup B)$. Since P is an independent set in $G^{\hexagram}(P \cup B)$, it can contain at most one endpoint of each edge in M, as well as all unmatched points, so

$$n = |P| \leq |M| + (n + b - 2|M|) \leq n + b - (cb + cn + d).$$

Solving for b gives $\beta(n) = b \geq (cn+d)/(1-c)$. ◀

In particular, if Conjecture 1 holds, then $\mu(n) \geq (n-1)/2$. Hence, $c = -d = 1/2$, so $\beta(n) \geq 2(n-1)/2 = n-1$ and Conjecture 2 holds. So Conjecture 1 implies Conjecture 2. As a second consequence, we know that $(3n-8)/7$ is a valid lower bound on $\mu(n)$ by Theorem 1, therefore (with $c = 3/7$) we have $\beta(n) \geq 7/4 \cdot (3n-8)/7 = 3n/4 - 2$, proving Corollary 3.

Acknowledgements. This work was done by a University of Waterloo problem solving group. We thank Alexi Turcotte and Anurag Murty Naredla, for helpful discussions.

References

1 B. Aronov, M. Dulieu, and F. Hurtado. Witness (Delaunay) graphs. *Computational Geometry*, 44(6-7):329–344, 2011.

2 F. Aurenhammer and G. Paulini. On shape Delaunay tessellations. *Information Processing Letters*, 114(10):535–541, 2014.

3 J. Babu, A. Biniaz, A. Maheshwari, and M. H. M. Smid. Fixed-orientation equilateral triangle matching of point sets. *Theoretical Computer Science*, 555:55–70, 2014. Also in *WALCOM'13*.

4 D. Bauer, H. Broersma, and E. Schmeichel. Toughness in graphs—a survey. *Graphs and Combinatorics*, 22(1):1–35, 2006.

5 C. Berge. Sur le couplage maximum d'un graphe. *Comptes Rendus de l'Académie des Sciences, Paris*, 247:258–259, 1958.

6 T. Biedl, A. Biniaz, V. Irvine, K. Jain, P. Kindermann, and A. Lubiw. Maximum matchings and minimum blocking sets in θ_6-graphs. *arXiv:1901.01476*, 2018.

7 T. Biedl, E. D. Demaine, C. A. Duncan, R. Fleischer, and S. G. Kobourov. Tight bounds on maximal and maximum matchings. *Discrete Mathematics*, 285(1-3):7–15, 2004. Also in *ISAAC'01*.

8 A. Biniaz, A. Maheshwari, and M. H. M. Smid. Higher-order triangular-distance Delaunay graphs: Graph-theoretical properties. *Computational Geometry: Theory and Applications*, 48(9):646–660, 2015. Also in *CALDAM'15*.

9 K. L. Clarkson. Approximation algorithms for shortest path motion planning. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing* (*STOC*), pages 56–65, 1987.

10 M. B. Dillencourt. Toughness and Delaunay triangulations. *Discrete and Computational Geometry*, 5:575–601, 1990.

11 J. M. Keil. Approximating the complete Euclidean graph. In *Proceedings of the 1st Scandinavian Workshop on Algorithm Theory* (*SWAT*), pages 208–213, 1988.

12 W. T. Tutte. The factorization of linear graphs. *Journal of the London Mathematical Society*, 22:107–111, 1947.

Rigid Graphs that are Movable*

Georg Grasegger[1], Jan Legerský[2], and Josef Schicho[2]

1 **Johann Radon Institute for Computational and Applied Mathematics (RICAM), Austrian Academy of Sciences**
georg.grasegger@ricam.oeaw.ac.at

2 **Research Institute for Symbolic Computation (RISC), Johannes Kepler University Linz**
jan.legersky@risc.jku.at, josef.schicho@risc.jku.at

Abstract

A graph is called movable if there exists a proper flexible labeling, i.e., an edge labeling such that there are infinitely many injective realizations of the graph in the plane, counted modulo rigid motions, such that the distances between adjacent vertices equal the labels. Of special interest is the class of generically rigid graphs that are movable due to a non-generic proper flexible labeling. We introduce two methods for investigating possible proper flexible labelings. The first one is based on restrictions to 4-cycles and gives an easy classification of all but one non-bipartite 6 and 7-vertex graphs in the class. Using our second method, we prove that every proper flexible labeling of this one graph forces the vertices in its only 3-cycle to be collinear.

1 Introduction

In Rigidity Theory, realizations of a graph in $\mathbb{R}^2$ are required to be such that the distances of adjacent vertices are equal to a given labeling of edges by positive real numbers. Such a labeling is called *(proper) flexible* if the number of (injective) realizations, counted modulo rigid transformations, is infinite. Otherwise, the labeling is called rigid. We call a graph movable if there is a proper flexible labeling.

A result of Pollaczek-Geiringer [6], rediscovered by Laman [5], shows that a graph is *generically rigid*, i.e., a generic realization defines a rigid labeling, if and only if the graph contains a Laman subgraph with the same set of vertices. A graph $G = (V_G, E_G)$ is called Laman if $|E_G| = 2|V_G| - 3$, and $|E_H| \leq 2|V_H| - 3$ for all subgraphs H of G. Hence, every graph that is not spanned by a Laman graph is movable.

A natural question is which generically rigid graphs are movable, due to a non-generic proper flexible labeling. For instance, two ways of making the bipartite Laman graph $K_{3,3}$ movable were given by Dixon more than one hundred years ago [2, 9, 7], and it was proven much later in 2007 that these give all proper flexible labelings [8]. In [3], we provide a combinatorial characterization of existence of a flexible labeling, not necessarily proper: it exists if and only if the graph has a so called *NAC-coloring* (see Figure 1). In [4] we classified all movable graphs up to 8 vertices.

The movable graphs up to 7 vertices with special properties are listed in Figure 2. They do not have a degree two vertex, they are spanned by a Laman graph and they are maximal with respect of being subgraph of a movable graph with the same number of vertices. These are the interesting ones since a graph with a degree two vertex v is movable if and only if the graph with v removed is movable; and a spanning subgraph of a movable graph is movable.

* 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, W1214-N15 (project DK9).

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

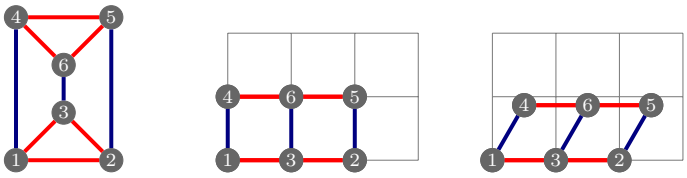


Figure 1 Motion from a NAC-coloring.

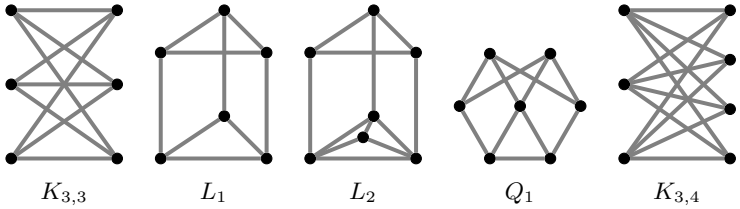


Figure 2 Movable graphs with a spanning Laman subgraph and at most 7 vertices.

The classification of proper flexible labelings of $K_{3,3}$ and $K_{3,4}$ is known. In this paper, we introduce methods allowing a step towards classification for the other graphs. The first one is based on restriction to 4-cycles and gives an easy proof that L_1 and L_2 have only one type of proper flexible labelings (Section 3). The proper flexible labeling provided in [4] makes the vertices of the triangle in Q_1 collinear. Our second method allows to show that this is always the case (Section 4). The method uses leading coefficients of certain Laurent series and is based on tropicalization ideas presented in [1].

2 Preliminaries

We recall the concept of a NAC-coloring, which is a basic object for our further considerations.

▶ **Definition 2.1.** Let G be a graph. A coloring of edges $\delta\colon E_G \to \{\text{blue, red}\}$ is called a *NAC-coloring*, if it is surjective and for every cycle C in G, either all edges of C have the same color, or C contains at least 2 edges in each color. If NAC-colorings $\delta, \overline{\delta}$ of G are such that $\delta(e) = \text{blue} \iff \overline{\delta}(e) = \text{red}$ for all $e \in E_G$, then they are called *conjugated*.

We remark that NAC stands for "No Almost Cycle", i.e., there is no cycle with all but one edges having the same color. Next, we summarize the concepts from Rigidity Theory.

▶ **Definition 2.2.** Let G be a graph such that $|E_G| \geq 1$ and let $\lambda\colon E_G \to \mathbb{R}_+$ be an edge labeling of G. A map $\rho = (\rho_x, \rho_y)\colon V_G \to \mathbb{R}^2$ is a *realization of G compatible with λ* if $\|\rho(u) - \rho(v)\| = \lambda(uv)$ for all edges $uv \in E_G$. The labeling λ is called *(proper) flexible* if the number of (injective) realizations of G compatible with λ up to direct Euclidean isometries is infinite. A graph is called *movable* if it has a proper flexible labeling.

We fix an edge $\bar{u}\bar{v}$ by setting $x_{\bar{u}} = y_{\bar{u}} = y_{\bar{v}} = 0$ and $x_{\bar{v}} = \lambda_{\bar{u}\bar{v}}$ for removing rotations and translations (reflection on x-axis is kept). Then the realizations are the solutions of the system of equations for lengths $\lambda_{uv} = \lambda(uv)$ and coordinates (x_u, y_u) for $u \in V_G$ given by

$$(x_u - x_v)^2 + (y_u - y_v)^2 = \lambda_{uv}^2 \quad \text{for all } uv \in E_G \setminus \{\bar{u}\bar{v}\}. \tag{1}$$

For working with function fields, we consider the irreducible components of the solution set.

▶ **Definition 2.3.** Let λ be a flexible labeling of G. We say that $\mathcal{C}$ is an *algebraic motion of* (G,λ), if it is an irreducible algebraic curve of realizations compatible with λ, such that $\rho(\bar{u}) = (0,0)$ and $\rho(\bar{v}) = (\lambda_{\bar{u}\bar{v}}, 0)$ for all $\rho \in \mathcal{C}$.

In order to link an algebraic motion with a NAC-coloring, we define the following functions:

▶ **Definition 2.4.** Let λ be a flexible labeling of a graph G. Let $F(\mathcal{C})$ be the complex function field of an algebraic motion $\mathcal{C}$ of (G,λ). For every $u,v \in V_G$ such that $uv \in E_G$, we define $W_{u,v}, Z_{u,v} \in F(\mathcal{C})$ by $W_{u,v} = (x_v - x_u) + i(y_v - y_u)$ and $Z_{u,v} = (x_v - x_u) - i(y_v - y_u)$.

Using (1), we have $W_{\bar{u},\bar{v}} = \lambda_{\bar{u}\bar{v}}, Z_{\bar{u},\bar{v}} = \lambda_{\bar{u}\bar{v}}$ and $W_{u,v}Z_{u,v} = \lambda_{uv}^2$ for all $uv \in E_G$. Moreover, the following equations hold for every cycle $(u_0, u_1, \dots, u_n, u_{n+1} = u_0)$ in G:

$$\sum_{i=0}^{n} W_{u_i,u_{i+1}} = 0 \quad \text{and} \quad \sum_{i=0}^{n} Z_{u_i,u_{i+1}} = 0\,. \tag{2}$$

▶ **Definition 2.5.** Let $\mathcal{C}$ be an algebraic motion of (G,λ). A NAC-coloring δ of G is called *active* if there exists a valuation ν of $F(\mathcal{C})$ and $\alpha \in \mathbb{Q}$ such that $\delta(uv) = \text{red}$ if and only if $\nu(W_{u,v}) > \alpha$ for all $uv \in E_G$. The set of all active NAC-colorings is denoted by $\text{NAC}_G(\mathcal{C})$.

We remark that $\text{NAC}_G(\mathcal{C})$ is closed under conjugation and independent of the fixed edge [4].

3 Restriction to 4-cycles

One way of investigating proper flexible labelings is to look at restrictions to 4-cycle subgraphs, since the active NAC-colorings of an algebraic motion of (C_4,λ) can be described:

▶ **Lemma 3.1.** *Let $\mathcal{C}$ be an algebraic motion of a 4-cycle graph C_4 with a labeling λ. Table 1 summarizes $\text{NAC}_{C_4}(\mathcal{C})$ depending on λ using the notation depicted on Figure 3.*

Proof. Explicit computation — solving the system of equations for $W_{u,v}$ and $Z_{u,v}$ and determining valuations giving active NAC-colorings. ◀

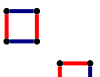
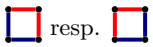
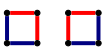
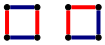
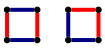
Quadrilateral	Motion	active NAC-colorings		Equations
Rhombus	parallel		O	$\left\{\begin{array}{r}\lambda_{12} = \lambda_{23} = \\ = \lambda_{34} = \lambda_{14}\end{array}\right\}$
	degenerate #1 resp. #2	 resp.	L resp. R	
Parallelogram	parallel		O	$\left\{\begin{array}{l}\lambda_{12} = \lambda_{34},\\ \lambda_{23} = \lambda_{14}\end{array}\right\}$
	antiparallel		L, R	
Deltoid 1	nondegenerate		O, R	$\left\{\begin{array}{l}\lambda_{12} = \lambda_{14},\\ \lambda_{23} = \lambda_{34}\end{array}\right\}$
	degenerate		L	
Deltoid 2	nondegenerate		O, L	$\left\{\begin{array}{l}\lambda_{12} = \lambda_{23},\\ \lambda_{34} = \lambda_{14}\end{array}\right\}$
	degenerate		R	
General			O, L, R	otherwise

Table 1 Active NAC-colorings of possible motions of a (C_4,λ).

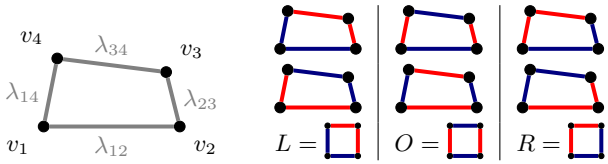


Figure 3 Labeling of the 4-cycle C_4 and notation for conjugated NAC-colorings.

Assuming an algebraic motion $\mathcal{C}$ of a graph G, the active NAC-colorings of the projection of $\mathcal{C}$ to a 4-cycle subgraph of G are precisely the restrictions of the active NAC-colorings of $\mathcal{C}$. It is well known that L_1 and L_2 are movable by making the vertical edges in Figure 2 parallel and same lengths. Looking at 4-cycles gives easily that this is the only option:

▶ **Corollary 3.2.** *If λ is a proper flexible labeling of L_1, resp. L_2, then every 4-cycle that is colored non-trivially by δ_1, resp. δ_2, (see Figure 4) is a parallelogram.*

Proof. Since δ_i is the only NAC-coloring of L_i modulo conjugation, it is the only active NAC-coloring in every algebraic motion. The restriction of δ_i to each non-trivially colored 4-cycle is of type O ( ). According to Table 1 it must be in a parallel motion. ◀

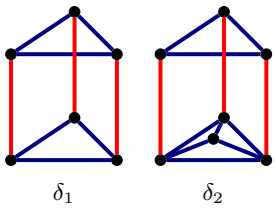


Figure 4 The only NAC-colorings of L_1 and L_2 modulo conjugation.

4 Leading coefficients system

If a movable graph G is spanned by a Laman graph, the edge lengths λ_{uv} must be non-generic. We introduce a method deriving some algebraic equation(s) for λ_{uv}, which must be satisfied if a NAC-coloring δ is active due to a valuation ν under a certain assumption.

Let an edge $\bar{u}\bar{v}$ be fixed. Then, $\nu(W_{\bar{u},\bar{v}}) = 0$. We can assume that $\delta(\bar{u}\bar{v}) =$ blue, otherwise we replace δ by its conjugated NAC-coloring. There must be a red edge $\hat{u}\hat{v}$ with $\nu(W_{\hat{u},\hat{v}}) = \alpha > 0$. We assume that ν yields no other active NAC-coloring besides δ. This assumption implies that $\{\nu(W_{u,v})\colon uv \in E_G\} = \{0, \alpha\}$. Notice that if ν yields another active NAC-coloring δ', then the set $\{(\delta(e), \delta'(e))\colon e \in E_{Q_1}\}$ has 3 elements.

There are Laurent series parametrizations of $W_{u,v}$ and $Z_{u,v}$ such that $\operatorname{ord}(W_{u,v}) = \nu(W_{u,v})$ and $\operatorname{ord}(Z_{u,v}) = \nu(Z_{u,v})$. Since $\nu(W_{\hat{u},\hat{v}}) = \alpha > 0$, we can reparametrize so that $W_{\hat{u},\hat{v}} = \lambda_{\hat{u}\hat{v}}t$. Hence, there is $\varepsilon > 0$ such that the parametrizations are

$$
\begin{aligned}
&W_{\bar{u},\bar{v}} = \lambda_{\bar{u}\bar{v}}\,, \quad Z_{\bar{u},\bar{v}} = \lambda_{\bar{u}\bar{v}}\,, \quad W_{\hat{u},\hat{v}} = \lambda_{\hat{u}\hat{v}}t\,, \quad Z_{\hat{u},\hat{v}} = \lambda_{\hat{u}\hat{v}}t^{-1}\,,\\
&W_{u,v} = w_{uv} + O(t^{\varepsilon})\,, \qquad Z_{u,v} = z_{uv} + O(t^{\varepsilon}) \qquad \text{for } uv \in E_G \setminus \{\bar{u}\bar{v}\},\ \delta(uv) = \text{blue}\,,\\
&W_{u,v} = w_{uv}t + O(t^{1+\varepsilon})\,, \quad Z_{u,v} = z_{uv}t^{-1} + O(t^{-1+\varepsilon}) \qquad \text{for } uv \in E_G \setminus \{\hat{u}\hat{v}\},\ \delta(uv) = \text{red}\,.
\end{aligned}
$$

The constraints from edge lengths give $w_{uv}z_{uv} = \lambda_{uv}^2$ for all $uv \in E_G \setminus \{\bar{u}\bar{v}, \hat{u}\hat{v}\}$.

For every cycle $C = (u_0, u_1, \ldots, u_n, u_{n+1} = u_0)$ in G, the first equation in (2) gives

$$\sum_{\substack{i\in\{0,\ldots,n\}\\ \delta(u_iu_{i+1})=\text{red}}} (w_{u_iu_{i+1}}t + O(t^{1+\varepsilon})) + \sum_{\substack{i\in\{0,\ldots,n\}\\ \delta(u_iu_{i+1})=\text{blue}}} (w_{u_iu_{i+1}} + O(t^{\varepsilon})) = 0\,.$$

Comparing leading coefficients gives $\sum w_{u_iu_{i+1}} = 0$, where the sum is over all $i \in \{0,\ldots,n\}$ such that $\delta(u_iu_{i+1}) =$ blue if there exists a blue edge, or over all edges in C otherwise. Similarly, we obtain an equation in z_{uv}'s from the second equation in (2).

From the obtained equations from all cycles and edge lengths, we eliminate w_{uv} and z_{uv} for all $uv \in E_G \setminus \{\bar{u}\bar{v}, \hat{u}\hat{v}\}$ using Gröbner basis. If the graph G is spanned by a Laman graph, we expect to get some algebraic equation(s) in λ_{uv} for $uv \in E_G$. We use the described procedure for the graph Q_1. The NAC-colorings of Q_1, modulo conjugation, are in Figure 5.

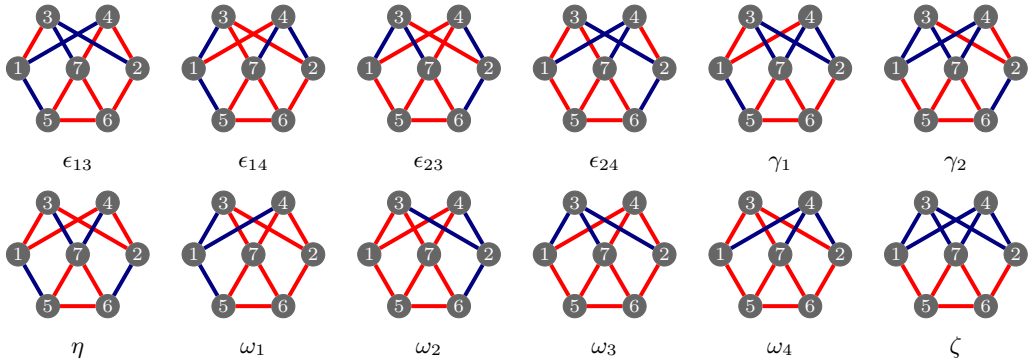


Figure 5 NAC-colorings of the graph Q_1.

▶ **Lemma 4.1.** *Let $\mathcal{C}$ be an algebraic motion of (Q_1, λ). If $\eta \in \mathrm{NAC}_{Q_1}(\mathcal{C})$, resp. $\epsilon_{13} \in \mathrm{NAC}_{Q_1}(\mathcal{C})$, then the vertices of the triangle (5,6,7) are collinear, or $\lambda_{24} = \lambda_{23}$ and $\lambda_{14} = \lambda_{13}$, resp. $\lambda_{26} = \lambda_{67}$ and $\lambda_{24} = \lambda_{47}$.*

Proof. The procedure described at the beginning of this section can be used for η, since for every other NAC-coloring δ of Q_1, the set $\{(\eta(e), \delta(e)) \colon e \in E_{Q_1}\}$ has 4 elements. The equation $\lambda_{57}^2 r^2 + \lambda_{67}^2 s^2 + (\lambda_{56}^2 - \lambda_{57}^2 - \lambda_{67}^2)rs = 0$ is obtained, where $r = \lambda_{24}^2 - \lambda_{23}^2$ and $s = \lambda_{14}^2 - \lambda_{13}^2$. Considering the equation as a polynomial in r, the discriminant is $(\lambda_{56} + \lambda_{57} + \lambda_{67})(\lambda_{56} + \lambda_{57} - \lambda_{67})(\lambda_{56} - \lambda_{57} + \lambda_{67})(\lambda_{56} - \lambda_{57} - \lambda_{67})s^2$. But this is always non-positive from the triangle inequality. Hence, the triangle must be degenerate, or $s = 0$. If $s = 0$, then also $r = 0$ and the statement follows. The proof for ϵ_{13} is similar. ◀

We prove the following lemma by combining the previous one with the restrictions to 4-cycles.

▶ **Lemma 4.2.** *Let $\mathcal{C}$ be an algebraic motion of (Q_1, λ) such that λ is a proper flexible labeling and the vertices 5,6,7 are not collinear. If $\epsilon_{13} \in \mathrm{NAC}_{Q_1}(\mathcal{C})$, then $\epsilon_{14}, \gamma_1, \gamma_2, \omega_1, \omega_2, \omega_4, \zeta \notin \mathrm{NAC}_{Q_1}(\mathcal{C})$, either $\epsilon_{23} \in \mathrm{NAC}_{Q_1}(\mathcal{C})$ or $\epsilon_{24} \in \mathrm{NAC}_{Q_1}(\mathcal{C})$, and $\eta \in \mathrm{NAC}_{Q_1}(\mathcal{C})$.*

Proof. By the assumption and Lemma 4.1, if $\epsilon_{13} \in \mathrm{NAC}_{Q_1}(\mathcal{C})$, then $\lambda_{26} = \lambda_{67}$ and $\lambda_{24} = \lambda_{47}$. But then the 4-cycle (2,4,7,6) is a deltoid or rhombus. Hence, the restriction of any active NAC-coloring to (2,4,7,6) cannot be of type R by Lemma 3.1, i.e., $\epsilon_{14}, \gamma_1, \omega_4, \zeta \notin \mathrm{NAC}_{Q_1}(\mathcal{C})$ by Table 2. Since the 4-cycle (2,3,7,4) cannot be an antiparallelogram, there must be an active NAC-coloring whose restriction is of type O, namely, $\epsilon_{23} \in \mathrm{NAC}_{Q_1}(\mathcal{C})$ or $\epsilon_{24} \in \mathrm{NAC}_{Q_1}(\mathcal{C})$. Since ϵ_{13} excludes ϵ_{14} to be active, ϵ_{23} excludes ϵ_{24} by graph symmetry. Therefore, either

$\epsilon_{23} \in \mathrm{NAC}_{Q_1}(\mathcal{C})$ or $\epsilon_{24} \in \mathrm{NAC}_{Q_1}(\mathcal{C})$. By the symmetric approach to the fact that ϵ_{13} excludes γ_1, we also get that both ϵ_{23} and ϵ_{24} prohibit γ_2 to be active. Since the 4-cycle (2,4,7,6), resp. (2,3,7,6), is not an antiparallelogram, there must be an active NAC-coloring restricting to O, namely ϵ_{24} or η, resp. ϵ_{23} or η (γ_2 is already excluded). In the combination with the previous, we can conclude that $\eta \in \mathrm{NAC}_{Q_1}(\mathcal{C})$. Therefore, $\lambda_{24} = \lambda_{23}$ and $\lambda_{14} = \lambda_{13}$ by Lemma 4.1. This shows that the 4-cycle (1,3,2,4) is a deltoid or rhombus which prohibits L. Thus, $\omega_1, \omega_2 \notin \mathrm{NAC}_{Q_1}(\mathcal{C})$. ◀

4-cycle	ϵ_{13}	ϵ_{14}	ϵ_{23}	ϵ_{24}	γ_1	γ_2	η	ω_1	ω_2	ω_3	ω_4	ζ
$(1,3,2,4)$	O	O	O	O	L	L	S	L	L	R	R	S
$(1,3,7,4)$	O	O	R	R	L	S	L	L	S	R	R	S
$(2,3,7,4)$	R	R	O	O	S	L	L	S	L	R	R	S
$(1,3,7,5)$	O	L	R	S	O	R	O	L	S	R	S	R
$(1,4,7,5)$	L	O	S	R	O	R	O	L	S	S	R	R
$(2,3,7,6)$	R	S	O	L	R	O	O	S	L	R	S	R
$(2,4,7,6)$	S	R	L	O	R	O	O	S	L	S	R	R

Table 2 Types of NAC-colorings of Q_1 restricted to 4-cycles using the notation from Figure 3 and S meaning all edges have the same color.

We conclude that the triangle (5,6,7) in Q_1 is always degenerate.

▶ **Theorem 4.3.** *If $\mathcal{C}$ is an algebraic motion of Q_1 with infinitely many injective realization, then the vertices 5,6 and 7 are always collinear.*

Proof. If no ϵ_{ij} is active, then the 4-cycles (1,3,2,4), (1,3,7,4) and (2,3,7,4) are all antiparallelograms. But this is not possible for injective realizations. Hence, by symmetry we can assume w.l.o.g. that ϵ_{13} is active. Suppose by contradiction that the triangle (5,6,7) is not degenerated. By Lemma 4.2 (and its symmetric version for ϵ_{24}), the only possibilities for $\mathrm{NAC}_{Q_1}(\mathcal{C})$ are $\{\epsilon_{13}, \epsilon_{23}, \eta\}$, $\{\epsilon_{13}, \epsilon_{23}, \eta, \omega_3\}$ and $\{\epsilon_{13}, \epsilon_{24}, \eta\}$. By careful determination of the types of all 4-cycles for each of these sets of active NAC-colorings, the equality of lengths always contradicts injective realizations. For instance for $\mathrm{NAC}_{Q_1}(\mathcal{C}) = \{\epsilon_{13}, \epsilon_{23}, \eta\}$, all edges among 1,2,3,4 and 7 would have to have the same lengths, which is not possible. ◀

References

1 J. Capco, M. Gallet, G. Grasegger, C. Koutschan, N. Lubbes, and J. Schicho. The number of realizations of a laman graph. *SIAM Journal on Applied Algebra and Geometry*, 2(1):94–125, 2018. doi:10.1137/17M1118312.

2 A. C. Dixon. On certain deformable frameworks. *Messenger*, 29(2):1–21, 1899.

3 G. Grasegger, J. Legerský, and J. Schicho. Graphs with Flexible Labelings. *Discrete & Computational Geometry*, 2018. doi:10.1007/s00454-018-0026-9.

4 G. Grasegger, J. Legerský, and J. Schicho. Graphs with flexible labelings allowing injective realizations, 2018. https://arxiv.org/abs/1811.06709.

5 G. Laman. On graphs and rigidity of plane skeletal structures. *Journal of Engineering Mathematics*, 4:331–340, 1970. doi:10.1007/BF01534980.

6 H. Pollaczek-Geiringer. Über die Gliederung ebener Fachwerke. *Zeitschrift für Angewandte Mathematik und Mechanik (ZAMM)*, 7:58–72, 1927. doi:10.1002/zamm.19270070107.

7 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.

8 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.

9 W. Wunderlich. On deformable nine-bar linkages with six triple joints. *Indagationes Mathematicae (Proceedings)*, 79(3):257–262, 1976.

Reliable Geometric Spanners

Kevin Buchin[1], Sariel Har-Peled[2], and Dániel Oláh[*1]

1 Department of Mathematics and Computing Science, TU Eindhoven
2 Department of Computer Science, University of Illinois

Abstract

We show how to construct a $(1+\varepsilon)$-spanner over a set P of n points in $\mathbb{R}^d$ that is resilient to a catastrophic failure of nodes. Specifically, for prescribed parameters $\vartheta, \varepsilon \in (0,1)$, the computed spanner G has $\mathcal{O}\big(\varepsilon^{-7d}\log^7\varepsilon^{-1}\cdot\vartheta^{-6}n\log n(\log\log n)^6\big)$ edges. Furthermore, for *any* k, and *any* deleted set $B\subseteq P$ of k points, the residual graph $G\setminus B$ is a $(1+\varepsilon)$-spanner for all the points of P except for $(1+\vartheta)k$ of them. No previous constructions, beyond the trivial clique with $\mathcal{O}(n^2)$ edges, were known such that only a tiny additional fraction (i.e., ϑ) lose their distance preserving connectivity.

1 Introduction

Spanners. A Euclidean graph is a graph whose vertices are points in $\mathbb{R}^d$ and the edges are weighted by the Euclidean distance between their endpoints. Let $G=(P,E)$ be a Euclidean graph and $p,q\in P$ be two vertices of G. For a parameter $t\geq 1$, a path between p and q in G is a ***t*-path** if the length of the path is at most $t\,\|p-q\|$, where $\|p-q\|$ is the Euclidean distance between p and q. The graph G is a ***t*-spanner** of P if there is a t-path between any pair of points $p,q\in P$. We denote the length of the shortest path between $p,q\in P$ in the graph G by $\mathsf{d}(p,q)$.

Spanners have been studied extensively. The main goal in spanner constructions is to have small *size*, that is, to use as few edges as possible. Other desirable properties are low degrees [1, 8, 15], low weight [5, 10], low diameter [2, 3] or to be resistant against failures [6, 11, 12, 13]. The book by Narasimhan and Smid [14] gives a comprehensive overview.

Robustness. In this paper, our goal is to construct spanners that are robust according to the notion introduced by Bose *et al.* [6]. Intuitively, a spanner is robust if the deletion of k vertices only harms a few other vertices. Formally, a graph G is an $f(k)$-robust t-spanner, for some positive monotone function f, if for any set B of k vertices deleted in the graph, the remaining graph $G\setminus B$ is still a t-spanner for at least $n-f(k)$ of the vertices. Note, that the graph $G\setminus B$ has $n-k$ vertices – namely, there are at most $\mathcal{L}(k)=f(k)-k$ additional vertices that no longer have good connectivity to the remaining graph. The quantity $\mathcal{L}(k)$ is the ***loss***. We are interested in minimizing the loss.

The natural question is how many edges are needed to achieve a certain robustness (since the clique has the desired property). That is, for a given parameter t and function f, what is the minimal size that is needed to obtain an $f(k)$-robust t-spanner on any set of n points.

A priori it is not clear that such a sparse graph should exist (for t a constant) for a point set in $\mathbb{R}^d$, since the robustness property looks quite strong. Surprisingly, Bose *et al.* [6] showed that one can construct a $\mathcal{O}(k^2)$-robust $\mathcal{O}(1)$-spanner with $\mathcal{O}(n\log n)$ edges. Bose *et al.* [6] proved various other bounds in the same vein on the size for one-dimensional and higher-dimensional point set. Their most closely related result is that for the one-dimensional

* Supported by the Netherlands Organisation for Scientific Research (NWO) through Gravitation-grant NETWORKS-024.002.003.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

point set $P = \{1, 2, \ldots, n\}$ and for any $t \geq 1$ at least $\Omega(n \log n)$ edges are needed to construct an $\mathcal{O}(k)$-robust t-spanner.

ϑ-reliable spanners. We are interested in building spanners where the loss is only fractional. Specifically, given a parameter ϑ, we consider the function $f(k) = (1+\vartheta)k$. The loss in this case is $\mathcal{L}(k) = f(k) - k = \vartheta k$. A $(1+\vartheta)k$-robust t-spanner is a ***ϑ-reliable t-spanner***.

Exact reliable spanners. If the input point set is in one dimension, then one can easily construct a 1-spanner for the points, which means that the exact distances between points on the line are preserved by the spanner. This of course can be done easily by connecting the points from left to right. It becomes significantly more challenging to construct such an exact spanner that is reliable.

1.1 Our results

We investigate how to construct reliable spanners with very small loss – that is ϑ-reliable spanners. To the best of our knowledge nothing was known on this case before this work.

(A) **Exact reliable spanners in one dimension.** We show how to construct an $\mathcal{O}(1)$-reliable exact spanner on any one-dimensional set of n points with $\mathcal{O}(n \log n)$ edges. The idea of the construction is to build a binary tree over the points, and to build bipartite expanders between certain subsets of nodes in the same layer. One can think of this construction as building different layers of expanders for different resolutions. The construction is described in Section 3. See Theorem 3.3 for the result.
One can get added redundancy by systematically shifting the layers. Done carefully, this results in a ϑ-reliable exact spanner. See Theorem 3.4 for the result.

(B) **Reliable $(1+\varepsilon)$-spanners in higher dimensions.** We next show a surprisingly simple and elegant construction of ϑ-reliable spanners in two and higher dimensions, using a recent result of Chan *et al.* [9], which show that one needs to maintain only a "few" linear orders. This immediately reduces the d dimensional problem to maintaining a reliable spanner for each of this orderings, which is the problem we already solved. See Section 4 for details.

Omitted proofs and more results can be found in the full version [7]. A very recent result of Bose *et al.* [4] obtains similar bounds on the size of reliable spanners in higher dimensions.

2 Preliminaries

For a set X of vertices in a graph $G = (V, E)$, let $\Gamma(X) = \left\{ v \in V \;\middle|\; uv \in E \text{ for a } u \in X \right\}$ be the ***neighbors*** of X in G. The following lemma, which is a standard expander construction, provides the main building block of our one-dimensional construction.

▶ **Lemma 2.1.** *Let L, R be two disjoint sets, with a total of n elements, and let $\xi \in (0,1)$ be a parameter. One can build a bipartite graph $G = (L \cup R, E)$ with $\mathcal{O}(n/\xi^2)$ edges, such that*

(I) for any subset $X \subseteq L$, with $|X| \geq \xi|L|$, we have that $|\Gamma(X)| > (1-\xi)|R|$, and

(II) for any subset $Y \subseteq R$, with $|Y| \geq \xi|R|$, we have that $|\Gamma(Y)| > (1-\xi)|L|$.

Let $[n]$ denote the set $\{1, 2, \ldots, n\}$ and let $[i : j] = \{i, i+1, \ldots, j\}$. Our purpose is to build a reliable 1-spanner in one dimension. Intuitively, a point in $[n]$ is in trouble, if many of its close by neighbors belong to the failure set B. Such an element is in the shadow of B, defined formally next.

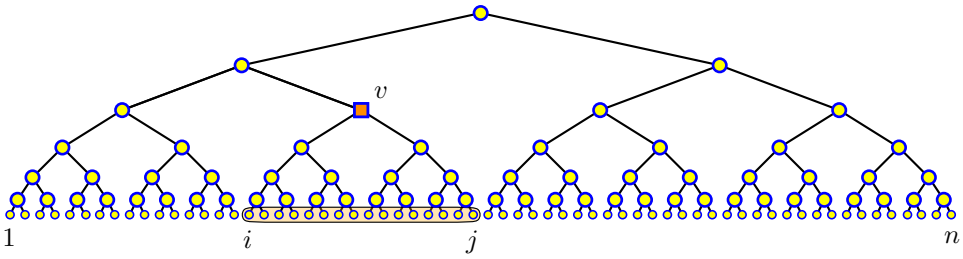


Figure 1 The binary tree built over $[n]$. The block of node v is the interval $[i : j]$.

▶ **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 $\left|[i : j] \cap B\right| \geq \alpha \left|[i : j]\right|$. Similarly, i is in the ***right* α*-shadow*** of B, if and only if there exists an integer i, such that $h \leq i$ and $|[h : i] \cap B| \geq \alpha \, |[h : i]|$. 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.** *Fix a set $B \subseteq [n]$ and let $\alpha \in (0, 1)$ be a parameter. Then, we have that $|\mathcal{S}_{\rightarrow}(B)| \leq (1 + \lceil 1/\alpha \rceil) \, |B|$. In particular, the size of $\mathcal{S}(\alpha, B)$ is at most $2(1 + \lceil 1/\alpha \rceil) \, |B|$.*

3 Reliable spanners in one dimension

3.1 Constructing the graph H

Assume n is a power of two, and consider building the natural full binary tree T with the numbers of $[n]$ as the leaves. Every node v of T corresponds to an interval of numbers of the form $[i : j]$ its canonical interval, which we refer to as the block of v, see Figure 1. Let $\mathcal{I}$ be the resulting set of all blocks. In each level one can sort the blocks of the tree from left to right. Two adjacent blocks of the same level are neighbors. For a block $I \in \mathcal{I}$, let $\text{next}(I)$ and $\text{prev}(I)$ be the blocks (in the same level) directly to the right and left of I, respectively. We build the graph of Lemma 2.1 with $\xi = 1/16$ for any two neighboring blocks in $\mathcal{I}$. Let H be the resulting graph when taking the union over all the sets of edges generated by the above.

3.2 Analysis

In the following we show that the resulting graph H is an $\mathcal{O}(1)$-reliable 1-spanner on $\mathcal{O}(n \log n)$ edges. We start by verifying the size of the graph.

▶ **Lemma 3.1.** *The graph H has $\mathcal{O}(n \log n)$ edges.*

Proof. Let $h = \log n$ be the depth of the tree T. In each level $i = 1, 2, \ldots, h$ of T there are 2^{h-i} nodes and the blocks of these nodes have size 2^i. The number of pairs of adjacent blocks in level i is $2^{h-i} - 1$ and each pair contributes $\mathcal{O}(2^i)$ edges. Therefore, each level of T contributes $\mathcal{O}(n)$ edges. We get $\mathcal{O}(n \log n)$ for the overall size by summing up for all levels. ◀

Given two numbers i and j, where $i < j$, consider the two blocks $I, J \in \mathcal{I}$ that correspond to the two numbers at the bottom level. Set $I_0 = I$, and $J_0 = J$. We now describe a canonical walk from I to J, where initially $\ell = 0$. During the walk we have two active blocks I_ℓ and

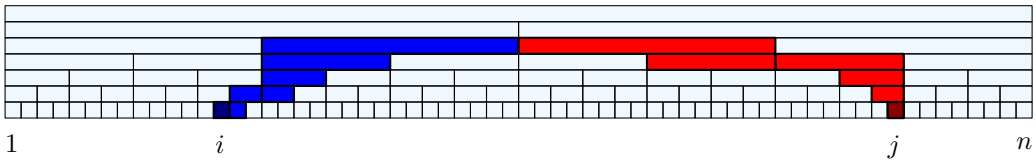


Figure 2 The canonical path between the vertices i and j. The blue blocks correspond to the ascent part and the red blocks correspond to the descent part of the walk.

J_ℓ, that are both in the same level. For any block $I \in \mathcal{I}$ we denote its parent by $p(I)$. At every iteration we bring the two active blocks closer to each other by moving up in the tree.

Specifically, repeatedly do the following:

(A) If I_ℓ and J_ℓ are neighbors then the walk is done.
(B) If I_ℓ is the right child of $p(I_\ell)$, then set $I_{\ell+1} = \text{next}(I_\ell)$ and $J_{\ell+1} = J_\ell$, and continue to the next iteration.
(C) If J_ℓ is the left child of $p(J_\ell)$, then set $I_{\ell+1} = I_\ell$ and $J_{\ell+1} = \text{prev}(J_\ell)$, and continue to the next iteration.
(D) Otherwise – the algorithm ascends. It sets $I_{\ell+1} = p(I_\ell)$, and $I_{\ell+1} = p(J_\ell)$, and it continues to the next iteration.

It is easy to verify that this walk is well defined, and let

$$\pi(i,j) \equiv \underbrace{I_0 \to I_1 \to \cdots \to I_\ell}_{\text{ASCENT}} \to \underbrace{J_\ell \to \cdots \to J_0}_{\text{DESCENT}}$$

be the resulting walk on the blocks where we removed repeated blocks. Figure 2 illustrates the path of blocks between two vertices i and j.

In the following, consider a fixed set $B \subseteq [n]$ of faulty nodes. A block $I \in \mathcal{I}$ is ***α*-contaminated**, for some $\alpha \in (0,1)$, if $|I \cap B| \geq \alpha |I|$.

▶ **Lemma 3.2.** *Consider two nodes $i, j \in [n]$, with $i < j$, and let $\pi(i,j)$ be the canonical path between i and j. If any block of $\pi = \pi(i,j)$ is α-contaminated, then i or j are in the $\alpha/3$-shadow of B.*

▶ **Theorem 3.3.** *The graph H constructed above on the set $[n]$ is an $\mathcal{O}(1)$-reliable exact spanner and has $\mathcal{O}(n \log n)$ edges.*

3.3 ϑ-reliable exact spanners

We can extend Theorem 3.3, to build a one dimensional graph H_ϑ, such that for any fixed $\vartheta > 0$ and any set B of k deleted vertices, at most $(1+\vartheta)k$ vertices are no longer connected by a 1-path after the removal of B. The basic idea is to retrace the construction of Theorem 3.3, and extend it to this more challenging case. The main new ingredient is a shifting scheme.

▶ **Theorem 3.4** ([7]). *For parameters n and $\vartheta > 0$, the graph H_ϑ constructed over $[n]$, is a ϑ-reliable exact spanner. Furthermore, H_ϑ has $\mathcal{O}(\vartheta^{-6} n \log n)$ edges.*

4 Building a reliable spanner in $\mathbb{R}^d$

In the following, we assume that $P \subseteq [0,1)^d$ – this can be done by an appropriate scaling and translation of space. For an ordering σ of $[0,1)^d$, and two points $p, q \in [0,1)^d$, such that $p \prec q$, let $(p,q)_\sigma = \left\{ z \in [0,1)^d \;\middle|\; p \prec z \prec q \right\}$ be the set of points between p and q in the order σ. We need the following minor variant of a result of Chan *et al.* [9].

▶ **Theorem 4.1** ([9]). *For $\varsigma \in (0,1)$, there is a set $\Pi^+(\varsigma)$ of $M(\varsigma) = \mathcal{O}(\varsigma^{-d}\log\varsigma^{-1})$ orderings of $[0,1)^d$, such that for any two (distinct) points $p, q \in [0,1)^d$, with $\ell = \|p-q\|$, there is an ordering $\sigma \in \Pi^+$, and a point $z \in [0,1)^d$, such that*

(i) $p \prec_\sigma q$,
(ii) $(p,z)_\sigma \subseteq \mathrm{ball}\big(p, \varsigma\ell\big)$,
(iii) $(z,q)_\sigma \subseteq \mathrm{ball}\big(q, \varsigma\ell\big)$, *and*
(iv) $z \in \mathrm{ball}\big(p, \varsigma\ell\big)$ *or* $z \in \mathrm{ball}\big(q, \varsigma\ell\big)$.

Furthermore, given such an ordering σ, and two points p, q, one can compute their ordering, according to σ, using $O(d\log\varsigma^{-1})$ arithmetic and bitwise-logical operations.

4.1 Construction

Given a set P of n points in $[0,1)^d$, and parameters $\varepsilon, \vartheta \in (0,1)$, let $\varsigma = \varepsilon/(c\log n)$,

$$M = 4M(\varsigma) = \mathcal{O}(\varsigma^{-d}\log\varsigma^{-1}) = \mathcal{O}\left(\varepsilon^{-d}\log^d n \log\frac{\log n}{\varepsilon}\right),$$

and c be some sufficiently large constant. Next, let $\vartheta' = \vartheta/M$, and let $\Pi^+ = \Pi^+(\varsigma)$ be the set of orderings of Theorem 4.1. For each ordering $\sigma \in \Pi^+$, compute the ϑ'-reliable exact spanner G_σ of P, see Theorem 3.4, according to σ. Let G be the resulting graph by taking the union of G_σ for all $\sigma \in \Pi^+$.

4.2 Analysis

▶ **Theorem 4.2.** *The graph G constructed above is a ϑ-reliable $(1+\varepsilon)$-spanner and has size $\mathcal{O}\left(\varepsilon^{-7d}\vartheta^{-6}n\log^{7d} n\log^7\frac{\log n}{\varepsilon}\right)$.*

Proof. Given a (failure) set $B \subseteq P$, let B^+ be the union of all the harmed sets resulting from B in G_σ, for all $\sigma \in \Pi^+$. We have that $|B^+| \leq (1 + M\cdot\vartheta')\,|B| = (1+\vartheta)\,|B|$.

Consider any two points $p, q \in P \setminus B^+$. By Theorem 4.1, for $\ell = \|p-q\|$, there exists an ordering $\sigma \in \Pi^+$, and a point $z \in [0,1)^d$, such that $(p,z)_\sigma \subseteq \mathrm{ball}(p,\varsigma\ell)$ and $(z,q)_\sigma \subseteq \mathrm{ball}(q,\varsigma\ell)$ (and z is in one of these balls).

By Theorem 3.4, the graph $G_\sigma \setminus B \subseteq G \setminus B$ contains a monotone path π, according to σ, with $h = \mathcal{O}(\log n)$ hops, connecting p to q. Let $p = p_1, \ldots, p_{h+1} = q$ be this path. Observe that there is a unique index i, such that $z \in (p_i, p_{i+1})$. We have the following:

(A) $\forall j \neq i \quad \|p_j - p_{j+1}\| \leq 2\varsigma\ell$. (B) $\|p_i - p_{i+1}\| \leq \ell + 2\varsigma\ell$.

As such, the total length of π is $\sum_{j=1}^{h}\|p_j - p_{j+1}\| = (1+2\varsigma h)\ell \leq (1+\varepsilon)\ell$, as desired, if c is sufficiently large. Namely, G is the desired reliable spanner.

The number of edges of G is

$$M\cdot\mathcal{O}\big((\vartheta')^{-6}n\log n\big) = \mathcal{O}\big(M(M/\vartheta)^6 n\log n\big) = \mathcal{O}\left(\varepsilon^{-7d}\vartheta^{-6}n\log^{7d} n\log^7\frac{\log n}{\varepsilon}\right). \quad ◀$$

4.3 Improved constructions

By setting $\varsigma = \varepsilon/c$ in the above construction and applying a more careful analysis we can improve this result, which is stated in the following theorem.

▶ **Theorem 4.3.** *One can construct a ϑ-reliable $(1+\varepsilon)$-spanner with size*

$$\mathcal{O}\left(\varepsilon^{-7d}\log^7\frac{1}{\varepsilon}\cdot\vartheta^{-6}n\log n(\log\log n)^6\right).$$

Using another construction we are able to obtain a ϑ-reliable $(1+\varepsilon)$-spanner with size $\mathcal{O}\big(\varepsilon^{-d}\vartheta^{-2}n\log n\big)$ if the underlying point set P has polynomially bounded spread, which is optimal. For both of these results see the full version [7].

References

1. B. Aronov, M. de Berg, O. Cheong, J. Gudmundsson, H. J. Haverkort, M. H. M. Smid, and A. Vigneron. Sparse geometric graphs with small dilation. *Computational Geometry: Theory and Applications*, 40(3):207–219, 2008.
2. S. Arya, D. M. Mount, and M. Smid. Randomized and deterministic algorithms for geometric spanners of small diameter. In *Proc. 35th Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), pages 703–712, 1994.
3. S. Arya, D. M. Mount, and M. Smid. Dynamic algorithms for geometric spanners of small diameter: Randomized solutions. *Computational Geometry: Theory and Applications*, 13(2):91–107, 1999.
4. P. Bose, P. Carmi, V. Dujmovic, and P. Morin. Near-optimal O(k)-robust geometric spanners. *CoRR*, abs/1812.09913, 2018.
5. P. Bose, P. Carmi, M. Farshi, A. Maheshwari, and M. Smid. Computing the greedy spanner in near-quadratic time. *Algorithmica*, 58(3):711–729, November 2010.
6. P. Bose, V. Dujmović, P. Morin, and M. Smid. Robust geometric spanners. *SIAM Journal on Computing*, 42(4):1720–1736, 2013.
7. K. Buchin, S. Har-Peled, and D. Oláh. A spanner for the day after. *CoRR*, abs/1811.06898, 2018.
8. P. Carmi and L. Chaitman. Stable roommates and geometric spanners. In *Proc. 22nd Canad. Conf. Comput. Geom.* (CCCG), pages 31–34, 2010.
9. T. M. Chan, S. Har-Peled, and M. Jones. On locality-sensitive orderings and their applications. *CoRR*, abs/1809.11147, 2018.
10. J. Gudmundsson, C. Levcopoulos, and G. Narasimhan. Fast greedy algorithms for constructing sparse geometric spanners. *SIAM J. Comput.*, 31(5):1479–1500, May 2002.
11. 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. ACM, 1998.
12. C. Levcopoulos, G. Narasimhan, and M. Smid. Improved algorithms for constructing fault-tolerant spanners. *Algorithmica*, 32(1):144–156, 2002.
13. T. Lukovszki. New results of fault tolerant geometric spanners. In *Proc. 6th Workshop Algorithms Data Struct.* (WADS), volume 1663 of *LNCS*, pages 193–204. Springer, 1999.
14. G. Narasimhan and M. Smid. *Geometric spanner networks*. Cambridge University Press, New York, NY, USA, 2007.
15. M. Smid. Geometric spanners with few edges and degree five. In *Proc. 12th Australasian Theo. Sym.* (CATS), volume 51 of *CRPIT*, pages 7–9, 2006.

On the 2-Colored Crossing Number*

Oswin Aichholzer[1], Ruy Fabila-Monroy[2], Adrian Fuchs[1], Carlos Hidalgo-Toscano[2], Irene Parada[1], Birgit Vogtenhuber[1], and Francisco Zaragoza[3]

1 **Graz University of Technology, Graz, Austria**
{oaich, iparada, bvogt}@ist.tugraz.at; adrian.fuchs@student.tugraz.at
2 **Departamento de Matemáticas, Cinvestav, Ciudad de México, México**
ruyfabila@math.cinvestav.edu.mx, cmhidalgo@math.cinvestav.mx
3 **Universidad Autónoma Metropolitana, Ciudad de México, México**
franz@correo.azc.uam.mx

Abstract

Let D be a straight-line drawing of a graph where every edge is colored with one of two possible colors. The rectilinear 2-colored crossing number of D is the minimum number of crossings between edges of the same color, taken over all possible colorings of D. We show lower and upper bounds on the rectilinear 2-colored crossing number for the complete graph K_n. Moreover, for fixed drawings of K_n we give bounds on the relation between its rectilinear 2-colored crossing number and its rectilinear crossing number.

1 Introduction

In any drawing D of a non-planar graph G in the plane, two of its edges will cross. From both a theoretical and practical point of view it is of interest to study the minimum number of pairs of edges that cross in any drawing of G. This is known as the *crossing number* $\mathrm{cr}(G)$ of G. There are many variants on crossing numbers. In this paper we focus on a variant mixing two of them: the *biplanar crossing number* and the *rectilinear crossing number*.

The *biplanar crossing number* of a graph G, $\mathrm{cr}_2(G)$, is the minimum of $\mathrm{cr}(G_1) + \mathrm{cr}(G_2)$ over all graphs G_1 and G_2 whose union is G. This parameter was introduced by Owens [12]. For a survey on biplanar crossing numbers of graphs see [9, 10].

A *straight-line drawing* of G is a drawing D of G in the plane in which the vertices are drawn as points in general position and the edges are drawn as straight line segments. We identify the vertices and edges of the underlying abstract graph with the corresponding ones in the straight-line drawing. The *rectilinear crossing number* of G, $\overline{\mathrm{cr}}(G)$, is the minimum number of pairs of edges that cross in any straight-line drawing of G. Of special relevance is $\overline{\mathrm{cr}}(K_n)$, the rectilinear crossing number of the complete graph on n vertices. The current best published bounds on $\overline{\mathrm{cr}}(K_n)$ are $0.379972\binom{n}{4} < \overline{\mathrm{cr}}(K_n) < 0.380473\binom{n}{4} + \Theta(n^3)$ [3, 11]. The upper bound has been improved in an upcoming paper [4] to $\overline{\mathrm{cr}}(K_n) < 0.3804493\binom{n}{4} + \Theta(n^3)$.

A 2-*edge-coloring* of a drawing D of a graph is an assignment of one of two possible colors to every edge of D. The *rectilinear 2-colored crossing number* of a graph G, $\overline{\mathrm{cr}}_2(G)$, is the minimum number of monochromatic crossings (pairs of edges of the same color that cross)

* O.A. and I.P. partially supported by the Austrian Science Fund (FWF) grant W1230. R.F. and C.H. partially supported by CONACYT (Mexico), grant 253261. B.V. 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.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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 a 2-edge-colored straight-line drawing of G. This parameter was introduced before and called the *geometric 2-planar crossing number* [13]. We prefer our terminology because the term k-planar is extensively used in graph drawing with a different meaning.

In this paper we focus on the case where G is the complete graph K_n, and we prove the following lower and upper bounds on $\overline{\mathrm{cr}}_2(K_n)$:

$$1/33\binom{n}{4} + \Theta(n^3) < \overline{\mathrm{cr}}_2(K_n) < 0.11798016\binom{n}{4} + \Theta(n^3).$$

Combining the lower bound on $\overline{\mathrm{cr}}(K_n)$ and the upper bound on $\overline{\mathrm{cr}}_2(K_n)$ implies that asymptotically, $\overline{\mathrm{cr}}_2(K_n)/\overline{\mathrm{cr}}(K_n) \leq 0.31049652 + o(1)$.

Note that drawings with few crossings don't necessarily admit a coloring with few monochromatic crossings. This observation motivates the following question: Given a fixed straight-line drawing D of K_n, what is the ratio between the number of monochromatic crossings for the best 2-edge-coloring of D and the number of (uncolored) crossings in D? A simple probabilistic argument shows that this ratio is at most $1/2$. We show that for sufficiently large n, it is less than $1/2 - c$ for some positive constant c.

In a slight abuse of notation, we denote with $\overline{\mathrm{cr}}(D)$ the number of pairs of edges in D that cross and call it the rectilinear crossing number of D. The (rectilinear) 2-colored crossing number of a straight-line drawing D, $\overline{\mathrm{cr}}_2(D)$, is then the minimum of $\overline{\mathrm{cr}}(D_1) + \overline{\mathrm{cr}}(D_2)$, over all straight-line drawings D_1 and D_2 whose union is D. For a given 2-edge-coloring χ of D, we denote with $\overline{\mathrm{cr}}_2(D, \chi)$ the number of monochromatic crossings in D. Thus, $\overline{\mathrm{cr}}_2(D)$ is the minimum of $\overline{\mathrm{cr}}_2(D, \chi)$ over all 2-edge-colorings χ of D.

2 Lower bounds on $\overline{\mathrm{cr}}_2(D)/\overline{\mathrm{cr}}(D)$

In this section we study the extreme values that $\overline{\mathrm{cr}}_2(D)/\overline{\mathrm{cr}}(D)$ can attain for straight-line drawings D of K_n. In the full version of this paper we explore certain classes of straight-line drawings of K_n. Among others, we show that there exist classes of drawings D of K_n for which $\overline{\mathrm{cr}}_2(D)/\overline{\mathrm{cr}}(D) \leq 1/3 + o(1)$ and that for convex straight-line drawings of K_n it holds that $\overline{\mathrm{cr}}_2(D)/\overline{\mathrm{cr}}(D) = 3/8 - o(1)$.

Using a simple probabilistic argument it can be shown that $\overline{\mathrm{cr}}_2(D)/\overline{\mathrm{cr}}(D) \leq 1/2$ for every straight-line drawing D of K_n. This lower bound can be improved for any such drawing.

▶ **Theorem 2.1.** *There exists a positive integer n_0 and a positive constant c such that for any straight-line drawing D of the complete graph K_n on $n \geq n_0$ vertices, $\overline{\mathrm{cr}}_2(D)/\overline{\mathrm{cr}}(D) \leq 1/2 - c$.*

The proof, based on the positive Fraction Erdős-Szekeres theorem [7], can be found in the full version of this paper.

3 Upper bounds on $\overline{\mathrm{cr}}_2(K_n)$

For the rectilinear crossing number $\overline{\mathrm{cr}}(K_n)$ the best upper bound [4] comes from finding examples of straight-line drawings of K_n with few crossings (for a small value of n) which are then used as a seed for the duplication process in [2, 3]. In this section, we prove that a more involved but similar approach can be adopted for the two-colored case.

Throughout this section P is a set of m points in general position in the plane. Let p be a point in P. Given a 2-edge-coloring χ of the edges of the straight-line drawing of K_m that P induces, we denote by $L(p)$ and $S(p)$ the edges incident to p of the larger and smaller color class at p, respectively. An edge e incident to p is called a *χ-halving edge of p* if the

number of edges of $L(p)$ to the right of the line ℓ_e spanned by e (and directed from q to p) and the number of edges of $L(p)$ to the left of ℓ_e differ by at most one. An assignment between the points of P and its χ-halving edges is called a *χ-halving matching of P*.

▶ **Theorem 3.1.** *Let P be a set of m points in general position with a two-coloring χ of the edges of the straight-line drawing of K_m it induces. If P has a χ-halving matching, then*

$$\overline{\mathrm{cr}}_2(K_n) \leq \frac{24A}{m^4}\binom{n}{4} + \Theta(n^3)$$

where A is a rational number that depends on P, χ, and the χ-halving matching of P.

Proof. First we describe a process to obtain from P a set Q of $2m$ points, a 2-edge-coloring χ' of the of the straight-line drawing of K_{2m} that Q induces, and a χ'-halving matching for Q. The set Q is constructed as follows. Let p be a point in P and $e = (p, q)$ its χ-halving edge in the matching. We add to Q two points p_1, p_2 placed along the line spanned by e and in a small neighborhood of p such that: (i) if f is an edge different from e that is incident to p, then p_1 and p_2 lie on different sides of the line spanned by f; (ii) if f is an edge different from e that is not incident to p, then p_1 and p_2 lie on the same side of the line spanned by f as p; and (iii) the point p_1 is farther away from q than p_2. The set Q has $2m$ points and the above conditions ensure that they are in general position.

Next, we define a coloring χ' and a χ'-halving matching for Q. For every edge (p, q) of P, color the four edges $(p_i, q_j), i, j \in \{1, 2\}$ with the same color as (p, q). Hence the only edges remaining to be colored are the edges (p_1, p_2) between the duplicates of a point $p \in P$. Let ℓ_e be the line spanned by e and directed from q to p and let q_1, q_2 be the points that originated from q, such that q_1 lies to the left of ℓ_e and q_2 lies to the right of ℓ_e. We assume that the colors are red and blue and that the larger color class at p is blue.

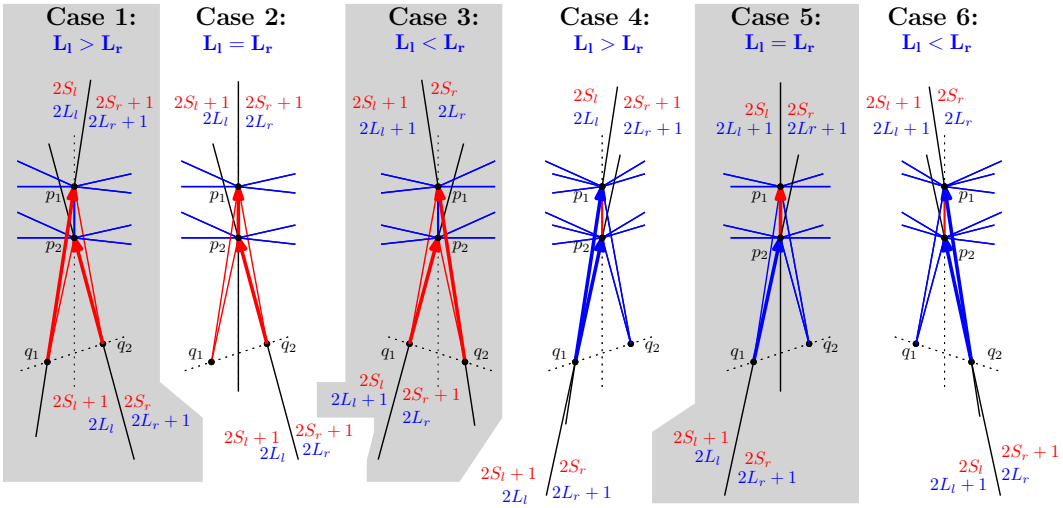


Figure 1 The cases in the duplication process of Theorem 3.1 when the larger color class at p is blue. The edge e of P has color red in the first 3 cases and color blue in the last 3 cases. The dotted lines represent the lines spanned by the χ-halving edges for P. The thick edges represent the χ'-halving edges for Q, where the arrow points to the point it is matched to. L_l and L_r (S_l and S_r) represent the number of blue (red) edges at p to the left and right of l_e, respectively. The number of colored edges on each side of the lines spanned by thick edges indicate the resulting numbers of red and blue edges to the left and right of the χ'-halving edges for Q.

There are six cases in which p can fall. They are all shown in Figure 1, which depicts among others the color that the edge (p_1, p_2) receives and that the edges matched to p_1 and p_2 are indeed χ'-halving edges in each case.

Note that no point in Q falls in Case 5. From now on, we assume that no point in P falls in Case 5. Our goal is to iterate the duplication process and obtain a bound on $\overline{\mathrm{cr}}_2(K_n)$. Let $k \geq 1$ be an integer and let (Q_k, χ_k) be the pair obtained by iterating the duplication process k times. We claim the following on $\overline{\mathrm{cr}}_2(Q_k, \chi_k)$, the number of monochromatic crossings in the straight-line 2-edge colored drawing of K_n induced by Q_k and χ_k:

▶ **Claim.** *After k iterations of the duplication process, the following holds*

$$\overline{\mathrm{cr}}_2(Q_k, \chi_k) = A \cdot 2^{4k} + B \cdot 2^{3k} + C \cdot 2^{2k} + D \cdot 2^k$$

where A, B, C and D are rational numbers that depend on P and its χ-halving matching.

The proof of this claim can be found in the full version of this paper. Letting $n = 2^k m$:

$$\overline{\mathrm{cr}}_2(K_n) \leq \overline{\mathrm{cr}}_2(Q_k, \chi_k) = \frac{24A}{m^4}\binom{n}{4} + \Theta(n^3)$$

which proves the theorem when n is of the form $2^k m$. The proof for $2^k m < n < 2^{k+1} m$ follows from showing that $\overline{\mathrm{cr}}(K_n)$ is an increasing function. ◀

3.1 Small configurations

The previous section implies that for a large number of vertices we can obtain straight-line drawings of the complete graph with a reasonable small 2-colored crossing number from *good* sets of constant size. Thus, in this section we describe how to obtain those small good sets.

Similar as in [4] we combine different methods to obtain straight-line drawings of the complete graph with low 2-colored crossing number. The overall approach is to apply three different methods in alternating order: we start with a known set, apply the duplication process from Theorem 3.2 to obtain a larger set, locally optimize it to get a better set, find good subsets, locally optimize them, duplicate the resulting sets and so on.

The currently best (w.r.t. to the crossing constant, see below) straight-line drawing D with 2-edges coloring χ we found[1] has $n = 135$ vertices, a 2-colored crossing number of $\overline{\mathrm{cr}}_2(D, \chi) = 1470756$, and contains a χ-halving matching.

3.2 Rectilinear 2-colored crossing constant

Let $\overline{\mathrm{cr}}_2$ be the *rectilinear 2-colored crossing constant*, that is, the constant such that the best straight-line drawing of K_n for large values of n has at most $\overline{\mathrm{cr}}_2\binom{n}{4}$ monochromatic crossings. Its existence follows from showing that $\lim_{n\to\infty} \overline{\mathrm{cr}}_2(K_n)/\binom{n}{4}$ exits and is a positive number.

From the previous results in this section we can derive an upper bound for the 2-colored crossing constant from a given set of constant size with a small 2-colored crossing number:

Plugging the values of the set of 135 points obtained in the last section into Theorem 3.2 (after duplicating once to get rid of Case 5) we get the upper bound of $\overline{\mathrm{cr}}_2 < 0.11798016$.

▶ **Theorem 3.2.** *The 2-colored crossing constant satisfies $\overline{\mathrm{cr}}_2 \leq \frac{182873519}{1550036250} < 0.11798016$.*

[1] The interested reader can get a file with the coordinates of the points, the colors of the edges, and a χ-halving matching from `http://www.crossingnumbers.org/projects/monochromatic/sets/n135.php`.

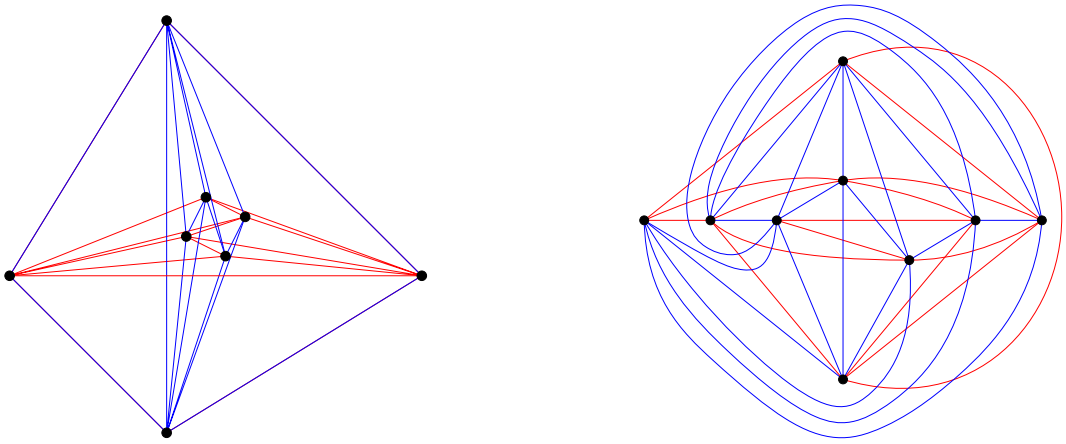


Figure 2 Left: a 2-colored rectilinear drawing of K_8 without monochromatic crossings. Right: a 2-colored drawing of K_9 with only one monochromatic (red) crossing.

In [3] a lower bound of $\overline{\mathrm{cr}} \geq \frac{277}{729} > 0.37997267$ has been shown for the rectilinear crossing constant. We can thus give an upper bound on the asymptotic ratio between the best 2-colored straight-line drawing of K_n and the best straight-line drawing of K_n of $\overline{\mathrm{cr}}_2/\overline{\mathrm{cr}} \leq 0.31049652$.

4 Lower bounds on $\overline{\mathrm{cr}}_2(K_n)$ and $\mathrm{cr}_2(K_n)$

The following result shows that from the 2-colored rectilinear crossing number of small sets we can obtain lower bounds for larger sets.

▶ **Lemma 4.1.** *Let* $\overline{\mathrm{cr}}_2(\hat{n}) = \hat{c}$ *for some* $\hat{n} \geq 4$. *Then for* $n > \hat{n}$ *we have* $\overline{\mathrm{cr}}_2(K_n) \geq \frac{24\hat{c}}{\hat{n}(\hat{n}-1)(\hat{n}-2)(\hat{n}-3)}\binom{n}{4}$ *which implies* $\overline{\mathrm{cr}}_2 \geq \frac{24\hat{c}}{\hat{n}(\hat{n}-1)(\hat{n}-2)(\hat{n}-3)}$

Proof. Every subset of $\hat{n}$ points of K_n induces a drawing with at least $\hat{c}$ crossings, and thus we have $\hat{c}\binom{n}{\hat{n}}$ crossings in total. In this way every crossing is counted $\binom{n-4}{\hat{n}-4}$ times. This results in a total of $\frac{24\hat{c}}{\hat{n}(\hat{n}-1)(\hat{n}-2)(\hat{n}-3)}\binom{n}{4}$ crossings. ◀

With a strategy based on the intersection graph of a given straight-line drawing, we have been able to determine all the 2-colored crossing numbers of all straight-line drawings of K_9 and prove that $\overline{\mathrm{cr}}_2(K_9) = 2$. More details about this strategy can be found in the full version. Using Lemma 4.1 for $\hat{n} = 9$ and $\hat{c} = 2$ we get a bound of $\overline{\mathrm{cr}}_2 \geq 1/63$. Repeating the process of computing lower bounds for sets of small cardinality we checked all order types of size 11 [5]. We obtained $\overline{\mathrm{cr}}_2(K_{11}) = 10$ and by Lemma 4.1 this gives the even better bound of $\overline{\mathrm{cr}}_2 \geq 1/33$.

4.1 Staight-line versus general drawings

The best straight-line drawings of K_n with $n \leq 8$ have no monochromatic crossing, see Figure 2 left for a straight-line 2-colored crossing-free drawing of K_8. In [13], Section 3, the authors claim that up to now no graph was known were the k-colored crossing number was strictly smaller than the rectilinear k-colored crossing number for any $k \geq 2$. From the previous section we know that $\overline{\mathrm{cr}}_2(K_9) = 2$. Inspecting rotation systems for $n = 9$ [1] which have the minimum number of 36 crossings, we have been able to construct a drawing of K_9 which has only one monochromatic crossing, see Figure 2 right. As the graph thickness of

K_9 is 3 [8, 14], we can not draw K_9 with just two colors without monochromatic crossings. Thus, the biplanar crossing number for K_9 is 1 and thus strictly smaller than $\overline{\text{cr}}_2(K_9) = 2$.

5 Conclusion and open problems

In this paper we have shown lower and upper bounds on the rectilinear 2-colored crossing number for K_n as well as its relation to the rectilinear crossing number for fixed drawings of K_n. Besides improving the given bounds, some open problems arise from our work. The first question is how fast we can compute the best edge-coloring of a given rectilinear drawing of K_n. A second question is on the structure of 2-colored crossing minimal sets. For the rectilinear crossing number it is known that optimal sets have a triangular convex hull [6]. For $n = 8, 9$ we have optimal sets with 3 and 4 extreme points, but so far all minimal sets for $n \geq 10$ have a triangular convex hull.

References

1 Bernardo M. Ábrego, Oswin Aichholzer, Silvia Fernández-Merchant, Thomas Hackl, Jürgen Pammer, Alexander Pilz, Pedro Ramos, Gelasio Salazar, and Birgit Vogtenhuber. All good drawings of small complete graphs. In *Proc. 31st European Workshop on Computational Geometry (EuroCG '15)*, pages 57–60, 2015.

2 Bernardo M. Ábrego and Silvia Fernández-Merchant. Geometric drawings of K_n with few crossings. *Journal of Combinatorial Theory, Series A*, 114(2):373–379, 2007. doi:10.1016/j.jcta.2006.05.003.

3 Bernardo M. Ábrego, Silvia Fernández-Merchant, Jesús Leaños, and Gelasio Salazar. A central approach to bound the number of crossings in a generalized configuration. *Electronic Notes in Discrete Mathematics*, 30:273–278, 2008. doi:10.1016/j.endm.2008.01.047.

4 Oswin Aichholzer, Frank Duque, Óscar Eduardo García-Quintero, Ruy Fabila-Monroy, and Carlos Hidalgo-Toscano. An ongoing project to improve the rectilinear and pseudolinear crossing constants. Manuscript, 2018.

5 Oswin Aichholzer and Hannes Krasser. Abstract order type extension and new results on the rectilinear crossing number. *Computational Geometry: Theory and Applications, Special Issue on the 21st European Workshop on Computational Geometry*, 36(1):2–15, 2006.

6 Oswin Aichholzer, David Orden, and Pedro Ramos. On the structure of sets attaining the rectilinear crossing number. In *Proc. 22nd European Workshop on Computational Geometry (EuroCG '06)*, pages 43–46, Delphi, Greece, 2006.

7 Imre Bárány and Pavel Valtr. A positive fraction Erdős-Szekeres theorem. *Discrete Comput. Geom.*, 19(3, Special Issue):335–342, 1998. Dedicated to the memory of Paul Erdős. doi:10.1007/PL00009350.

8 Joseph Battle, Frank Harary, and Yukihiro Kodama. Every planar graph with nine points has a nonplanar complement. *Bulletin of The American Mathematical Society*, 68, 1962. doi:10.1090/S0002-9904-1962-10850-7.

9 Éva Czabarka, Ondrej Sýkora, László A. Székely, and Imrich Vrt'o. *Biplanar crossing numbers I: a survey of results and problems*, pages 57–77. Springer Berlin Heidelberg, 2006.

10 Éva Czabarka, Ondrej Sýkora, László A. Székely, and Imrich Vrt'o. Biplanar crossing numbers II. Comparing crossing numbers and biplanar crossing numbers using the probabilistic method. *Random Structures & Algorithms*, 33(4):480–496, 2008. doi:10.1002/rsa.20221.

11 Ruy Fabila-Monroy and Jorge López. Computational search of small point sets with small rectilinear crossing number. *Journal of Graph Algorithms and Applications*, 18(3):393–399, 2014. doi:10.7155/jgaa.00328.

12 Andrew Owens. On the biplanar crossing number. *IEEE Transactions on Circuit Theory*, 18(2):277–280, 1971. doi:10.1109/TCT.1971.1083266.

13 János Pach, László A. Székely, Csaba D. Tóth, and Géza Tóth. Note on k-planar crossing numbers. *Computational Geometry*, 68:2–6, 2018. Special Issue in Memory of Ferran Hurtado. doi:10.1016/j.comgeo.2017.06.015.

14 William T. Tutte. The non-biplanar character of the complete 9-graph. *Canadian Mathematical Bulletin*, 6(3):319–330, 1963. doi:10.4153/CMB-1963-026-x.

Bundled Crossings Revisited

Steven Chaplick[*1], Thomas C. van Dijk[†1], Myroslav Kryven[‡1], Ji-won Park[2], Alexander Ravsky[§3], and Alexander Wolff[¶1]

1 **University of Würzburg, Würzburg, Germany**
`firstname.lastname@uni-wuerzburg.de`
2 **KAIST, Daejeon, Republic of Korea**
`wldnjs1727@kaist.ac.kr`
3 **Pidstryhach Institute for Applied Problems of Mechanics and Mathematics, Nat. Acad. Sciences of Ukraine, Lviv, Ukraine**
`alexander.ravsky@uni-wuerzburg.de`

Abstract

An effective way to reduce clutter in a graph drawing that has (many) crossings is to group edges into *bundles* when they travel in parallel. Each edge can participate in many such bundles. Any crossing in this bundled graph occurs between two bundles, i.e., as a *bundled crossing*. We minimize the number of bundled crossings in circular layouts, where vertices are placed on a circle and edges are routed inside the circle.

For a given graph the goal is to find a bundled drawing with at most k crossings. We show that the problem has an FPT algorithm (in k) when we require a simple circular layout.

1 Introduction

In traditional node–link diagrams, vertices are mapped to points in the plane and edges are usually drawn as straight-line segments connecting the vertices. For large and somewhat dense graphs, however, such layouts tend to be so cluttered that it is hard to see any structure in the data. For this reason, Holten [14] introduced *bundled drawings*, where edges that are close together and roughly go into the same direction are drawn using Bézier curves such that the grouping becomes visible. Due to the practical effectiveness of this approach, it has quickly been adopted by the information visualization community [9, 12, 15, 16, 19]. However, bundled drawings have only recently attracted study from a theoretical point of view. Nevertheless, in his survey on crossing minimization, Schaefer already listed bundled crossing minimization as an open problem [20, page 35].

Fink et al. [11] considered bundled crossings in the context of drawing metro maps. They suggested replacing the classical objective of crossing minimization [3, 13, 17] by what they called *block crossing minimization*. Given a set of x-monotone curves (the metro lines that go through two neighboring stations), a block crossing is the exchange of two adjacent blocks of curves. Fink et al. also introduced *monotone* block crossing minimization where each pair of lines can intersect at most once. They considered various network topologies: single edge, path, (upward) tree, planar graph, (bounded-degree) general graph.

Our research builds on recent work of Fink et al. [10] and Alam et al. [1] who extended the notion of block crossings from sets of x-monotone curves to general drawings of graphs.

[*] Supported by DFG grant WO 758/11-1.
[†] Supported by DFG grant DI 2161/2-1.
[‡] Supported by DAAD.
[§] Supported by Erasmus+.
[¶] Supported by DFG grant WO 758/9-1.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

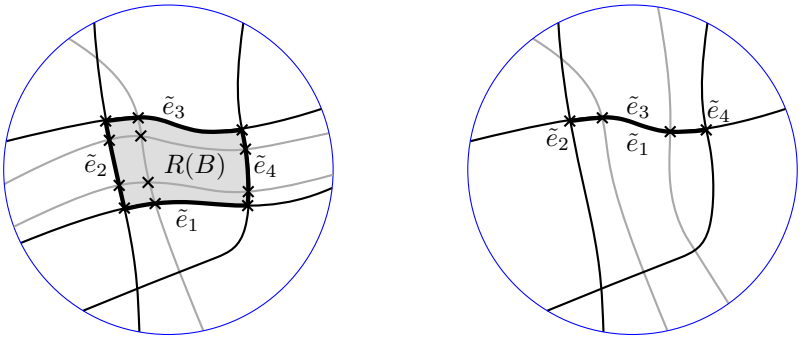


Figure 1 (a) A non-degenerate bundled crossing B and (b) a degenerate bundled crossing B' where one bundle consists of just one edge piece $\tilde{e}_1$ ($\tilde{e}_1 = \tilde{e}_3 = R(B')$) with endpoints $\tilde{e}_2$ and $\tilde{e}_4$.

It is common to define a drawing of a graph as a function that maps vertices to points in the plane and edges to Jordan arcs that connect the corresponding points. Here we will consider simple drawings, that is, any two edges intersect at most once and no edge self-intersects. We will often identify vertices with their points and edges with their curves.

Let D be a drawing, and let $I(D)$ be the set of intersection points among the edges in D. We say that a *bundling* of D is a partition of $I(D)$ into *bundled crossings*, where a set $B \subseteq I(D)$ is a bundled crossing if the following holds (see Fig. 1).

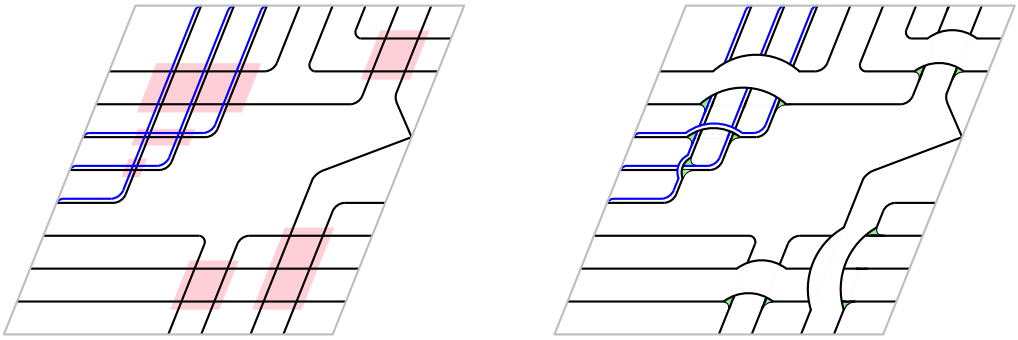
- B is contained in a closed Jordan region $R(B)$ whose boundary consists of four Jordan arcs $\tilde{e}_1$, $\tilde{e}_2$, $\tilde{e}_3$, and $\tilde{e}_4$ that are pieces of edges e_1, e_2, e_3, and e_4 in D.
- The pieces of the edges cut out by the region $R(B)$ can be partitioned into two sets $\tilde{E}_1$ and $\tilde{E}_2$ such that $\tilde{e}_1, \tilde{e}_3 \in \tilde{E}_1$, $\tilde{e}_2, \tilde{e}_4 \in \tilde{E}_2$, and each pair of edge pieces in $\tilde{E}_1 \times \tilde{E}_2$ has exactly one intersection point in $R(B)$, whereas no two edge pieces in $\tilde{E}_1$ (respectively $\tilde{E}_2$) have a common point in $R(B)$.

We call the sets of edges E_1 and E_2 corresponding to edge pieces $\tilde{E}_1$ and $\tilde{E}_2$ *bundles*. We call the edges that bound the two bundles of a bundled crossing *frame edges*. We say that a bundled crossing is *degenerate* if at least one of the bundles consists of only one edge piece; see Fig. 1(b). In this case, the region of the plane associated with the crossing coincides with that edge piece. In particular, any point in $I(D)$ by itself is a degenerate bundled crossing.

We consider circular layouts, where vertices are on a circle and edges are inside the circle. We denote by $\mathrm{bc}^\circ(G)$ the *circular bundled crossing numbers* of a graph G, i.e., the smallest number of bundled crossings over all bundlings of all simple circular layouts of G.

For computing $\mathrm{bc}^\circ(G)$, Alam et al. [1] gave an algorithm whose approximation factor depends on the density of the graph. They posed the existence of an FPT algorithm for $\mathrm{bc}^\circ(G)$ as an open question, which we answer in the affirmative. In this note, we first show how to decide whether $\mathrm{bc}^\circ(G) \leq 1$. Then, we generalize our result as follows.

▶ **Theorem 1.** *There is a computable function f such that for any n-vertex graph G and integer k, we can check, in $O(f(k)n)$ time, if $\mathrm{bc}^\circ(G) \leq k$, i.e., if G admits a circular layout with k bundled crossings. Within the same time bound, we can compute such a layout.*



(a) (b)

Figure 2 (a) A bundled drawing D with six bundled crossings (pink); parallel (blue) edges can be inserted to avoid degenerate bundled crossings; (b) the corresponding surface of genus 6; the components of the surface which are not regions are marked in green

2 An FPT Algorithm for Simple Circular Layouts

Our algorithm is inspired by works on circular layouts with at most k crossings [2] and circular layouts where each edge is crossed at most k times [4]. Both first observed that the graphs admitting such circular layouts have treewidth $O(k)$, and then developed algorithms using Courcelle's theorem, which establishes that expressions in extended monadic second order logic can be evaluated efficiently. We define treewidth and MSO_2 in the full version [5].

▶ **Theorem 2** (Courcelle [7, 8]). *For any integer $t \geq 0$ and any MSO_2 formula ψ of length ℓ, an algorithm can be constructed which takes a graph G with n vertices, m edges, and treewidth at most t and decides in $O(f(t,\ell)\cdot(n+m))$ time whether $G \models \psi$ where the function f from this time bound is a computable function of t and ℓ.*

We recall the observation of Alam et al. [1] that a drawing with k bundled crossings can be lifted onto a surface of genus k; see Fig. 2. Then we examine the structure of such a surface and present our algorithm for the case of one bundled crossing and finally for k bundled crossings.

2.1 Constructing the surface determined by a bundled drawing

Consider a bundled circular drawing D, i.e., it is drawn on a disk $\mathfrak{D}$ residing on a sphere, where the boundary of $\mathfrak{D}$ is the circle of D. Note that inserting parallel edges into the drawing (i.e., making our graph a multi-graph) can be done without modifying the bundled drawing, but allows us to assume that every bundled crossing has four distinct frame edges; see Fig. 2. Each bundled crossing B defines a Jordan curve C_B made up of the four Jordan arcs $\tilde{e}_1$, $\tilde{e}_2$, $\tilde{e}_3$, $\tilde{e}_4$ in clockwise order taken from its four frame edges $e_1, \ldots, e_4$ respectively (here (e_1, e_3) and (e_2, e_4) frame the two bundles; $e_i = u_i v_i$). Let C'_B (see Fig. 3) denote a Jordan curve on $\mathfrak{D}$ outside of C_B where every point on C'_B lies at a sufficiently small distance $\epsilon > 0$ from C_B so that C'_B only contains the crossings in B and the distance from C'_B to the crossings outside of C'_B is at least $\frac{2}{3}$ of the distance from C_B to these crossings. Note that C'_B consists of eight Jordan arcs (in clockwise order) $c'_{2,1}, c'_{1,3}, c'_{3,2}, c'_{2,4}, c'_{4,3}, c'_{3,1}, c'_{1,4}, c'_{4,2}$, where $c'_{i,j}$ goes from e_i to e_j. The surface $\mathfrak{D}'$ is constructed by creating a flat handle on top of $\mathfrak{D}$ which connects $c'_{1,3}$ to $c'_{3,1}$ (when we lift the drawing onto this surface the bundle containing e_1 and e_3 will go over this handle), and doing so for each bundled crossing. We lift the drawing D

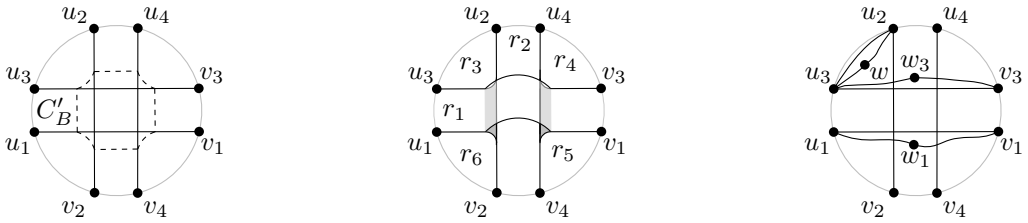


Figure 3 The curve C'_B; the regions $r_1, \ldots, r_6$; augmented graphs G'_{r_1} and G'_{r_3}

onto $\mathfrak{D}'$ obtaining the lifted drawing D'. Clearly, D' is crossing-free. Note that each Jordan curve C'_B remains on our original disk. We will now cut $\mathfrak{D}'$ into *components* (maximal connected subsets) using the frame edges and the Jordan curves C'_B for each B. Namely, for each bundled crossing B, we first cut $\mathfrak{D}'$ along each of the frame edges $e_1, \ldots, e_4$ of B. We additionally cut $\mathfrak{D}'$ along the four *corner* Jordan curves $c'_{2,1}, c'_{3,2}, c'_{4,3}$, and $c'_{1,4}$ of C'_B. This results in a subdivision of $\mathfrak{D}'$ which we call $\mathfrak{S}$. Here, we also use $D_{\mathfrak{S}}$ to denote the sub-drawing of D' on $\mathfrak{S}$, i.e., $D_{\mathfrak{S}}$ is missing the frame edges since these have been cut out. Let us now consider the components of $\mathfrak{S}$. Notice that every edge of $D_{\mathfrak{S}}$ is contained in one component of $\mathfrak{S}$. Furthermore, in order for a component $\mathfrak{s}$ of $\mathfrak{S}$ to contain an edge of $D_{\mathfrak{S}}$, $\mathfrak{s}$ must have two endpoints on its boundary—to be precise, we consider the boundary of $\mathfrak{s}$ in $\mathfrak{D}'$ whenever we think of the boundary of such a component of $\mathfrak{S}$. With this in mind, we focus on each component of $\mathfrak{S}$ with a vertex of G on its boundary and call it a *region*. Observe that a crossing in D which does not involve a frame edge corresponds, in $D_{\mathfrak{S}}$, to a pair of edges where one goes over a handle and the other goes underneath.

2.2 Recognizing a graph with k bundled crossings

Consider a bundled circular drawing D of G consisting of one bundled crossing. The bundled crossing consists of two bundles, so we have up to four frame edges, whose set will be denoted by $\mathcal{F}$. By $V(\mathcal{F})$, we denote the set of vertices incident to frame edges. Via the construction above, we obtain the subdivided surface $\mathfrak{S}$; see Fig. 3. Let r_1 and r_2 be the regions each bounded by the pair of frame edges corresponding to one of the bundles, and let $r_3, \ldots, r_6$ be the regions each bounded by one edge from one pair and one from the other pair; see Fig. 3. These are all the regions of $\mathfrak{S}$. Since, as mentioned before, each of the non-frame edges of G (i.e., each $e \in E(G) \setminus \mathcal{F}$) along with two endpoints are contained in exactly one of these regions, each component of $G \setminus V(\mathcal{F})$ including the edges connecting it to vertices of $V(\mathcal{F})$ is drawn in $D_{\mathfrak{S}}$ in some region of $\mathfrak{S}$. In this sense, for each region r of $\mathfrak{S}$, we use G_r to denote the subgraph of G induced by the components of $G \setminus V(\mathcal{F})$ contained in r in $D_{\mathfrak{S}}$ including the edges connecting them to elements of $V(\mathcal{F})$. Additionally, each vertex of G is incident to an edge in $\mathcal{F}$ (in which case it is on the boundary of at least two regions) or it is on the boundary of exactly one region.

Notice that there are two types of regions: $\{r_1, r_2\}$ and $\{r_3, r_4, r_5, r_6\}$. Consider a region of the first type, for example r_1, and note that it is a topological disk[1], i.e., G_{r_1} is outerplanar. Moreover, it has a special drawing where the two frame edges e_1 and e_3 bounding the region r_1 are on the outerface. Now, consider adding a new vertex w_j, for $j = 1, 3$ adjacent to both u_j and v_j so that w_j is placed slightly outside of the region; see Fig. 3. Denote the resulting augmented graph by $G^*_{r_1}$ and the corresponding drawing by

[1] We slightly abuse this notion to also mean a simply connected set.

$D^*_{r_1}$ – it is easy to see that $D^*_{r_1}$ is outerplanar. Moreover, in every outerplanar embedding of $G^*_{r_1}$, the vertices u_j, w_j, v_j, $j = 1, 3$, occur consecutively on the outerface.

Similarly for a region of the second type, for example r_3, the graph G_{r_3} is outerplanar also with a special drawing where all the vertices must be on the arc u_3u_2 of the disk subtended by the two frame edges e_3 and e_2 bounding the region r_3. We construct the augmented graph $G^*_{r_3}$ by adding to G_{r_3} an edge u_3u_2 and adding a vertex w adjacent to both u_3 and u_2. Again, $G^*_{r_3}$ is outerplanar as r_3 is a topological disk. Moreover, in every outerplanar embedding of $G^*_{r_3}$, the vertices u_3, w, u_2 occur consecutively on the outerface.

In other words, G_{r_i} "fits" into r_i because its augmented graph $G^*_{r_i}$ is outerplanar ($\star$) – note: that we do not require the specific outerplanar embedding of G_{r_i} for this augmentation.

To sum up, G has a circular drawing D with at most one bundled crossing, because there exist (i) a set of $\beta \leq 4$ frame edges $\mathcal{F} = \{e_1, e_2, \ldots, e_\beta\}$, (ii) a particular circular drawing $D_\mathcal{F}$ of frame edges, (iii) the drawing of the one bundled crossing B, and (iv) corresponding regions $r_1, \ldots, r_\gamma (\gamma \leq 6)$ of the subdivided surface $\mathfrak{S}$ such that the following properties hold:

1. The set of edges $E(G)$ is partitioned into $E_0, E_1, \ldots, E_\gamma$.
2. There is a bijection from E_0 to $\mathcal{F}$ so that the subgraph of G formed by E_0 is isomorphic to the graph formed by $\mathcal{F}$.
3. No vertex in $V(G) \setminus V(E_0)$ has incident edges $e \in E_i$, $e' \in E_j$ for $i \neq j$.
4. For each $v \in V(E_0)$, and each edge e incident to v, exactly one of the following is true: (i) $e \in E_0$ or (ii) $e \in E_i$ and v is on the boundary of r_i.
5. For each region r_i, let G_i be the graph formed by E_i and vertices in $V(E_0)$ on the boundary of r_i (even if they are not incident to an edge in E_i), and let G^*_i be the corresponding augmented graph (i.e., as in $\star$ above). Then, G^*_i must be outerplanar.

To test for a drawing with one bundled crossing, we first enumerate drawings $D_\mathcal{F}$ of up to four lines in the circle. For each drawing $D_\mathcal{F}$ that is valid for frame edges of one bundled crossing, we define our surface and its regions (which will allow the augmentation to be well-defined). Then, we will build an MSO_2 formula to express Properties 1–5 above. We have intentionally phrased these properties in a logical way so that it is clear that they are expressible in MSO_2. The only condition which is not obviously expressible is the outerplanarity check. For this, we recall that outerplanarity is characterized by two forbidden minors (i.e., K_4 and $K_{2,3}$) [6] and that, for every fixed graph H, there is an MSO_2 formula $\textsc{minor}_H$ so that for all graphs G, $G \models \textsc{minor}_H$ if and only if G contains H as a minor [8, Corollary 1.15]. Thus, Properties 1–5 can be expressed as an MSO_2 formula ψ and, by Courcelle's theorem, there is a computable function f such that we can test if $G \models \psi$ in $O(f(\psi, t)n)$ time for input graphs of treewidth at most t. Since outerplanar graphs have treewidth at most two [18], the region graphs are outerplanar, and adding the (up to) 8 frame vertices raises the treewidth by at most 8, G must also have treewidth at most 10.

The key ingredient above was that every region was a topological disk. However, in a subdivision $\mathfrak{S}$ constructed from a bundled drawing with k bundled crossings this is not trivial as regions can go over and under many handles; see Fig. 2. We state this result in the following lemma, whose proof can be found in the full version [5].

▶ **Lemma 3.** *Each region r of $\mathfrak{S}$ is a topological disk.*

Properties 1–5 and Lemma 3 allow us to construct an MSO_2 formula φ to test whether $\mathrm{bc}^\circ(G) \leq k$ with pattern $\mathcal{F}$ of frame edges. The size of φ depends only on k. This lemma further implies that the treewidth of a graph G with $\mathrm{bc}^\circ(G) \leq k$ is at most $8k + 2$ since deleting a vertex from a graph lowers its treewidth by at most one and the treewidth of an

outerplanar graph is at most two [18]. So, applying Courcelle's Theorem (2) on φ of each pattern $\mathcal{F}$ leads to an FPT algorithm to test whether $\mathrm{bc}^\circ(G) \leq k$. This proves Theorem 1.

References

1 M. Alam, M. Fink, and S. Pupyrev. The bundled crossing number. In Y. Hu and M. Nöllenburg, editors, *GD*, volume 9801 of *LNCS*, pages 399–412. Springer, 2016. URL: http://arxiv.org/abs/1608.08161, doi:10.1007/978-3-319-50106-2_31.

2 M. J. Bannister and D. Eppstein. Crossing minimization for 1-page and 2-page drawings of graphs with bounded treewidth. *Journal of Graph Algorithms and Applications*, 22(4):577–606, 2018. doi:10.7155/jgaa.00479.

3 C. Buchheim, D. Ebner, M. Jünger, G. W. Klau, P. Mutzel, and R. Weiskircher. Exact crossing minimization. In P. Healy and N. S. Nikolov, editors, *GD*, volume 3843 of *LNCS*, pages 37–48. Springer, 2006. doi:10.1007/11618058_4.

4 S. Chaplick, M. Kryven, G. Liotta, A. Löffler, and A. Wolff. Beyond outerplanarity. In F. Frati and K.-L. Ma, editors, *GD*, volume 10692 of *LNCS*, pages 546–559. Springer, 2018. URL: http://arxiv.org/abs/1708.08723, doi:10.1007/978-3-319-73915-1_42.

5 S. Chaplick, T. C. van Dijk, M. Kryven, J. won Park, A. Ravsky, and A. Wolff. Bundled crossings revisited. Arxiv report, 2018. URL: http://arxiv.org/abs/1812.04263.

6 G. Chartrand and F. Harary. Planar permutation graphs. *Annales de l'I.H.P. Probabilités et statistiques*, 3(4):433–438, 1967. URL: http://eudml.org/doc/76875.

7 B. Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Inform. Comput.*, 85(1):12–75, 1990. doi:10.1016/0890-5401(90)90043-H.

8 B. Courcelle and J. Engelfriet. *Graph Structure and Monadic Second-Order Logic: A Language-Theoretic Approach*. Cambridge Univ. Press, 2012.

9 W. Cui, H. Zhou, H. Qu, P. C. Wong, and X. Li. Geometry-based edge clustering for graph visualization. *IEEE Trans. Vis. Comput. Graph.*, 14(6):1277–1284, 2008. doi:10.1109/TVCG.2008.135.

10 M. Fink, J. Hershberger, S. Suri, and K. Verbeek. Bundled crossings in embedded graphs. In E. Kranakis, G. Navarro, and E. Chávez, editors, *LATIN*, volume 9644 of *LNCS*, pages 454–468. Springer, 2016. doi:10.1007/978-3-662-49529-2_34.

11 M. Fink, S. Pupyrev, and A. Wolff. Ordering metro lines by block crossings. *J. Graph Algorithms Appl.*, 19(1):111–153, 2015. doi:10.7155/jgaa.00351.

12 E. R. Gansner, Y. Hu, S. North, and C. Scheidegger. Multilevel agglomerative edge bundling for visualizing large graphs. In G. D. Battista, J.-D. Fekete, and H. Qu, editors, *PACIFICVIS*, pages 187–194. IEEE, 2011. doi:10.1109/PACIFICVIS.2011.5742389.

13 M. R. Garey and D. Johnson. Crossing number is NP-complete. *SIAM J. Algebr. Discrete Meth.*, 4:312–316, 1983. doi:10.1137/0604033.

14 D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Trans. Vis. Comput. Graph.*, 12(5):741–748, 2006. doi:10.1109/TVCG.2006.147.

15 C. Hurter, O. Ersoy, S. I. Fabrikant, T. R. Klein, and A. C. Telea. Bundled visualization of dynamicgraph and trail data. *IEEE Trans. Vis. Comput. Graphics*, 20(8):1141–1157, 2014. doi:10.1109/TVCG.2013.246.

16 C. Hurter, O. Ersoy, and A. Telea. Graph bundling by kernel density estimation. *Comput. Graph. Forum*, 31:865–874, 2012. doi:10.1111/j.1467-8659.2012.03079.x.

17 K. Kawarabayashi and B. Reed. Computing crossing number in linear time. In *STOC*, pages 382–390. ACM, 2007. doi:10.1145/1250790.1250848.

18 S. L. Mitchell. Linear algorithms to recognize outerplanar and maximal outerplanar graphs. *Inform. Process. Lett.*, 9(5):229–232, 1979. doi:10.1016/0020-0190(79)90075-9.

19 S. Pupyrev, L. Nachmanson, S. Bereg, and A. E. Holroyd. Edge routing with ordered bundles. In M. van Kreveld and B. Speckmann, editors, *Proc. 19th Int. Symp. Graph Drawing (GD'11)*, volume 7034 of *LNCS*, pages 136–147. Springer, 2012. doi:10.1007/978-3-642-25878-7_14.

20 M. Schaefer. The graph crossing number and its variants: A survey. *Electr. J. Combin.*, Dynamic Survey DS21, 2017. URL: http://www.combinatorics.org/ojs/index.php/eljc/article/view/DS21.

Simplification of Polyline Bundles

Joachim Spoerhase[1], Sabine Storandt[2], and Johannes Zink[3]

1 **Aalto University, Finland**
`joachim.spoerhase@aalto.fi`
2 **AG Algorithmik, Universität Konstanz.**
`storandt@inf.uni-konstanz.de`
3 **Lehrstuhl für Informatik I, Universität Würzburg.**
`zink@informatik.uni-wuerzburg.de`

Abstract

We propose a generalization to the well-known problem of polyline simplification in two variants. We are given, instead of a single polyline, a set of polylines possibly sharing some edges and vertices. We show that this more general problem is *NP*-hard by a reduction from Max-2-SAT. On the positive side, we show fixed-parameter tractability in the number of shared vertices.

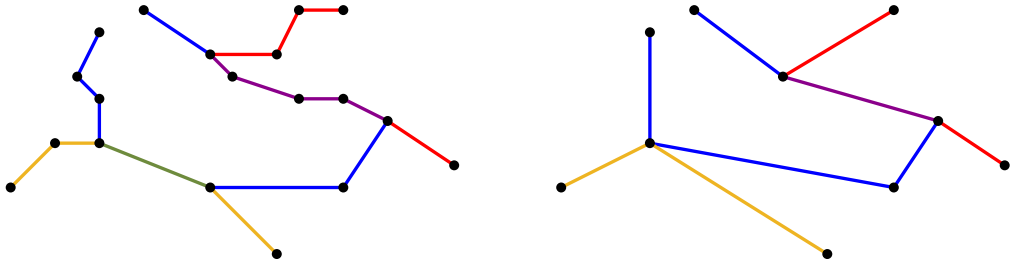
1 Introduction

Visualization of geographical information is a task of high practical relevance, e.g., for the creation of online maps. Such maps are most helpful if the information is neatly displayed and can be grasped quickly and unambiguously. This means that the full data often needs to be filtered and abstracted. Many important elements in maps like borders, streets, rivers, or trajectories are displayed as polylines (also known as polygonal chains). For such a polyline, a simplification is supposed to be as sparse as possible and as close to the original as necessary. A simplified polyline is constructed by a subset of vertices of the original polyline such that the (local) distance to the original polyline does not exceed a specifiable value according to a given distance measure, e.g., the Hausdorff distance [4] or the Fréchet distance [1]. The first such algorithm, which is still of high practical importance, was proposed by Ramer [7] and by Douglas and Peucker [3]. Hershberger and Snoeyink [5] proposed an implementation of this algorithm that runs in $O(n \log n)$ time, where n is the number of vertices in the polyline. It is a heuristic algorithm as it does not guarantee optimality (or something close to it) in terms of retained vertices. An optimal algorithm in this sense was first proposed by Imai and Iri [6]. Chan and Chin [2] improved the running time of this algorithm to $O(n^2)$.

From a Single Polyline to a Bundle of Polylines

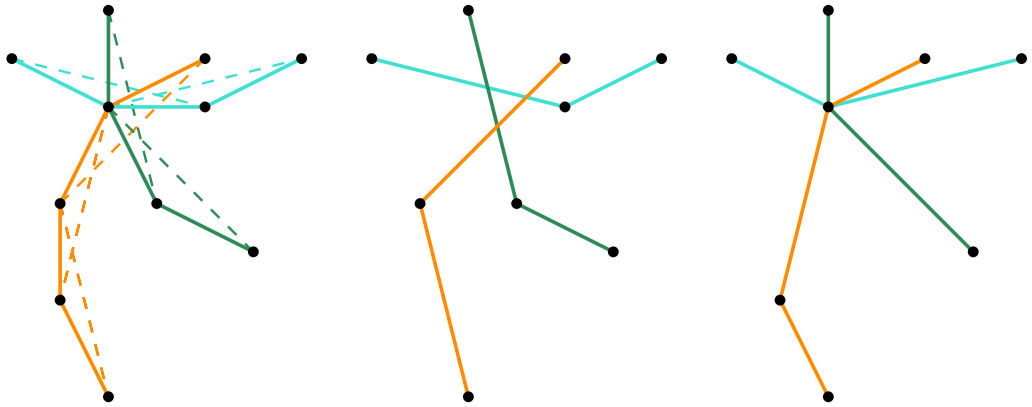
On a map, there are usually multiple polylines to display. Such polylines may share vertices and edges sectionwise. For example, when considering (GPS) trajectories like car-routes, different trajectories may partially share edges and vertices when cars have been on the same roads. Another example is a schematic map of a public transport network. Bus lines are the polylines and the vertices are the stations. In the city center, there are many different bus lines at the same stations that fan out when going to the outer districts, where they possibly share stations with further different bus lines. One might consider simplifying the polylines of a bundle independently. This has some drawbacks, though. On the one hand, the total complexity might even increase when the shared parts are simplified in many different ways. On the other hand, it might suggest a misleading picture when we remove common edges and vertices of some polylines, but not of all. The viewer might get the wrong impression that the one route has taken some street or passed through some area and the other has not, while in reality both took the same route in this place. E.g., if there is only one way to

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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) initial bundle of polylines **(b)** simplified bundle of polylines

Figure 1 Example of a bundle of three polylines before and after the simplification.



(a) initial bundle with shortcuts **(b)** optimal for Min-Edges **(c)** optimal for Min-Vertices

Figure 2 Example of three polylines, where the goals Min-Edges and Min-Vertices differ.

pass through some point, then the simplifications of all polylines going through this point should still share the corresponding vertex or edge if it is kept. Therefore, we require that a vertex in a simplification of a bundle of polylines is either kept for all polylines containing it or discarded in all polylines. In Figure 1, we give an example of a simplification of a bundle of polylines. Natural minimization goals are to minimize either the total number of vertices (Min-Vertices) or the total number of line segments, i.e., edges (Min-Edges). Both goals generalize the previously described minimization problem for a single polyline. However, they may differ from each other like in Figure 2. In this extended abstract, we focus on Polyline-Bundle-Simplification with the goal Min-Edges to be formalized next. With small adaptions, our results also hold for the goal Min-Vertices.

2 Problem Definition

In an instance of the problem Polyline-Bundle-Simplification with goal Min-Edges, we are given a set $V = \{v_1, \ldots, v_n\}$ of n points in the plane, and a set $\mathcal{L} = \{L_1, \ldots, L_\ell\}$ of ℓ polygonal chains $L_i = (s_i, \ldots, t_i)$ represented as lists of vertices from V, as well as a distance parameter ϵ referring to a distance measure d (e.g., Hausdorff distance). The goal is to obtain a subset $V^* \subseteq V$ of the points, such that for each L_i its induced simplification S_i, which is $L_i \cap V^*$ while preserving the order of vertices,

- contains the start and the end vertex of L_i, i.e., $s_i, t_i \in S_i$, and
- has at most a distance of ϵ to L_i, i.e., for each line segment (a, b) of S_i and the

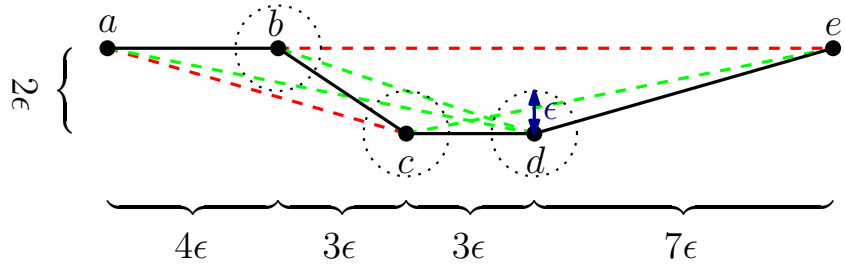


Figure 3 The literal-gadget depicted in black with valid (invalid) shortcuts in green (red).

corresponding sub-polyline of L_i from a to b, abbreviated by $L_i[a, \dots, b]$, we have $d((a, b), L_i[a, \cdots, b]) \leq \epsilon$,

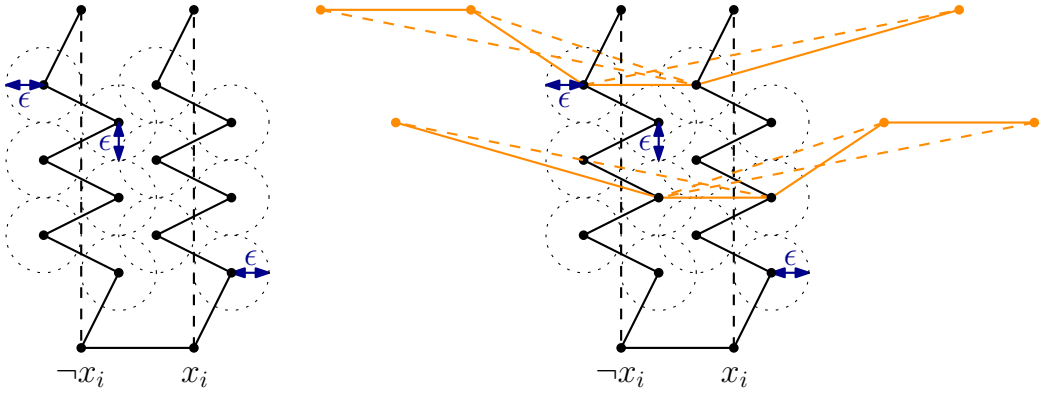
and the total number of edges induced by V^* is minimized. Here an edge (v, w) is induced by V^* if there exists at least one polyline L_i with $L_i[v, \dots, w] \cap V^* = \{v, w\}$. Note that we do not count the edge multiple times if there are multiple such polylines.

3 NP-Hardness

The problem Polyline-Bundle-Simplification with goal Min-Edges is *NP*-hard even for only two polylines (hence not *FPT* in ℓ unless $P = NP$). We show this by a reduction from Max-2-SAT, which is known to be *NP*-complete. In this reduction, we model a given 2-SAT formula by an instance of Polyline-Bundle-Simplification with goal Min-Edges using two polylines. Our reduction uses three types of gadgets, which allow some shortcuts inside the gadgets: *literal-gadgets*, *variable-synchronization-gadgets*, and *clause-synchronization-gadgets*. We obtain the first polyline by connecting all literal-gadgets such that no new shortcuts are possible. The second polyline is the connection of all variable-synchronization-gadgets and all clause-synchronization-gadgets such that no new shortcuts are possible. Next, we specify the three gadgets of our reduction.

Literal-Gadget. We model each literal of each clause by a *literal-gadget*. In Figure 3, a literal-gadget for modeling a positive literal x is depicted. For negative literals, it is the same but mirrored horizontally. It consists of five serially connected vertices (drawn in black). Valid shortcuts are dashed in green, invalid shortcuts in red. The vertices a and e cannot be skipped and the inner vertices b, c, and d are shared with the second polyline. There are three mutually exclusive shortcuts: skipping b and c, skipping c, and skipping d. Skipping c (together with or without b) or d is always possible and corresponds to the truth assignment of this clause. Since the number of edges is minimized, the shortcut that skips b and c will be chosen whenever possible (in compliance with the variable-synchronization-gadget, with which c and d are shared, and the clause-synchronization-gadget, with which b is shared). The interpretation is as follows: if c is skipped, x is set to true; if d is skipped, x is set to false; if b is skipped, this literal satisfies its clause. So b is the "critical" vertex indicating that a clause is satisfied. In a literal-gadget for a negative literal, b is between d and e, and not between a and c. Clearly, all vertices lie on a grid point of a grid with square length ϵ.

Variable-Synchronization-Gadget. For each variable, we use a *variable-synchronization-gadget* to enforce a consistent truth assignment for a variable x_i. In Figure 4, a variable-synchronization-gadget for synchronizing six literal-gadgets is depicted. Shortcuts are depicted as dashed segments in the color of its polyline. The number of vertices in the



(a) The gadget alone.

(b) The combination of a variable-synchronization-gadget (black) and literal-gadgets (orange). Only 2 of 6 literal-gadgets are drawn here.

Figure 4 The variable-synchronization-gadget.

section going zigzag corresponds to the number of occurrences of this variable—regardless of positive or negative. Except for the two vertices on the top and the two vertices on the bottom of the gadget, all vertices are shared with the literal gadgets—each two vertices with the same y-coordinate are part of the same literal gadget (the vertices c and d in Figure 3). There are only two shortcuts: skipping all shared left vertices and skipping all shared right vertices. The interpretation is as follows: if we skip the left vertices and keep the right vertices, x_i is set to true and the other way round x_i is set to false. An inconsistent assignment is not possible: we cannot take both shortcuts, since we cannot skip both lower vertices in a literal gadget. Taking none of these shortcuts would violate the minimality, since consistently skipping the same lower vertex in the literal gadgets is always possible (otherwise we cannot take any shortcut of the concerned literal-gadgets). Again, all vertices lie on a grid point of a grid with square length ϵ.

Clause-Synchronization-Gadget. We use a *clause-synchronization-gadget* for each clause with two literals. Its purpose is to reward satisfied clauses uniformly, i.e., it prohibits "double" satisfied clauses from being rewarded better than "once" satisfied clauses. In Figure 5, a clause-synchronization-gadget is depicted in black. It consists of four serially connected vertices, which connect two literal-gadgets (gray color) that correspond to two literals of the same clause. The inner vertices b_1 and b_2 are shared with the two b-vertices of these literal-gadgets (compare with Figure 3). Valid shortcuts are dashed in green, invalid shortcuts in red. There are two mutually exclusive shortcuts: skipping b_1 and skipping b_2. If one of them is used, then the b-vertex of one literal-gadget is skipped. This is only possible when the assigned truth value satisfies the corresponding literal, which in turn satisfies the corresponding clause. We can say: for each satisfied clause, we get the reward of reducing the total number of edges by two when we skip such a b-vertex (the one edge in the literal-gadget, the other edge in the clause-synchronization-gadget), which we cannot remove otherwise. Since there is no shortcut from s to t, it is not possible to skip both b-vertices corresponding to the same clause and, therefore, also not possible to get a greater reward if both literals of the same clause are set to true. Since we minimize the number of remaining edges, as many clauses as possible are satisfied this way because if at least one of the corresponding literal-gadgets is set true, we clearly can also skip the b-vertex of this literal-gadget. Hence, only if none of the two b-vertices is skipped, the corresponding clause remains unsatisfied.

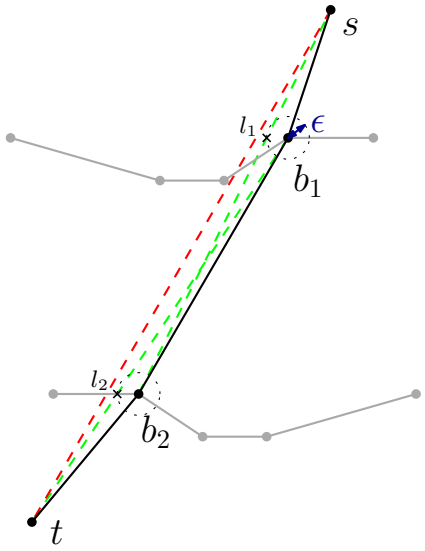


Figure 5 The clause-synchronization-gadget.

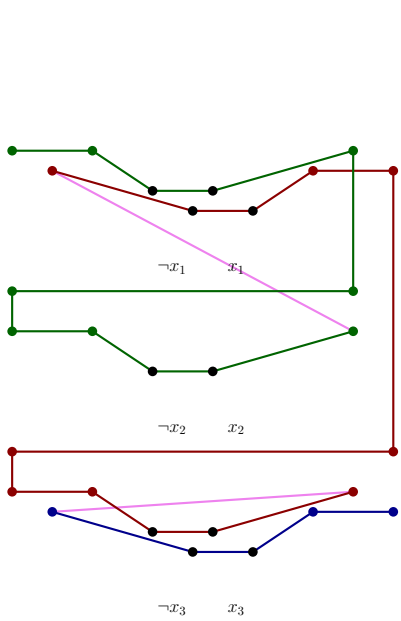
We can construct this gadget such that its vertices are on grid points of a polynomial-size grid with square length ϵ. The vertices b_1 and b_2 are already on the grid. Consider the grid points l_1, l_2 to the left of them with distance ϵ. Our shortcuts $s \to b_2$ and $b_1 \to t$ will lie on the lines defined by $\overline{l_1 b_2}$ and $\overline{b_1 l_2}$, respectively. Lengthen the line segments $\overline{l_1 b_2}$ and $\overline{b_1 l_2}$ at l_1 and l_2 by a factor of 2 (we use 1.5 in Figures 5 and 6 to keep it overseeable)—the endpoints are the grid points that will be our s and t. Observe that the shortcut $s \to t$ is always invalid if we place all variable-synchronization-gadgets in a column above the other with sufficient vertical spacing (constant in ϵ).

Complete Reduction

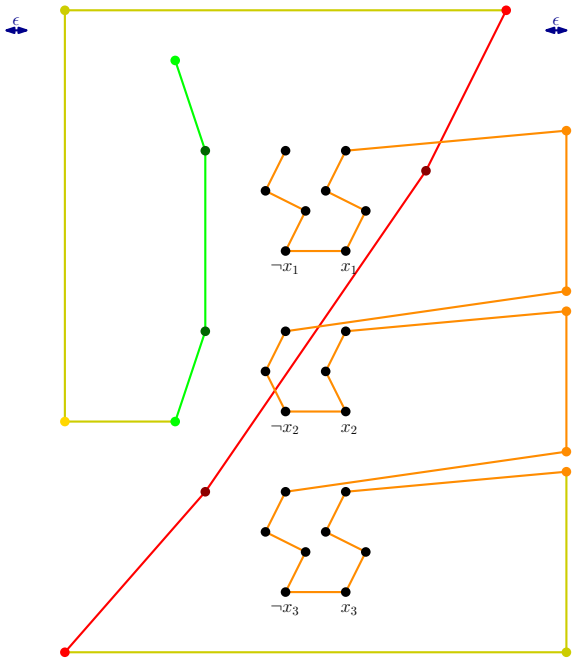
Given a Max-2-SAT instance, we can reduce it in polynomial time to a Polyline-Bundle-Simplification instance with goal Min-Edges: set ϵ to 1 and create for each variable a variable-synchronization-gadget with size equal to the number of occurrences of this variable and place them one above the other onto an integer grid. This defines exact positions for the literal-gadgets and then for the clause-synchronization-gadgets. Connect all literal-gadgets (first polyline) and, separately, all variable-synchronization-gadgets and clause-synchronization-gadgets (second polyline) in a shortcut-free way. This is possible on a polynomial-size grid. From a solution minimizing the number of edges of this polyline bundle simplification instance with two polylines, we can immediately obtain a solution of the corresponding Max-2-SAT instance—the total number of removed b-vertices equals the maximum number of satisfiable clauses. Thus, we conclude the following theorem:

▶ **Theorem 3.1.** Polyline-Bundle-Simplification *with goal* Min-Edges *is NP-hard even for two polylines.* ◀

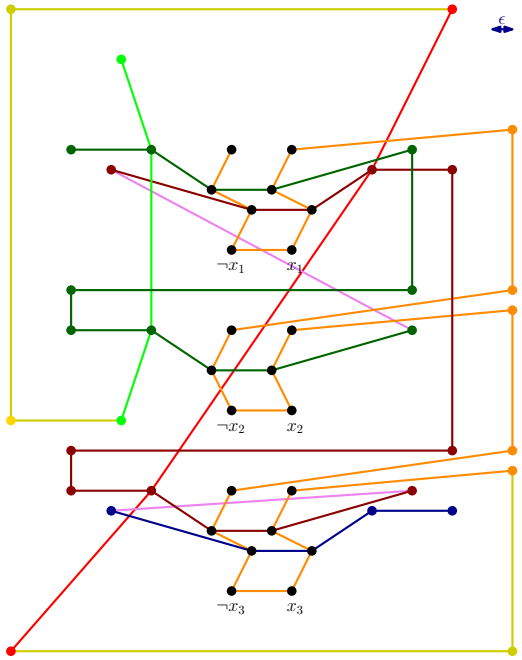
We give a small but full example to the presented reduction in Figure 6. We use the 2-SAT formula $(x_1 \vee x_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_3)$. Note that the last clause $(\neg x_3)$ consists of only one literal and, therefore, does not have a clause-synchronization-gadget.



(a) first polyline (connecting literal-gadgets)



(b) second polyline (connecting both types of synchronization-gadgets)



(c) both polylines

Figure 6 Full example of our *NP*-hardness reduction: $(x_1 \vee x_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_3)$

4 Fixed-Parameter Tractability

For both goals (Min-Edges or Min-Vertices) our problem is fixed-parameter tractable in the number of shared vertices, that is, vertices contained in more than one polyline or multiple times in the same polyline. We call the set of those vertices V_{shared} and let $k := |V_{shared}|$.

▶ **Theorem 4.1.** Polyline-Bundle-Simplification *is fixed-parameter tractable in* k.

Proof sketch. We sketch an algorithm that solves Polyline-Bundle-Simplification in $O(2^k \cdot \ell n^3)$ time. The idea is to fix for each subset $V' \subseteq V_{shared}$ the vertices in V' to be contained in V^* and the vertices in $V_{shared} \setminus V'$ to be excluded from V^*. Then the optimal simplification of the remaining parts, which are simple polylines, can be computed in the classic way [6]. In the end, we take the best solution among all 2^k subsets of V_{shared}. ◀

5 Conclusion

We have generalized the well-known problem of polyline simplification from a single polyline to multiple interfering polylines. Unlike the special case of a single polyline, simplifying a bundle of polylines turned out to be *NP*-hard. The problem is fixed-parameter tractable in the number of shared vertices, but not in the number of polylines.

The *NP*-hardness result gives rise to the question of approximability. It can be shown that the reduction from Max-2-SAT gives even *APX*-hardness. Therefore, it is an interesting question if there is a constant-factor approximation algorithm for our problem.

References

1 Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry and Applications*, 5:75–91, 1995. URL: https://doi.org/10.1142/S0218195995000064, doi:10.1142/S0218195995000064.

2 W. S. Chan and F. Chin. Approximation of polygonal curves with minimum number of line segments or minimum error. *International Journal of Computational Geometry and Applications*, 6(1):59–77, 1996. URL: https://doi.org/10.1142/S0218195996000058, doi:10.1142/S0218195996000058.

3 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*, 10(2):112–122, 1973.

4 Felix Hausdorff. *Grundzüge der Mengenlehre*. Veit and Company, Leipzig, 1914. URL: https://archive.org/details/grundzgedermen00hausuoft.

5 John Hershberger and Jack Snoeyink. Speeding up the douglas-peucker line-simplification algorithm. In *Proc. 5th Intl. Symp. Spatial Data Handling (SDH'92)*, pages 134–143. IGU Commission on GIS, 1992.

6 Hiroshi Imai and Masao Iri. Polygonal approximations of a curve – formulations and algorithms. In Godfried T. Toussaint, editor, *Computational Morphology*, volume 6 of *Machine Intelligence and Pattern Recognition*, pages 71 – 86. North-Holland, 1988. URL: http://www.sciencedirect.com/science/article/pii/B9780444704672500114, doi:https://doi.org/10.1016/B978-0-444-70467-2.50011-4.

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

Shooting Stars in Simple Drawings of $K_{m,n}$*

Oswin Aichholzer[1], Irene Parada[1], Manfred Scheucher[2], Birgit Vogtenhuber[1], and Alexandra Weinberger[1]

1 **Graz University of Technology, Austria**
[oaich|iparada|bvogt]@ist.tugraz.at, alexandra.weinberger@student.tugraz.at
2 **Institut für Mathematik, Technische Universität Berlin, Germany**
[scheucher]@math.tu-berlin.de

Abstract

In this work we study the existence of plane spanning trees in simple drawings of the complete bipartite graph $K_{m,n}$. We show that every simple drawing of $K_{2,n}$ and $K_{3,n}$, $n \geq 1$, as well as every outer drawing of $K_{m,n}$ for any $m, n \geq 1$, contains plane spanning trees. Moreover, for all these cases we show the existence of special plane spanning trees, which we call shooting stars. Shooting stars are spanning trees that contain the star of a vertex, i.e., all its incident edges.

1 Introduction

In a drawing of a graph in the Euclidean plane the vertices are drawn as distinct points and the edges are drawn as continuous arcs connecting its two end points. Depending on which properties of the graph are to be considered, there might be additional requirements on how the graph is drawn. Typically the drawing of an edge has to be non-self-crossing and must not pass through any point representing a vertex other than its two end points. In addition, in a *simple drawing* of a graph any pair of edges crosses at most once, either in their interior or at a common end point, no tangencies are allowed and no three edges pass through a single crossing. These drawings are also called *good drawings* [1, 3] or *(simple) topological graphs* [5, 6].

The probably most restricted version of drawings are straight-line drawings, also called *geometric graphs*, where an edge is drawn as straight-line segment connecting its two end points. Thus, the placement of the vertices in the plane entirely determines the full drawing.

Both classes of drawings are of special interest if we want to minimize the number of crossings in a drawing of a given graph. If such a drawing does not contain any crossing at all then it is called *plane*. In this work we are interested in *plane spanning subdrawings* of a given drawing, that is, drawings without crossings that contain all the vertices of the given drawing and a subset of its edges.

The existence of plane subdrawings of simple drawings of the complete graph K_n has received quite a lot of attention. For example, Ruiz-Vargas [9] showed that every simple drawing of K_n contains $\Omega(n^{1/2-\epsilon})$ pairwise disjoint edges for any $\epsilon > 0$, by this improving over many previous bounds [6, 7, 10]. Fulek and Ruiz-Vargas [4] proved that given a simple drawing of K_n, a plane cycle C in the drawing, and any vertex v that is not part of C, at least two edges connecting v to C do not intersect C. Hence every simple drawing of K_n contains a plane sub-drawing with at least $2n - 3$ edges. Rafla [8] conjectured that every simple drawing of K_n contains a plane Hamiltonian cycle, a statement that is known to be

* O.A., M.S., and B.V. partially supported by the ESF EUROCORES programme EuroGIGA – CRP ComPoSe, Austrian Science Fund (FWF): I648-N18. I.P. supported by the Austrian Science Fund (FWF): W1230. M.S. partially supported by the Austrian Science Fund (FWF): P23629-N18 and the DFG Grant FE 340/12-1. B.V. and A.W. partially supported by the Austrian Science Fund within the collaborative DACH project *Arrangements and Drawings* as FWF project I 3340-N35.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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.

true for several classes of simple drawings (e.g., 2-page book drawings, monotone drawings, cylindrical drawings), but is still open in the general case. Pach et al. [6] proved that every simple drawing of K_n contains a plane drawing of any fixed tree with at most $c \log^{1/6} n$ vertices.

In this paper we concentrate on the existence of plane spanning trees in simple drawings. Obviously, any simple (or straight-line) drawing of the complete graph K_n contains some plane spanning trees: choose any vertex v and all the edges incident to v. As these edges cannot cross we obtain a plane spanning star.

For the complete bipartite graph $K_{m,n}$, the situation is less obvious. As a warm up exercise, let us first consider straight-line drawings of $K_{m,n}$. Let V_1 and V_2 be the sides of the bipartition of the vertex set of $K_{m,n}$. Chose any vertex $v_0 \in V_1$ and draw the star consisting of all edges v_0v with $v \in V_2$. This star induces a partition of the plane into wedges centered at v_0, where one wedge might have an opening angle larger than π. Draw a virtual angular bisector within each wedge and connect the vertices of V_1 which lie in each half of a wedge to the corresponding end point $v \neq v_0$ of the star edge. This results in a plane spanning tree with root v_0 and height 2 that includes all the edges incident to v_0. In the following, we call such a (not necessarily rectilinear) plane spanning tree a *shooting star (rooted at v_0)*.

For simple drawings, we are not aware of a similarly easy construction. Actually, it is still an open problem whether every simple drawing of $K_{m,n}$ contains a plane spanning tree. In this paper we solve that problem for the cases of $K_{2,n}$ and $K_{3,n}$ (see Section 2), as well as for *outer drawings* of $K_{m,n}$ (see Section 3, where also a definition of these drawings can be found). In all those cases, we show the existence of a shooting star rooted at any of the vertices of one side of the bipartition (the smaller one in case of $K_{2,n}$ and $K_{3,n}$ and the one lying on the outer boundary in the case of outer drawings).

2 Plane Spanning Trees in Simple Drawings of $K_{2,n}$ and $K_{3,n}$

In this section we prove that every simple drawing of $K_{2,n}$ and $K_{3,n}$ contains plane spanning trees of a certain structure. In order to do so, we introduce some notions and provide some auxiliary results.

For a given simple drawing of K_n with vertex set V and two fixed vertices $g \neq r \in V$, we define a relation $\rightarrow_{gr}$ on the remaining vertices $V \backslash \{g, r\}$, where $a \rightarrow_{gr} b$ if and only if the arc ra properly crosses gb. In the following, we simply write $a \rightarrow b$ if the two vertices g and r are clear from the context.

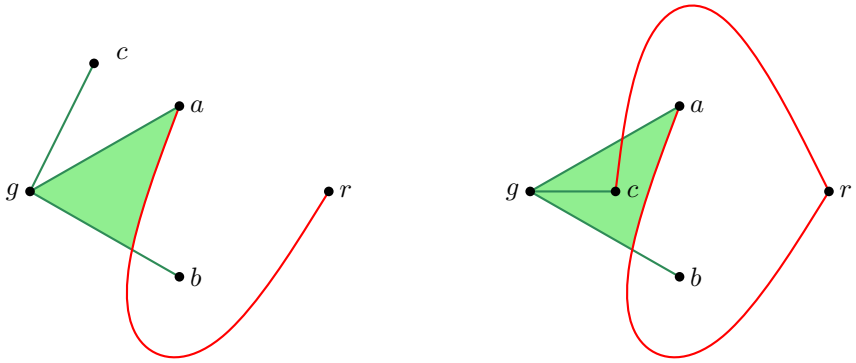
▶ **Lemma 2.1.** *The relation $\rightarrow$ is asymmetric and acyclic, that is, there are no vertices $v_1, v_2, \ldots, v_k$ ($k \in \mathbb{N}$) with $v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_k \rightarrow v_1$.*

Proof. We give a proof by induction on k.

Induction basis: The case $k = 1$ is trivial. The case $k = 2$ follows from the fact that there is at most one proper crossing in every 4-tuple in a simple drawing – if ra crosses gb then rb cannot cross ga. For the case $k = 3$ assume there are three vertices a, b, c with $a \rightarrow b \rightarrow c \rightarrow a$. Let $\triangle$ denote the area bounded by the edges ga, gb, ra and not containing the vertex r, as illustrated in Figure 1. We distinguish the following two cases:

Case 1: $c \notin \triangle$. Since $c \rightarrow a$ holds, the arc rc crosses ga, and therefore the boundary of $\triangle$. Since $r \notin \triangle$ and since rc cannot cross ra, rc must also cross gb. Thus we have $c \rightarrow b$, which is a contradiction to $b \rightarrow c$.

Case 2: $c \in \triangle$. Since $a \rightarrow b$, the arc rb cannot cross ga. Moreover, since rb can neither cross ra nor gb, it is therefore completely outside of $\triangle$. Since gc is completely contained in $\triangle$, rb and gc cannot cross, and therefore, $b \not\rightarrow c$. Contradiction.



(a) Case 1 **(b)** Case 2

Figure 1 An illustration of the two cases of base case $k = 3$ from Lemma 2.1. The area $\triangle$ is colored light green.

Since c can neither be inside nor outside $\triangle$, the statement is proven.

Induction step: Suppose – towards a contradiction – that there exist $v_1, \ldots, v_k$ with $k \geq 4$ and $v_1 \to v_2 \to \ldots \to v_k \to v_1$. We write $a = v_1$, $b = v_2$, $w = v_{k-1}$, and $z = v_k$. Let $\triangle$ denote the area bounded by the edges ga,gb, and ra that does not contain the vertex r. We distinguish the following two cases:

Case 1: $z \notin \triangle$. We continue analogously to Case 1 of base case $k = 2$. Since $z \to a$ holds, rz crosses ga, and therefore the boundary of $\triangle$. Since $r \notin \triangle$ and since rz cannot cross ra, rz must also cross gb. Thus we have $z \to b$.

Case 2: $z \in \triangle$. Since $w \to z$ holds, rw crosses gz at some point inside $\triangle$. Since $r \notin \triangle$ and since rw cannot cross ra, it must cross ga or gb (or both). Thus we have $w \to a$ or $w \to b$.

In both cases, we can find $v'_1, \ldots, v'_l$ for some $l < k$ with $v'_1 \to \ldots \to v'_l \to v'_1$, which is a contradiction. This completes the proof of the lemma. ◀

▶ **Theorem 2.2.** *Let D be a simple drawing of the complete bipartite graph $K_{2,n}$ with sides of the bipartition $\{g, r\}$ and P. Then, for every $k \in \{0, \ldots, n\}$, D contains a plane spanning tree with k edges incident to g and $n - k + 1$ edges incident to r.*

Proof. According to Lemma 2.1, we can find a labeling $v_1, \ldots, v_n$ of the vertices in P such that $v_i \to_{gr} v_j$ only holds if $i < j$. Let S_1 be the star with center g and children $\{v_1, \ldots, v_k\}$ and let S_2 be the star with center r and children $\{v_k, \ldots, v_n\}$. By definition of relation $\to_{gr}$, the edges of S_1 and S_2 do not cross, and hence we have a plane spanning tree. ◀

▶ **Corollary 2.3.** *Let D be a simple drawing of the complete bipartite graph $K_{2,n}$ with sides of the bipartition $\{g, r\}$ and P. Then for each $c \in \{g, r\}$, D contains a shooting star rooted at c.*

Proof. Consider again the proof of Theorem 2.2. With the according labeling of P, no edge rv_i can cross the edge gv_1. Hence, the plane spanning tree consisting of all the edges incident to r together with the edge gv_1 gives the desired shooting star rooted at r. Similarly, the tree with all edges incident to g and the edge rv_n is a shooting star rooted at g. ◀

We also have an analogous result, showing shooting stars also exist for simple drawings of $K_{3,n}$. Due to lack of space, we only state the theorem. Its proof is deferred to the foll version of this paper.

▶ **Theorem 2.4.** *Let D be a simple drawing of the complete bipartite graph $K_{3,n}$ with sides of the bipartition $\{g, r, b\}$ and P. Then for each $c \in \{g, r, b\}$, D contains a shooting star rooted at c.*

3 Shooting Stars in Outer Drawings of $K_{m,n}$

In this section we will study the problem of finding plane spanning trees in a special kind of simple drawings of bipartite graphs, namely outer drawings. The concept of outer drawings was recently introduced in [2]. They are defined as follows:

▶ **Definition 3.1.** A simple drawing of a $K_{m,n}$ in which all the m vertices of one side of the bipartition lie on the outer boundary of the drawing is called *outer drawing*.

We denote by P the side of the bipartition whose vertices must lie on the outer boundary of the drawing. The other side of the bipartition is denoted by S. Note that points of S may also lie on the outer boundary but don't have to.

We now proceed to prove that there is a plane spanning tree in every outer drawing of a $K_{m,n}$ with $m, n \in \mathbb{N}$.

▶ **Theorem 3.2.** *Let D be an outer drawing of the complete bipartite graph $K_{m,n}$ with sides of the bipartition P and S where the vertices of P lie on the outer boundary. Let p be an arbitrary vertex in P. Then D contains a shooting star rooted at p.*

Proof. First, we label the vertices in P. We start in $p_1 := p$ and go clockwise along the outer boundary and denote the vertices of P by p_2 to p_m following the order in which they occur. Let T_1 be the sub graph that is induced by all edges incident to p_1. Notice that T_1 is a plane tree. We will add edges to T_1 until it becomes a spanning tree. We do so inductively by first adding an edge incident to p_2, then an edge incident to p_3 and so on until we add an edge incident to the vertex p_m. We denote by T_i the tree that we get by adding to T_{i-1} the selected edge incident to p_i for $2 \leq i \leq m$. We will show that it is possible to add edges such that T_i is always plane. After adding the last edge the statement then follows.

In the first step, for T_2, we need to find an edge that is incident to p_2 and does not cross any edge incident to p. We know from Theorem 2.2 that there is at least one such edge. We add that edge to T_1 and get a plane tree T_2. For T_i we need to find an edge that is incident to p_i and does not cross any of the edges of T_{i-1}. We denote by e_{i-1} the edge in T_{i-1} that is incident to p_{i-1} and by s_{i-1} the vertex in S that e_{i-1} is incident to. We also denote the edge that is incident to s_{i-1} and p by e_p. See Figure 2 for an illustration. The part of the boundary that goes from p clockwise until p_{i-1} together with the edges e_{i-1} and e_p encloses a region that we call R_1. The vertices p_2 to p_{i-1} all lie on that part of the boundary, because of the way we labeled them. We call the rest of the area inside the outer boundary R_2.

▶ Claim 1. All edges in T_{i-1} that are not incident to p lie completely inside R_1.

Proof. Since the boundary of R_1 consists of edges in T_{i-1} and the outer boundary, all edges of T_{i-1} that lie partly inside R_2 have to lie completely inside it. The edges in T_{i-1} that are not incident to p are incident with the vertices p_2 to p_{i-1}. As they have to lie on the part of the outer boundary that is also part of the boundary of R_1, the edges incident to these vertices have to lie partly inside R_1. Thus these edges have to lie completely inside R_1. ◀

Let us now consider the region R_2. The sub graph induced by p, p_i, and all vertices of S that lie in R_2 is a $K_{2,n'}$ with $n' \in \mathbb{N}$. By Theorem 2.2 there is an edge incident to p_i that does not cross any edges incident to p. This edge can neither cross the outer boundary nor

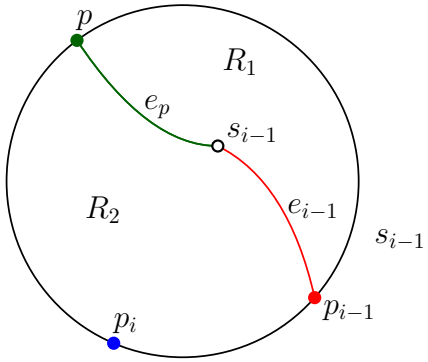


Figure 2 The edges e_p and e_{i-1} together with the outer boundary form two regions.

e_p and it can only cross e_{i-1} once. Since the edge has both end points in R_2, it follows that the edge has to lie completely in R_2. From Claim 1 it follows that it does not cross any of the edges of T_{i-1} that are not incident with p. As it doesn't cross any edges incident with p either, it follows that it doesn't cross any of the edges of T_{i-1}. Thus, we can add that edge and obtain a plane tree T_i. We continue to do so until we added an edge for every vertex in P. The plane spanning tree T_m is a shooting star. ◀

4 Conclusion

We have shown that particular cases of simple drawings of the complete bipartite graph $K_{m,n}$, namely all simple drawings of $K_{2,n}$ and $K_{3,n}$, $n \geq 1$, as well as all outer drawings of $K_{m,n}$ for any $n, m \geq 1$, contain plane spanning trees that are shooting stars. As we showed in the introduction, a similar result applies to straight-line drawings of $K_{m,n}$. In the full version of this paper we show that our results also extend to other classes of drawings of complete bipartite graphs. It is still an interesting open question whether every simple drawing of $K_{m,n}$ has a shooting star, or at least a plane spanning tree.

References

1 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.

2 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.

3 Paul Erdős and Richard K. Guy. Crossing number problems. *The American Mathematical Monthly*, 80(1):52–58, 1973. doi:10.2307/2319261.

4 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.

5 Jan Kynčl. Enumeration of simple complete topological graphs. *European Journal of Combinatorics*, 30:1676–1685, 2009. doi:10.1016/j.ejc.2009.03.005.

6 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.

7 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.

8 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.

9 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.

10 Andrew Suk. Disjoint edges in complete topological graphs. *Discrete & Computational Geometry*, 49(2):280–286, 2013. doi:10.1007/s00454-012-9481-x.

Extending Simple Drawings*

Alan Arroyo[1], Martin Derka[2], and Irene Parada[3]

1 **IST Austria, Klosterneuburg, Austria**
alanmarcelo.arroyoguevara@ist.ac.at
2 **Carleton University, Ontario, Canada**
mderka@uwaterloo.ca
3 **Graz University of Technology, Graz, Austria**
iparada@ist.tugraz.at

Abstract

Simple drawings are those in which (i) every pair of edges have at most one point in common, and it is either an endpoint or a proper crossing; and (ii) no three edges cross in the same point. In this paper we study the problem of extending a simple drawing $D(G)$ of a graph $G = (V, E)$, by adding a set of edges (of the complete graph with vertex set V) such that the result is a simple drawing with $D(G)$ as a subdrawing. In the context of rectilinear drawings, the problem is trivial. In contrast, we prove that finding the maximum amount of edges from a prescribed set that extend a simple drawing is NP-hard.

1 Introduction

A *simple drawing* of a graph G (also known as *good drawing* or as *simple topological graph* in the literature) is a drawing $D(G)$ of G in the plane such that every pair of edges share at most one point that is either a proper crossing (no tangent edges allowed) or an endpoint. Moreover, no three edges intersect in the same point and edges must not contain other vertices. In some contexts, such as the study of crossing numbers, simple drawings play a central role. Despite them being widely studied, there are basic aspects that remain unknown.

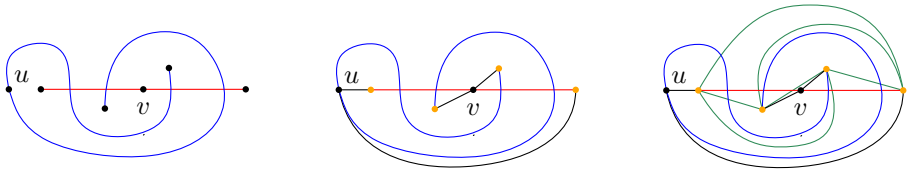
The long-standing conjectures on the crossing numbers of K_n and $K_{n,m}$, known as the Harary-Hill and Zarankiewicz's conjectures, respectively, have drawn particular interest in the study of simple drawings of complete and complete bipartite graphs. Although these problems remain open, their intensive study has produced deep results about simple drawings of K_n [6, 9] and $K_{n,m}$ [2].

In contrast to what we know about K_n, little is known about simple drawings of general graphs. In [8] it was observed that, when studying simple drawings of general graphs, it would be natural to try extend them, by adding the missing edges between non-adjacent vertices, to simple drawings of complete graphs. One of the main results in this paper suggests that there is no hope on efficiently deciding when such closure operation can be performed.

Given a simple drawing $D(G)$ of a graph $G = (V, E)$, and a set M of edges of the complete graph with vertex set V, an *extension* of $D(G)$ with a set of edges M is a simple drawing $D'(H)$ of the graph $H = (V, E \cup M)$ that contains $D(G)$ as a subdrawing. If that extension exists we say the the edge uv can be *added* to $D(G)$. An extension with one given edge is not always possible, as shown by Kynčl [7] (in Figure 1a the edge uv cannot be added). We can extend this example to a simple drawing of $K_{2,4}$ (Figure 1b) and we can use this to construct larger drawings of $K_{n,m}$ in which an edge uv cannot be added. Moreover, Kyncl's drawing can be extended to a simple drawing of K_6 missing an edge that cannot be added

* This work was started at the Crossing Numbers Workshop 2016 in Strobl (Austria). M.D. was partially supported by NSERC. I.P. is supported by the Austrian Science Fund (FWF): W1230.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 19–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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) Example by Kynčl [7]. **(b)** Drawing of $K_{2,4}$. **(c)** Drawing of $K_6 \setminus uv$.

Figure 1 Drawings that cannot be extended with the edge uv.

(Figure 1c), and again we can use this drawing to construct larger drawings of K_n missing an edge that cannot be added.

Extensions have been previously considered in the context of *saturated* simple drawings, that is, drawings where no edge can be added [8, 4]. In the context of saturated drawings, the main interest is on finding the minimum number of edges that a saturated graph on n vertices can have. This minimum was first shown to be at most $17.5n$ [8] and later $7n$ [4].

In this paper, we focus on extensions of simple drawings of general graphs. In Section 2 we show that given a simple drawing $D(G)$ of a graph $G = (V, E)$ and a set M of edges of the complete graph with vertex set V and with $M \cap E = \emptyset$, it is NP-hard to find the maximum subset of edges from M that can be added to $D(G)$. In the full version of the paper we also study the case in which only one edge is to be added. In Section 3 we discuss these results and present open questions.

2 Hardness of Extending Simple Drawings

In this section we prove the following result:

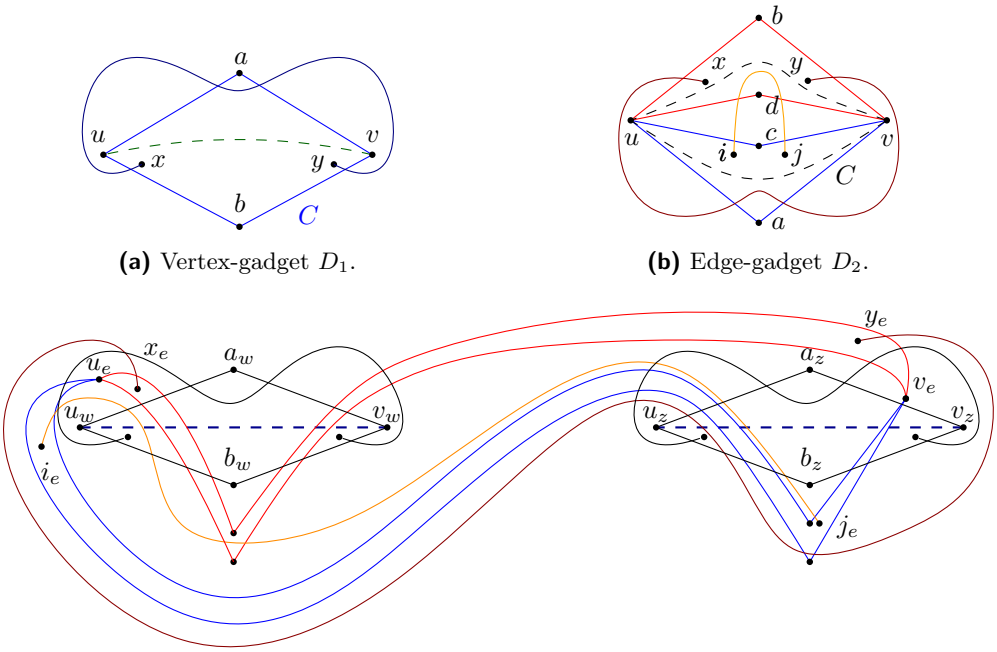
▶ **Theorem 2.1.** *Given a simple drawing $D(G)$ of a graph $G = (V, E)$ and a set M of edges of the complete graph with the vertex set V and with $E \cap M = \emptyset$, it is NP-hard to find a maximum subset of edges $M' \subseteq M$ that extends $D(G)$.*

Our proof of Theorem 2.1 is based on a reduction from the maximum independent set problem (MIS). An independent set of a graph $G = (V, E)$ is a set of vertices $S \subseteq V$ such that no two vertices in S are incident to the same edge. The problem of determining the maximum independent set (MIS) of a given graph is NP-hard in general, and it remains NP-hard when the input is a planar graph with maximum degree 3 [3, Lemma 1]. We first describe the construction of a simple drawing $D'(G')$ given an MIS instance. Then we argue that for a well selected set of edges M that are not present in $D'(G')$, finding a maximum subset $M' \subseteq M$ that can simultaneously extend $D'(G')$ is equivalent to finding a maximum independent set in the input instance.

2.1 Constructing a drawing from a given graph

We begin by introducing our two basic gadgets D_1 and D_2 (shown in Figure 2). The vertex gadget D_1 consists of a cycle C on four vertices a, v, b, u drawn in the plane without any crossings. We add two additional vertices x and y to its interior and connect them with an edge that, starting in x crosses edge bu to the exterior of C, continues through ua to the interior of C, crosses av to the exterior of C, and vb to the interior of C where it ends in y.

The drawing D_1 has the property that the only way of adding edge uv is by following an arc such as the dashed one depicted in Figure 2a (with maybe also crossing the edge xy, but



(a) Vertex-gadget D_1.

(b) Edge-gadget D_2.

(c) Constructed drawing obtained by a reduction from the path graph on two vertices w and z.

Figure 2 Basic gadgets and drawings.

staying in the same region). Routing through the exterior of C would force either a double crossing with edge xy, or a crossing with an edge incident to u or v.

The edge gadget D_2 is obtained by adding additional vertices to the interior of D_1. Specifically, we add two vertices d and c and edges ud, vd and uc, vc that are drawn so that $udvc$ is a crossing-free cycle in the interior of C. Note that since it is crossing-free, the vertices x and y are in the interior of cycle $ubvd$. We add two more vertices, called i and j, in the interior of $ucva$ and we connect them with an edge that after starting in i, crosses edges uc, ud, vd, vc, and ends in j (without crossing any other edges). See also Figure 2b.

Similarly as in the case of D_1, to extend D_2 into a simple drawing with the edge uv, the edge needs to be routed through the interior of C either in the interior of cycle $ubvd$ or of $ucva$ as depicted in Figure 2b (with maybe also crossing the edge xy or the edge ij but staying in the same region). Furthermore, it cannot be routed through the interior of $udvc$ as it would need to intersect either an edge incident to u or v, or cross the edge ij twice.

In Figure 2c we can see a combination of an edge gadget and two vertex gadgets: it shows a copy D_2^e of the gadget D_2 (that we will say *corresponds* to an edge $e := wz$) over two different copies, D_1^w and D_1^z, of the gadget D_1 (that we will say *correspond* to vertices w and z, respectively). Notice that we add the label of the vertex or edge corresponding to the gadget (in this case either w or z or e) as a superindex. Since the region where both $v_w u_w$ and $v_z u_z$ can be drawn is forced, adding both prevents $v_e u_e$ from being added. Adding either only edge $v_w u_w$ or only edge $v_z u_z$ leaves exactly one possible region for edge $v_e u_e$.

We have all the main ingredients for our construction. Suppose that we are given a planar graph $G = (V, E)$ with maximum degree at most 3. This graph admits a 2-page book embedding $D(G)$ [5, 1]. In a 2-page book embedding all the vertices are placed on a (horizontal) line and the edges are arcs lying either in the upper half-plane on in the lower one and there are no proper crossings. The following lemma shows that replacing each vertex

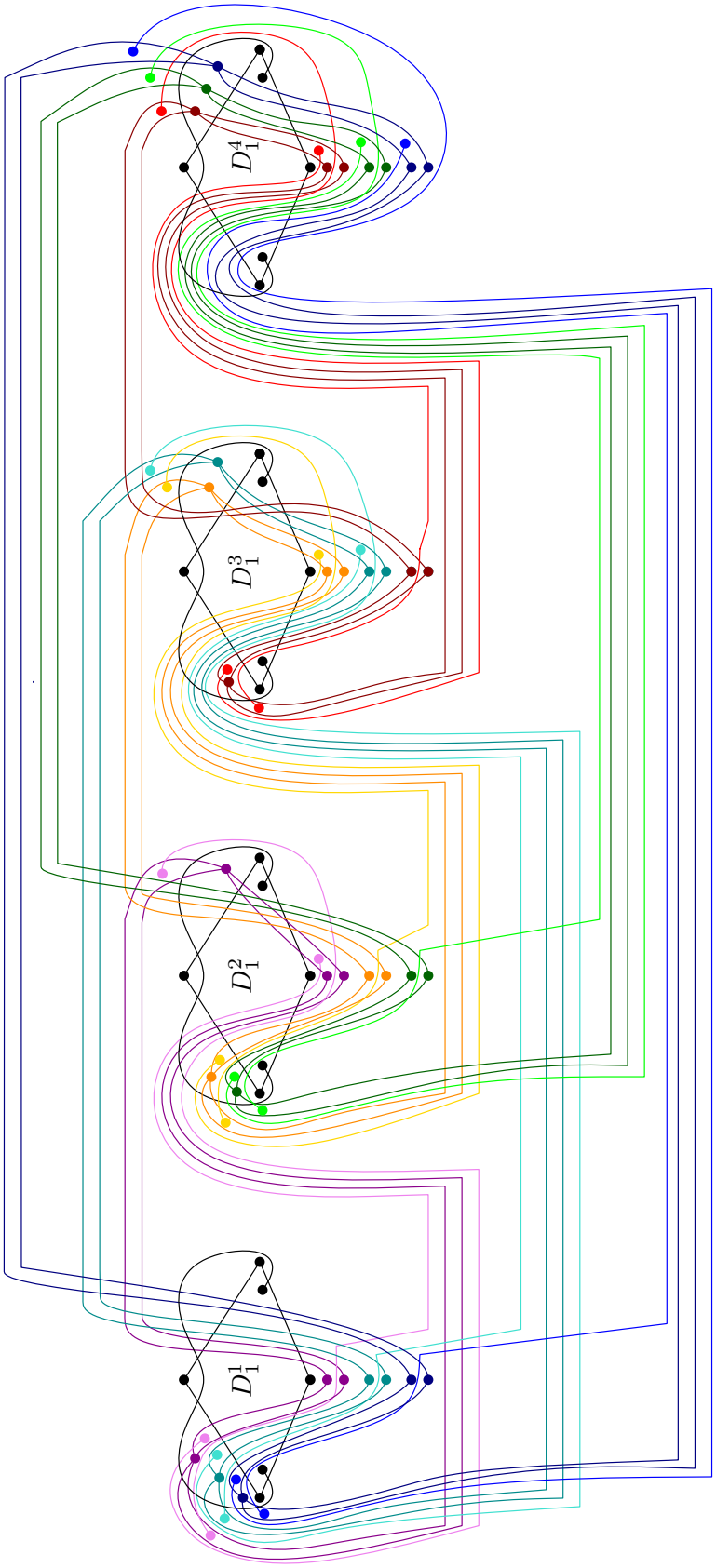


Figure 3 Drawing obtained by a reduction from K_4.

$w \in V$ in the drawing by a vertex gadget D_1^w and each edge $e \in E$ by an edge gadget D_2^e, we construct a simple drawing $D'(G')$.

▶ **Lemma 2.2.** *Given a 2-page book embedding $D(G)$ of a graph $G = (V, E)$, we can replace every vertex by a vertex gadget and every edge by an edge gadget to obtain a simple drawing.*

Proof. We will show that the copies $\{D_2^e : e \in E\}$ can be added to $\bigcup_{w \in V} D_1^w$ such that for every edge $e \in E$ incident to w and z ($w, z \in V$), $D_1^w \cup D_1^z \cup D_2^e$ is as in Figure 2c (up to interchanging the indices w and z), and the resulting drawing is a simple drawing.

First, for each vertex $w \in V$ we place the gadget D_1^w in its position, so all the copies of D_1 lie (equidistant) in a horizontal line. For the edges of G, since the drawing in Figure 2c is not symmetric, we choose an orientation. We orient all the edges in the 2-page book embedding $D(G)$ from left to right. We start adding the corresponding D_2 gadgets from left to right and from the shortest edges to the longest (where the length is the Euclidean distance between the endpoints). For an edge wz the intersections of the gadget D_2^{wz} (i) with the edges $u_w a_w$ and $u_w b_w$ are placed to the left of all the previous intersections of other edge gadgets with that edge; (ii) with the edge $v_w b_w$ are placed to the right of all the previous intersections with that edge; (iii) with the edge $v_w a_w$ are placed to the right of previous intersections with gadgets D_2^{wt} and to the left of previous intersections with gadgets D_2^{tw}; (iv) with the edges $u_z a_z$ and $u_z b_z$ are placed to the left of the previous intersections with gadgets D_2^{tz} (v) with the edge $v_z b_z$ are placed to the left of all previous intersections; and (vi) with the edge $v_z a_z$ are placed to the left of all previous intersections with gadgets D_2^{tz}. See Figure 3.

Moreover, the segments of some of the edges in the edge gadgets connecting from one vertex gadget to another vertex gadget can be drawn as strips in either the upper or lower half-plane with respect to the horizontal line. In those strips, segments of edges in the same strip don't cross and segments of edges in different strips cross at most once. See Figure 3.

Since neither of the gadgets of two incident edges cross, and edges between different gadgets are vertex-disjoint, we only have to worry about edges from different gadgets crossing more than once. By construction, no edge in an edge gadget intersects more than once with an edge in a vertex gadget. Thus, it remains to show that any two edges e_1 and e_2 from two distinct gadgets cross at most once. Such two edges are included in a subgraph H of G with exactly four vertices. The drawing induced by the four vertex gadgets and the at most six edge gadgets is homeomorphic to a subdrawing of the drawing in Figure 3. It is routine to check that this drawing a simple drawing, and thus e_1 and e_2 cross at most once. ◀

2.2 Reduction from Maximum Independent Set

For the decision version of the problem, given a planar graph $G = (V, E)$ with vertex degree at most 3 and a constant k, we reduce the problem of deciding if G has an independent set of size k to the problem of deciding if the simple drawing $D'(G')$ with a candidate set of edges M (where $M = \{u_w v_w : w \in V\} \cup \{u_e v_e : e \in E\}$) can be extended with a set of edges $M' \subseteq M$ with cardinality $|M'| = |E| + k$.

▶ **Lemma 2.3.** *The construction exhibited in the previous subsection is a polynomial-time reduction from independent set in planar maximum degree 3 graphs.*

Proof. To show the correctness of the (polynomial) reduction we first show that if G has an independent set I of size k then we can extend $D'(G')$ with a set M' of $|E| + k$ edges of M. Clearly, the k edges $\{u_w v_w : w \in I\}$ can be added to $D'(G')$ by the construction of the gadgets. Since I is an independent set, each edge has at most one endpoint in I. Thus, in

every edge gadget D_2^e at most one of the two possibilities for adding the edge $u_e v_e$ is blocked by the previous k added edges. We therefore can also add the $|E|$ edges $\{u_e v_e : e \in E\}$.

Conversely, assume that the set $M' \subset M$ of $|E| + k$ edges can be added to $D'(G')$. If the set of vertices $\{w : u_w v_w \in M'\}$ is an independent set of G, then we are done, since at most $|E|$ edges of the added ones can be from edge gadgets, so at least k are from vertex gadgets. Otherwise, there are two edges $u_w v_w$ and $u_z v_z$ in M' such that the corresponding vertices $w, z \in V$ are connected by the edge $wz \in E$. This implies that the edge $u_{wz} v_{wz}$ belongs to M but it cannot be in M'. By removing the edge $u_w v_w$ and adding the edge $u_{wz} v_{wz}$ to $D'(G')$ we obtain another valid extension with the same cardinality but one less edge belonging to a vertex gadget. Iteratively repeating this, we end with an extension N of $D'(G')$ that has cardinality $|E| + k$ and such that the set of vertices $\{w : u_w v_w \in N\}$ is an independent set of G of size at least k. ◀

3 Conclusions

In this paper we showed that, given a simple drawing $D(G)$ of a graph $G = (V, E)$ and a prescribed set M of edges of the complete graph with vertex set V, it is NP-hard to find the maximum number of edges from M that can be added to $D(G)$ such that the resulting drawing is simple. Focusing on the case $|M| = 1$, in the full version of this paper, on the one hand, we considered the problem in a dual setting and showed that this slight generalization is NP-complete and, on the other hand, we found sufficient conditions guaranteeing a polynomial-time decision. We hope that the work done in this direction paves the way to show the following:

▶ **Conjecture 1.** *Given a simple drawing $D(G)$ of a graph G and a pair u, v of non-adjacent edges, we can decide in polynomial time whether we can add uv to $D(G)$.*

Finally, modifying both the examples in Figure 1 and a previous example in [8, Figure 11] one can obtain arbitrarily large non-extensible drawings (where a given edge cannot be added) of graphs including complete bipartite graphs, complete graphs missing one edge, and matchings. Moreover, a modification of [8, Figure 1] shows that there are arbitrarily large examples that cannot be extended with an edge but such that the removal of any vertex or edge allows it to be extensible with any missing edge. So there is no hope of characterizing non-extensible drawings in terms of subdrawings. It is also not true that any graph with no isolated points has a non-extensible drawing, as any drawing of $K_{1,m}$ can be extended with any missing edge. This motivates the following problem:

▶ **Problem 1.** *Characterize all graphs that admit a non-extensible drawing.*

References

1 Michael A. Bekos, Martin Gronemann, and Chrysanthi N. Raftopoulou. Two-page book embeddings of 4-planar graphs. *Algorithmica*, 75(1):158–185, 2016.

2 Jean Cardinal and Stefan Felsner. Topological drawings of complete bipartite graphs. In *Proc. 24th Int. Symp. on Graph Drawing and Network Visualization (GD'16)*, pages 441–456, 2016.

3 Michael R. Garey and David S. Johnson. The rectilinear Steiner tree problem is NP-complete. *SIAM Journal on Applied Mathematics*, 32(4):826–834, 1977.

4 P. Hajnal, A. Igamberdiev, G. Rote, and A. Schulz. Saturated simple and 2-simple topological graphs with few edges. In Ernst W. Mayr, editor, *Graph-Theoretic Concepts in Computer Science*, pages 391–405, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

5 Lenwood Scott Heath. *Algorithms for embedding graphs in books*. PhD thesis, University of North Carolina, 1985.

6 Jan Kynčl. Simple realizability of complete abstract topological graphs simplified. In *Proc. 23rd Int. Symp. on Graph Drawing and Network Visualization (GD'15)*, pages 309–320, 2015.

7 Jan Kynčl. Improved enumeration of simple topological graphs. *Discrete & Computational Geometry*, 50(3):727–770, 2013.

8 Jan Kynčl, János Pach, Radoš Radoičić, and Géza Tóth. Saturated simple and k-simple topological graphs. *Computational Geometry*, 48(4):295–310, 2015.

9 János Pach, József Solymosi, and Géza Tóth. Unavoidable configurations in complete topological graphs. *Discrete & Computational Geometry*, 30(2):311–320, 2003.

Computing Optimal Tangles Faster

Oksana Firman[*1], Philipp Kindermann[1], Alexander Ravsky[†2], Alexander Wolff[‡1], and Johannes Zink[1]

1 **Institut für Informatik, Universität Würzburg**
`firstname.lastname@uni-wuerzburg.de`

2 **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 *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 numbers between 1 and n) 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. The aim is to find a tangle that realizes L using the smallest number of layers. We show that this problem is NP-hard, and we give an algorithm that computes an optimal tangle for n wires and a given list L of swaps in $O((2|L|/n^2+1)^{n^2/2}\varphi^n n)$ time, where $\varphi \approx 1.618$ is the golden ratio. We can treat lists where every swap occurs at most once in $O(n!\varphi^n)$ time. We implemented the algorithm for the general case and compared it to an existing algorithm.

1 Introduction

Our research is based on a recent paper of Olszewski et al. [4] who use *tangles* (which they call *templates*) to visualize 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. In the framework of Olszewski et al., one is given a set of wires that hang off a horizontal line in a fixed order, and a multiset of swaps between the wires; a tangle then is a visualization of these swaps, i.e., an order in which the swaps are performed, where only adjacent wires can be swapped and disjoint swaps can be done simultaneously. Olszewski et al. gave an algorithm for minimizing the height of a tangle. They didn't analyze the asymptotic running time of their algorithm (which we estimate below), but tested it on a set of benchmarks.

Wang [5] used the same optimization criterion for tangles, given only the final permutation. She showed that, in an optimal tangle, no swap occurs more than once. She used odd-even sort, a parallel variant of bubble sort, to compute tangles with at most one layer more than the minimum. Bereg et al. [1, 2] showed, given a final permutation, how to minimize the number of bends or *moves* (which are maximal "diagonal" segments of the wires).

Framework, Terminology, and Notation. We modify the terminology of Olszewski et al. [4] in order to introduce a formal algebraic framework for the problem. Given a natural number n of wires, a *(swap) list* $L = (l_{ij})$ of *order* n is a symmetric $n \times n$ matrix with non-negative

* Supported by DAAD.
† Supported by Erasmus+.
‡ Supported by DFG grant WO 758/9-1.

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. 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 swapped if n is even.)

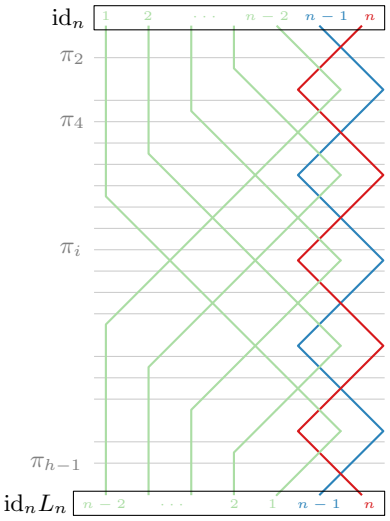


Figure 1 A list L_n for n wires (left) and the unique tangle of minimum height realizing L_n (right) for the start permutation $\mathrm{id}_n = 123\dots n$. Here, $n = 7$. The tangle is not simple because $\pi_2 = \pi_4$.

entries and zero diagonal. The *length* of L is defined as $|L| = \sum_{i<j} l_{ij}$. A list $L' = (l'_{ij})$ is a *sublist* of L if $l'_{ij} \le l_{ij}$ for each $i, j \in [n]$. A list is *simple* if all its entries are zeroes or ones.

A *permutation* is a bijection of the set $[n] = \{1, \dots, n\}$ onto itself. The set S_n of all permutations of the set $[n]$ is a group whose multiplication is a composition of maps (i.e., $(\pi\sigma)(i) = \pi(\sigma(i))$ for each pair of permutations $\pi, \sigma \in S_n$ and each $i \in [n]$). The identity of the group S_n is the identity permutation id_n. We write a permutation $\pi \in S_n$ as the sequence of numbers $\pi^{-1}(1)\pi^{-1}(2)\dots\pi^{-1}(n)$. For instance, the permutation π of $[4]$ with $\pi(1) = 3$, $\pi(2) = 4$, $\pi(3) = 2$, and $\pi(4) = 1$ is written as 4312. We denote the set of all permutations of order 2 in S_n by $S_{n,2}$, i.e., $\pi \in S_{n,2}$ if and only if $\pi\pi = \mathrm{id}_n$, e.g., $2143 \in S_{4,2}$.

For $i, j \in [n]$, the *swap* ij is the permutation that exchanges i and j, whereas the other elements of $[n]$ remain fixed. A set S of swaps is *disjoint* if each element of $[n]$ participates in at most one swap of S. Therefore, the product $\prod S$ of all elements of a disjoint set S of swaps does not depend on the order of factors and belongs to $S_{n,2}$. Conversely, for each permutation $\varepsilon \in S_{n,2}$ there exists a unique disjoint set $S(\varepsilon)$ of swaps such that $\varepsilon = \prod S(\varepsilon)$.

A permutation $\pi \in S_n$ *supports* a permutation $\varepsilon \in S_{n,2}$ if, for each swap $ij \in S(\varepsilon)$, i and j are neighbors in π. By induction, we can easily show that any permutation $\pi \in S_n$ supports exactly $F_{n+1} - 1$ permutations of order 2 where F_n is the n-th Fibonacci number.

Permutations π and σ are *adjacent* if there exists a permutation $\varepsilon \in S_{n,2}$ such that π supports ε and $\sigma = \pi\varepsilon$. In this case, $\sigma\varepsilon = \pi\varepsilon\varepsilon = \pi$ and σ supports ε, too. A *tangle* T of *height* h is a sequence $\langle \pi_1, \pi_2, \dots, \pi_h \rangle$ of permutations in which every two consecutive permutations are adjacent. A *subtangle* of T is a sequence $\langle \pi_k, \pi_{k+1}, \dots, \pi_l \rangle$ of consecutive permutations of T. Let $L(T) = (l_{ij})$ be the symmetric $n \times n$ matrix with zero diagonal where l_{ij} is the number of occurrences of swap ij in T. We say that T *realizes* $L(T)$; see Fig. 1. A list is π-*feasible* if it can be realized by a tangle starting from a permutation π. An id_n-feasible list is *feasible*; e.g., the list defined by the swaps 13 and 24 is *not* feasible.

A list $L = (l_{ij})$ also can be considered as a multiset of swaps, where l_{ij} is the multiplicity of swap ij. In particular, the notation $ij \in L$ means $l_{ij} > 0$. A tangle is *simple* if all its permutations are distinct. In particular, the height of a simple tangle is at most $n!$.

For each permutation $\pi \in S_n$ and a list $L = (l_{ij})$, we define a map $\pi L\colon [n] \to \mathbb{Z}$,

$$i \mapsto \pi(i) + |\{j\colon \pi(i) < \pi(j) \le n \text{ and } l_{ij} \text{ is odd}\}| - |\{j\colon 1 \le \pi(j) < \pi(i) \text{ and } l_{ij} \text{ is odd}\}|.$$

For each wire $i \in [n]$, $\pi L(i)$ is the position of the wire after all swaps in L have been applied to π. A list L is called π-*consistent* if $\pi L \in S_n$, or, more rigorously, if πL induces a permutation of $[n]$. An id_n-consistent list is *consistent*. For example, the list $\{12, 23, 13\}$ is consistent, whereas the list $\{13\}$ isn't. If L is not consistent, then it is clearly not feasible. For a list $L = (l_{ij})$, we define $1(L) = (l_{ij} \bmod 2)$. Since $\mathrm{id}_n L = \mathrm{id}_n 1(L)$, the list L is consistent if and only if $1(L)$ is consistent. We can compute $1(L)$ and check its consistency in $O(n + |1(L)|) = O(n^2)$ time. Hence, in the sequel we assume that all lists are consistent.

The *height* $h(L)$ of a feasible list L is the minimum height of a tangle that realizes L. A tangle T is *optimal* if $h(T) = h(L(T))$. In the Tangle-Height Minimization problem, we are given a swap list L and the goal is to compute an optimal tangle T realizing L. As initial wire order, we always assume the identity id_n.

Our Contribution. We show that Tangle-Height Minimization is NP-hard (see Section 2). We give an exact algorithm for simple lists running in $O(n!\varphi^n)$ time and an exact algorithm for general lists running in $O((2|L|/n^2+1)^{n^2/2}\varphi^n n)$ time, which is polynomial in $|L|$ for any fixed $n \ge 2$ (see Section 3). We implemented the algorithm for general lists and compared it to the algorithm of Olszewski et al. [4] using their benchmark set (see Section 4).

In order to be able to also compare the asymptotic runtime behaviors, we now analyze the algorithm of Olszewski et al. [4]. Their algorithm constructs a search tree whose height is bounded by the height $h(L)$ of an optimal tangle for the given list L. The tree has $1+d+d^2+\cdots+d^{h(L)-1} = (d^{h(L)}-1)/(d-1)$ vertices, where $d = F_{n+1}-1$ is a bound on the number of edges leaving a vertex, $F_n = (\varphi^n - (-\varphi)^{-n})/\sqrt{5} \in O(\varphi^n)$ is the n-th number in the Fibonacci sequence, and $\varphi = \frac{\sqrt{5}+1}{2} \approx 1.618$ is the golden ratio. Since it takes $O(n)$ time to deal with each vertex, the total running time is $O(\varphi^{(n+1)(h(L)-1)}5^{-(h(L)-1)/2}n)$. Since $2|L|/n \le h(L)-1 \le |L|$, this time is not better than $O(\varphi^{2|L|}5^{-|L|/n}n)$, which is exponential with respect to $|L|$ for fixed $n \ge 2$ and, hence, slower than our algorithm for the general case.

It is known (see, for instance, Wang [5]) that, for any simple list L, $h(L) \le n+1$. This implies that, on simple lists, the algorithm of Olszewski et al. runs in $O(\varphi^{(n+1)n}5^{-n}n) = e^{O(n^2)}$ time, whereas our algorithm for simple lists runs in $O(n!\varphi^n) = e^{O(n\log n)}$ time.

2 Complexity

▶ **Theorem 1.** *Given a list L of swaps and an integer $h > 0$, it is NP-hard to decide whether there is a tangle T of height $h(T) \le h$ realizing L.*

Proof sketch; for the full proof see [3]. From the given 3-Partition instance A, we construct in polynomial time a list L that can be realized by a tangle of height at most $mMB + O(m^2)$ if and only if A is a yes-instance. For an example instance, see Fig. 2.

In the list L we use two central wires ω and ω' swapping $2m$ times. Two consecutive swaps form a *loop*. We number the loops from top to bottom; depending on their index, loops are *even* or *odd*. We use wires $\beta_i, \beta_i', \gamma_i, \gamma_i', \delta_i, \delta_i'$ with $i \in [m]$ to enforce the following. For any tangle T realizing L, the height of the subtangle from the beginning of T to the end of the i-th odd loop is at least $(i-1)MB$ and the height of the subtangle from the $(i+1)$-th odd loop to the end of T is at least $(m-i)MB$, where $M \in \Theta(m^3)$ is some large scaling factor. For a tangle T to not exceed the maximum height, every even loop in T must have a

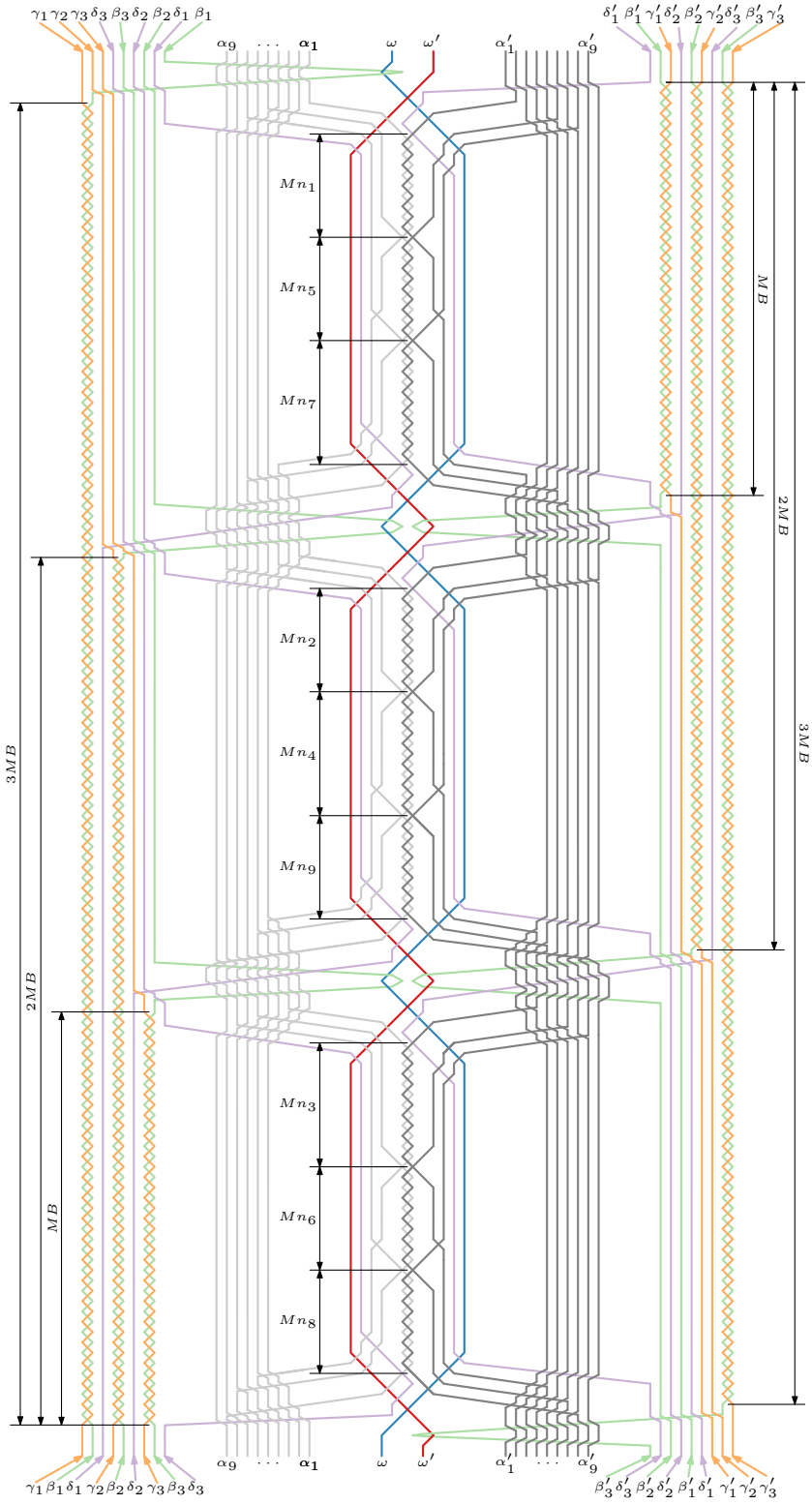


Figure 2 Example of our reduction from 3-Partition to Tangle-Height Minimization with $A_1 = \{n_1, n_5, n_7\}$, $A_2 = \{n_2, n_4, n_9\}$, $A_3 = \{n_3, n_6, n_8\}$, $m = 3$, $B = \sum_{i=1}^{3m} n_i/m$, and $M = 2m^3$.

height of about MB. We encode the numbers in A by introducing, for each $i \in [3m]$, two wires α_i and α'_i that swap Mn_i times. All α_i–α'_i swaps must occur inside exactly one of the even loops, but on different layers. The combination of these blocks of swaps inside the even loops corresponds to a partition of the given 3-Partition instance A. All tangles of height at most $mMB + O(m^2)$ correspond to a solution of A, and if there is a solution of A then there also is a tangle of height at most $mMB + O(m^2)$ realizing this solution. ◀

3 Exact Algorithms

The two algorithms that we describe in this section test whether a given list is feasible and, if yes, construct an optimal tangle realizing the list. For any permutation $\pi \in S_n$, we define the simple list $L(\pi) = (l_{ij})$ such that, for $0 \le i < j \le n$, $l_{ij} = 0$ if $\pi(i) < \pi(j)$, and $l_{ij} = 1$ otherwise. We use the following two lemmas, which we prove in the full version [3].

▶ **Lemma 2.** *For every permutation $\pi \in S_n$, $L(\pi)$ is the unique simple list with $\mathrm{id}_n L(\pi) = \pi$.*

▶ **Lemma 3.** *For every tangle $T = \langle \pi_1, \pi_2, \ldots, \pi_h \rangle$, we have $\pi_1 L(T) = \pi_h$.*

Simple lists. Let L be a consistent simple list. Wang's algorithm [5] creates a simple tangle from $\mathrm{id}_n L$, so L is feasible. Let $T = (\mathrm{id}_n = \pi_1, \pi_2, \ldots, \pi_h = \mathrm{id}_n L)$ be any tangle such that $L(T)$ is simple. Then, by Lemma 3, $\mathrm{id}_n L(T) = \pi_h$. By Lemma 2, $L(\pi_h)$ is the unique simple list with $\mathrm{id}_n L(\pi_h) = \pi_h = \mathrm{id}_n L$, so $L(T) = L(\pi_h) = L$ and thus T is a realization of L.

We compute an optimal tangle realizing $L = (l_{ij})$ as follows. Consider the graph G_L whose vertex set $V(G_L)$ consists of all permutations $\pi \in S_n$ with $L(\pi) \le L$ (componentwise). A directed edge (π, σ) between vertices $\pi, \sigma \in V(G_L)$ exists if and only if π and σ are adjacent as permutations and $L(\pi) \cap L(\pi^{-1}\sigma) = \varnothing$; the latter means that the set of (disjoint) swaps whose product transforms π to σ cannot contain swaps from the set whose product transforms id_n to π. The graph G_L has at most $n!$ vertices and maximum degree $F_{n+1} - 1 = O(\varphi^n)$, see Section 1. Furthermore, for each $h \ge 0$, there is a natural bijection between tangles of height $h+1$ realizing L and paths of length h in the graph G_L from the initial permutation id_n to $\mathrm{id}_n L$. A shortest such path can be found by BFS in $O(E(G_L)) = O(n!\varphi^n)$ time.

▶ **Theorem 4.** *For a simple list of order n,* Tangle-Height Minimization *can be solved in $O(n!\varphi^n)$ time.*

General lists. We can assume that $|L| \ge n/2$; otherwise, there is a wire $k \in [n]$ that doesn't belong to any swap. This wire splits L into smaller lists with independent realizations. (If there is a swap ij with $i < k < j$, then L is infeasible.)

Let $L = (l_{ij})$ be the given list. We compute an optimal tangle realizing L (if it exists) as follows. Let λ be the number of distinct sublists of L. We consider them in order of increasing length. Let L' be the next list to consider. We first check its consistency by computing the map $\mathrm{id}_n L'$. If L' is consistent, then we compute an optimal realization $T(L')$ of L' (if it exists), adding a permutation $\mathrm{id}_n L'$ to the end of a shortest tangle $T(L'') = \langle \pi_1, \ldots, \pi_h \rangle$ with π_h adjacent to $\mathrm{id}_n L'$ and $L'' + L(\langle \pi_h, \mathrm{id}_n L' \rangle) = L'$. This search also checks the feasibility of L' because such a tangle $T(L')$ exists if and only if the list L' is feasible. Since there are $F_{n+1} - 1$ permutations adjacent to $\mathrm{id}_n L'$, we have to check at most $F_{n+1} - 1$ lists L''. Hence, in total we spend $O(\lambda(F_{n+1} - 1)n)$ time for L. Assuming that $n \ge 2$, we bound λ as follows.

$$\lambda = \prod_{i<j}(l_{ij}+1) \le \left(\frac{\sum_{i<j}(l_{ij}+1)}{\binom{n}{2}}\right)^{\binom{n}{2}} = \left(\frac{|L|}{\binom{n}{2}} + 1\right)^{\binom{n}{2}} \le \left(\frac{2|L|}{n^2} + 1\right)^{n^2/2} \le e^{|L|}.$$

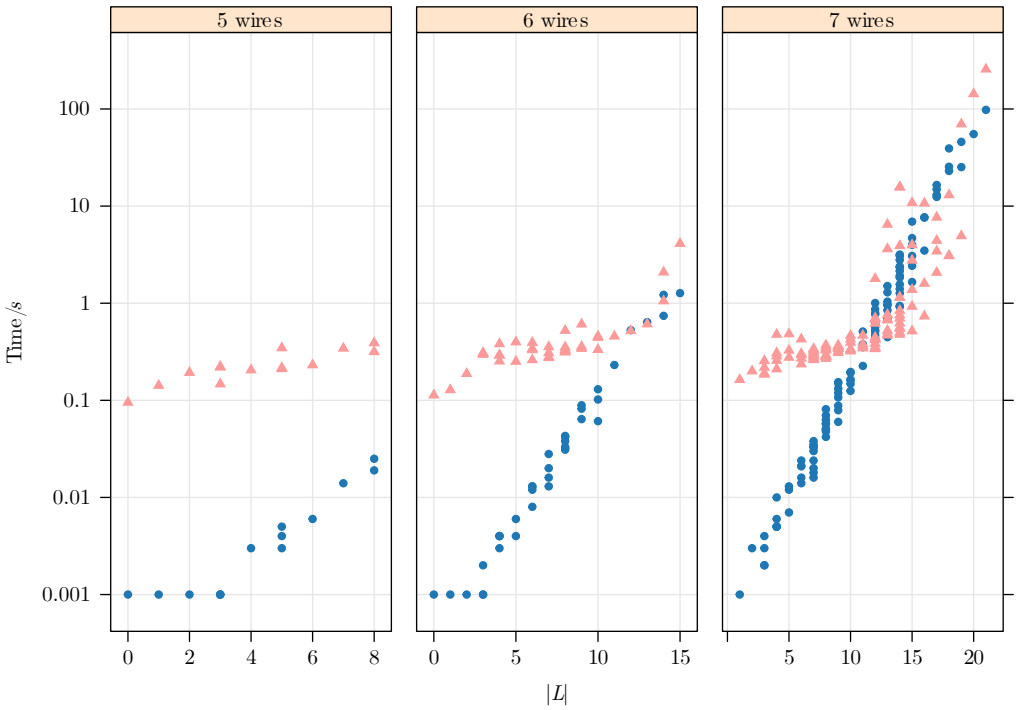


Figure 3 Comparison of our algorithm (blue circles) with the algorithm of Olszewski et al. (red triangles). The elapsed time is plotted on a log-scale.

We obtain the first inequality from the inequality between arithmetic and geometric means, the second one from Bernoulli's inequality, and the third one follows from $1 + x \leq e^x$.

▶ **Theorem 5.** *For a list L of order n,* Tangle-Height Minimization *can be solved in $O((2|L|/n^2 + 1)^{n^2/2}\varphi^n n)$ time.*

4 Experiments

We implemented the algorithm described in Theorem 5 and compared the running time of our implementation with the one of Olszewski et al. [4]. Their code and a database of all possible elementary linking matrices (most of them non-simple) of 5 wires (14 instances), 6 wires (38 instances), and 7 wires (115 instances) are available at `https://gitlab.uni.lu/PCOG`. Both their and our code is implemented in Python.

We ran our experiments on an Intel Core i7-4770K CPU with a clock speed of 3.50 GHz and 16 GB RAM under Windows 10 64bit. We measured the time to create an optimal tangle 5 times and took the arithmetic mean. The results are summarized in Fig. 3. For 8 of the instances with 7 wires, we stopped their algorithm after 2 hours without finding an optimal solution. We removed these instances from the analysis (although our algorithm found a solution all but of them in under four minutes, and the last one in 20 minutes).

On average, our algorithm was considerably faster; its running time was only 2.59% for 5 wires, 28.69% for 6 wires, and 72.85% for 7 wires of the running time of the algorithm by Olszewski et al. Our algorithm is also more space efficient; the memory usage peaked at 1.2 GB, while Olszewski et al. reportedly used up to 1 TB of memory in their experiments.

5 Open Problems

Is it NP-hard to test the feasibility of a given (non-simple) list? Even if feasibility turns out to be NP-hard, can we decide it faster than finding optimal tangles?

We call a list (l_{ij}) *non-separable* if, for any $i<k<j$, $l_{ik} = l_{kj} = 0$ implies $l_{ij} = 0$. Clearly, non-separability is necessary for a list to be feasible. For lists where all entries are even, we conjecture that this is also sufficient (which we have computer-verified for $n \leq 8$).

Acknowledgments. We thank Thomas C. van Dijk for stimulating discussions.

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: http://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 Oksana Firman, Philipp Kindermann, Alexander Ravsky, Alexander Wolff, and Johannes Zink. Computing optimal tangles faster. ArXiv report, 2019. URL: http://arxiv.org/abs/1901.06548.

4 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: http://arxiv.org/abs/1807.11853, doi:10.1007/978-3-030-04414-5_8.

5 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.