28th European Workshop on Computational Geometry

Booklet of Abstracts

EuroCG 2012

March 19-21, 2012

Assisi, Perugia, Italy

Preface

The 28th European Workshop on Computational Geometry (EuroCG 12) was held on March 19-21 in Assisi (Perugia), Italy. EuroCG is an annual, informal workshop whose goal is to provide a forum for scientists to meet, present their work, interact, and establish collaborations, in order to promote research in the field of Computational Geometry, within Europe and beyond.

We received 88 submissions, which underwent a limited refereeing process in order to ensure some minimal standards and to check for plausibility. We selected 73 submissions for presentation at the workshop, one of which was withdrawn later. Apart from abstracts of the 72 contributed talks, this booklet also contains abstracts of the three invited lectures, given by Olivier Devillers, Jan Kratochvíl, and Günter Rote. The content of this booklet, just like the pdf-files of abstracts available on the EuroCG website `http://eurocg.org`, must be regarded as a collection of preprints. Therefore, we expect most of the results presented here to be also submitted to peer-reviewed conferences and/or journals.

Many thanks to all authors and invited speakers for their participation, and to the members of the program committee and all external reviewers for their insightful comments. We also thank the organizing committee members: Carla Binucci, Emilio Di Giacomo, Luca Grilli, Fabrizio Montecchiani, and Salvatore A. Romeo. Finally, we are very grateful for the generous support of our sponsors: Camera di Commercio Perugia, EATCS Italian Chapter, Ordine degli Ingegneri Provincia di Perugia, Perugina, Vis4 s.r.l.

March 2012,

Walter Didimo and Giuseppe Liotta

Program Committee

Walter Didimo – University of Perugia

Sándor Fekete – Braunschweig University of Technology

Michael Hoffmann – ETH Zürich

Christian Knauer – Freie Universität Berlin

Marc van Kreveld – Utrecht University

Stefan Langerman – Universite Libre de Bruxelles

Sylvain Lazard – INRIA Nancy

Giuseppe Liotta – University of Perugia

Henk Meijer – Roosevelt Academy

Vera Sacristán – Universitat Politecnica de Catalunya

Additional Reviewers

Abellanas, Manuel

Aloupis, Greg

Binucci, Carla

Broutin, Nicolas

Cardinal, Jean

De Carufel, Jean-Lou

Devillers, Olivier

Di Giacomo, Emilio

Dujmović, Vida

Dupont, Laurent

Giannopoulos, Panos

Glisse, Marc

Goaoc, Xavier

Grilli, Luca

Haverkort, Herman

Hornus, Samuel

Korman, Matias

Kusters, Vincent

Lenhart, William

Levy, Bruno

Lieutier, Andre

Löffler, Maarten

Montecchiani, Fabrizio

Palop, Belen

Pfeifle, Julian

Ramos, Pedro

Saumell, Maria

Speckmann, Bettina

Staals, Frank

Stehn, Fabian

Tanigawa, Shin-Ichi

Taslakian, Perouz

Teillaud, Monique

Tiwary, Hans Raj

Werner, Daniel

Wood, David R.

Wuhrer, Stefanie

Table of Contents

Invited talks

Session 1.1 - A: Delaunay Graphs

Session 1.1 - B: Graph Drawing - I

Session 1.2 - A: Computational Models and Complexity Results

Session 2.3 - B: Proximity and Geometric Graphs

Session 3.1 - A: Voronoi Diagrams

Session 3.1 - B: Point Sets and Colors - II

Session 3.2 - A: Tiling, Folding, Piercing, and Gluing

Session 3.2 - B: Zonotopes, Convex Sets, and Mapping

Delaunay triangulations, theory vs practice.

Olivier Devillers

INRIA Sophia Antipolis - Méditerranée

Thirty years ago, at the early ages of computational geometry, the game of computational geometers was to design fancy algorithms with optimal theoretical complexities. The result was usually an algorithmic *journal article*, but not an *implementation*.

In the same period, some people were actually coding geometric algorithms, but without regard for the asymptotic complexities, and without proof of correctness of the result. They were not using the algorithms designed by theoreticians for two reasons [14]:
- these algorithms were too intricate and
- they rely on the arithmetic of real numbers, which differs from the floating-point arithmetic of computers.

These drawbacks of old computational geometry algorithms have been addressed in various ways to obtain algorithms that reconcile robustness, practical efficiency, and theoretical guarantees.

The aim of this talk is to illustrate some steps of this transition from theoretical stuff to efficient algorithms actually used in industrial applications. I will do this using my favorite problem: the construction of the Delaunay triangulation of a set of points and the dynamic maintenance of such a triangulation under point insertions and deletions.

Code used for the benchmarks is available at : `http://www.inria.fr/sophia/members/Olivier.Devillers/EuroCG2012/`.

CGAL benchmarks

To give an idea of the current performance of software computing Delaunay triangulation, we give the following timings obtained on a 16GByte, 2.3GHz workstation with CGAL 3.9 (compiled in release mode) using "Exact predicates inexact constructions kernel".

In the following table, "static insert" uses spatial sorting, "dynamic insert" uses the Delaunay hierarchy, and "deletion" removes all points in a random order.

Although theory claims that the deletion time does not depend on the triangulation size, we observe the contrary, especially in two dimensions. By profiling the code, we can observe that this phenomenon is due the load of the cache memory.

♯ points	static insert	dynamic insert	delete	maximal ♯ points before swap
	μs per point			
2D				
100K	0.90	2.8	1.7	
1M	0.92	5.8	2.5	
10M	1.06	9.0	3.0	
100M	1.15	13	3.2	
230M	1.2			
swap				230M
3D				
100K	7.8	18	102	
1M	8	25	106	
10M	8.2	33	109	
swap				60M

To evaluate the cost of a correct treatment of robustness issues, we compute the Delaunay triangulation of 10 millons points using two different kernels. The "Exact predicates inexact constructions kernel" correctly handles rounding errors and degenerate configurations while the `Simple_cartesian<double>` kernel runs a little bit faster but may fail to terminate, especially on degenerate or near degenerate input.

	2D	3D
Exact predicates inexact constructions kernel	10.6 s	82 s
`Simple_cartesian` <`double`> kernel	9.7 s	75 s

This is an extended abstract of a presentation given at EuroCG 2012. 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 in a conference with formal proceedings and/or in a journal.

Bibliographical notes

First algorithms for Delaunay triangulation appeared in the seventies.[1] The gift wrapping algorithm was proposed in 1970 to compute 2D Delaunay triangulation by Frederick, Wong, and Edge [31] and by Chand and Kapur for convex hull in 3D [12]. Incremental algorithms were introduced by Lawson in 2D in 1977 [37] and in 3D by Bowyer and Watson in 1981 [10, 44]. A gift wrapping 3D triangulation algorithm was proposed by Nguyen [40].

Regarding **worst-case optimal** algorithms, a divide-and-conquer approach was developed by Lee in 2D in 1978 [38] and Fortune proposed his plane sweep algorithm in 1986 [30]. For higher dimensions, the final solution appeared in 1993 with Chazelle's optimal algorithm for convex hull [13].

On the way to make algorithms easier to code, **randomization** was a very useful ingredient. Randomization was introduced in computational geometry by Clakson and Shor paper in 1989 [16]; from then, randomized incremental constructions (RIC) were widely used in the domain. In fact a RIC is actually a deterministic incremental algorithm but analyzed under the hypothesis of a random order for data insertion. Previously, Boissonnat and Teillaud had introduced an algorithm in this spirit for Delaunay triangulation: the Delaunay tree [9] but its correct analysis [8] was actually deduced few years later, by an application of Clarkson and Shor's techniques. A variant of the Delaunay tree was also proposed by Guibas, Knuth, and Sharir [35]. In 1992, Boissonnat, Devillers, Schott, Teillaud and Yvinec proposed the history graph (or influence graph) to allow insertions and deletions within RIC [7]. In 2002, Devillers used the Delaunay hierarchy [18] to reconcile a proven randomized complexity with good constants for both time and space.

One of the obstacles to the use of computational geometry algorithms by practitioners was their lack of **robustness** when using floating point arithmetic, since their proofs of correctness rely on the arithmetic of real numbers. One way to cope with the problem, as proposed by Sugihara and Iri [43], is to perform additional *combinatorial and topological* tests to enforce this correctness; this approach is used in Held's software VRONI [36]. With the exact computation paradigm [45] Yap proposed another solution that allows the use of algorithms developed with real arithmetic in mind: the basic geometric tests, called *predicates*, are carefully isolated in the algorithm and their evaluation is done exactly; efficient predicates for Delaunay triangulations were proposed by Shewchuk [42] and Devillers and Pion [22]. The use of the exact computation paradigm can be used only in the sensitive call to the predicate: this is the idea of structural filtering [32].

Close to robustness issues are the treatment of **degeneracies**. Symbolic perturbations are the generic answer to that problem [27, 28, 41] and exist in several variations in the literature, including special versions for Delaunay triangulations [2, 24].

Beyond the global algorithmic choices such as RIC, working on details of the algorithm design have an influence on the **constant factors** of the complexity. Many point location algorithms are implemented by walking in the triangulation, which can be done in several ways [23, 17]. Depending on the size of the point set, the location strategy can be adapted from a brute force search for very small point sets, to a walk, or jump & walk [39, 25], or Delaunay hierarchy [18] when the size increases. If the points are known in advance, a true random order have some disadvantage and *spatial sort* can be used for preprocessing [4, 11].

Deleting one point of degree d from a 2D Delaunay triangulation can be done in $O(d)$ time, by using a quite complicated algorithm originally designed for the Delaunay triangulation of a convex polygon by Aggarwal, Guibas, Saxe, and Shor [1]. A much simpler randomized $O(d)$ solution has been proposed by Chew [15]. In practice, various non-optimal solutions of complexity $O(d \log d)$ or $O(d^2)$ are often preferred since d is usually a small number [19]. A specialized implementation for small value of d, optimizing the number of incircle tests and the memory management can yield substantial improvement [20]. In 3D or higher dimensions, a way of managing point deletion is to compute from scratch the (small) Delaunay triangulation of the neighbors of the removed point, and to plug the relevant simplices inside the hole created in the whole triangulation by the removal of the simplices incident to the deleted point.

In dimension 3 or higher, the **size of the Delaunay** triangulation is not constrained by the number of points as it is in 2D. There are several results about that size under various hypotheses. For random points in a ball in any fixed dimension, Dwyer proved that the expected size is $\Theta(n)$ [26]. For random points in 3D on the boundary of a polyhedron, the expected size was proved to be $\Theta(n)$ [34] in the convex case and between $\Omega(n)$ and $O(n \log n)$ [33] in the non convex case by Golin and Na. If the probabilistic hypothesis is replaced by some hypotheses controlling the point distribution on the boundary of the polyhedron (called (ϵ, κ)-sampling) then the size of the triangulation has been proved to be $\Theta(n)$ [5] by

[1] thanks to Jonathan Shewchuk for his summary on this topic in the compgeom mailing list.

Attali and Boissonnat. The same authors with Lieutier proved that, applying this sampling hypotheses to a generic smooth surface, yields an $O(n \log n)$ complexity [6]. If the smooth surface is non generic (e.g. a cylinder) and sampled with the same hypotheses, the complexity can be $\Omega(n\sqrt{n})$ [29] as proved by a construction of Erickson. Erickson, Devillers, and Goaoc proved that the triangulation of randomly distributed points on a cylinder has complexity $\Theta(n \log n)$ [21]. In higher dimensions, Amenta, Attali, and Devillers proved that a good-sampling of a p-dimensional polyhedron embedded in dimension d has $O(n^k)$ size with $k = \frac{d+1-\left\lceil \frac{d+1}{p+1} \right\rceil}{p}$ [3].

References

[1] A. Aggarwal, L. J. Guibas, J. Saxe, and P. W. Shor. A linear-time algorithm for computing the Voronoi diagram of a convex polygon. *Discrete Comput. Geom.*, 4(6):591–604, 1989. http://www.springerlink.com/content/7722150064q64865/.

[2] Pierre Alliez, Olivier Devillers, and Jack Snoeyink. Removing degeneracies by perturbing the problem or the world. *Reliable Computing*, 6:61–79, 2000. Special Issue on Computational Geometry, http://hal.inria.fr/inria-00338566.

[3] Nina Amenta, Dominique Attali, and Olivier Devillers. A tight bound for the Delaunay triangulation of points on a polyhedron. Research Report 6522, INRIA, 2008. http://hal.inria.fr/inria-00277899.

[4] Nina Amenta, Sunghee Choi, and Günter Rote. Incremental constructions con BRIO. In *Proc. 19th Annu. Sympos. Comput. Geom.*, pages 211–219, 2003. http://page.inf.fu-berlin.de/~rote/Papers/pdf/Incremental+constructions+con+BRIO.pdf.

[5] 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. http://www.springerlink.com/content/udhapea67abmq086/.

[6] 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 237–246, 2003. http://dl.acm.org/citation.cfm?id=777823.

[7] J.-D. Boissonnat, O. Devillers, R. Schott, M. Teillaud, and M. Yvinec. Applications of random sampling to on-line algorithms in computational geometry. *Discrete and Computational Geometry*, 8:51–71, 1992. http://hal.inria.fr/inria-00090675.

[8] J.-D. Boissonnat and M. Teillaud. On the randomized construction of the Delaunay tree. *Theoret. Comput. Sci.*, 112:339–354, 1993. http://www.sciencedirect.com/science/article/pii/030439759390024N.

[9] Jean-Daniel Boissonnat and Monique Teillaud. A hierarchical representation of objects: The Delaunay tree. In *Proc. 2nd Annu. Sympos. Comput. Geom.*, pages 260–268, 1986. http://dl.acm.org/citation.cfm?id=10543.

[10] A. Bowyer. Computing Dirichlet tesselations. *Comput. J.*, 24:162–166, 1981. http://comjnl.oxfordjournals.org/content/24/2/162.short.

[11] Kevin Buchin. Constructing Delaunay triangulations along space-filling curves. In *Proc. 17th European Symposium on Algorithms*, volume 5757 of *Lecture Notes Comput. Sci.*, pages 119–130. Springer-Verlag, 2009. http://www.springerlink.com/index/m17216w072m54438.pdf.

[12] D. R. Chand and S. S. Kapur. An algorithm for convex polytopes. *J. ACM*, 17(1):78–86, January 1970. http://dl.acm.org/citation.cfm?id=321564.

[13] Bernard Chazelle. An optimal convex hull algorithm in any fixed dimension. *Discrete Comput. Geom.*, 10:377–409, 1993. http://www.springerlink.com/content/f62210744m187220/.

[14] Bernard Chazelle et al. Application challenges to computational geometry: CG impact task force report. In B. Chazelle, J. E. Goodman, and R. Pollack, editors, *Advances in Discrete and Computational Geometry*, volume 223 of *Contemporary Mathematics*, pages 407–463. American Mathematical Society, Providence, 1999.

[15] L. P. Chew. Building Voronoi diagrams for convex polygons in linear expected time. Technical Report PCS-TR90-147, Dept. Math. Comput. Sci., Dartmouth College, Hanover, NH, 1990. http://www.cs.dartmouth.edu/reports/TR90-147.pdf.

[16] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete Comput. Geom.*, 4:387–421, 1989. http://www.springerlink.com/content/b9n24vr730825p71/.

[17] Pedro Machado Manhães de Castro and Olivier Devillers. Simple and efficient distribution-sensitive point location, in triangulations. In *Workshop on Algorithm Engineering and Experiments*, pages 127–138, 2011. http://www.siam.org/proceedings/alenex/2011/alenex11.php.

[18] Olivier Devillers. The Delaunay hierarchy. *Internat. J. Found. Comput. Sci.*, 13:163–180, 2002. http://hal.inria.fr/inria-00166711.

[19] Olivier Devillers. On deletion in Delaunay triangulation. *Internat. J. Comput. Geom. Appl.*, 12:193–205, 2002. http://hal.inria.fr/inria-00167201.

[20] Olivier Devillers. Vertex removal in two dimensional Delaunay triangulation: Speed-up by low degrees optimization. *Computational Geometry: Theory and Applications*, 44:169–177, 2011. http://hal.inria.fr/inria-00560379/.

[21] Olivier Devillers, Jeff Erickson, and Xavier Goaoc. Empty-ellipse graphs. In *Proc. 19th ACM-SIAM Sympos. Discrete Algorithms*, pages 1249–1256, 2008. http://hal.inria.fr/inria-00176204.

[22] Olivier Devillers and Sylvain Pion. Efficient exact geometric predicates for Delaunay triangulations. In *Proc. 5th Workshop Algorithm Eng. Exper.*, pages 37–44, 2003. http://hal.inria.fr/inria-00344517/.

[23] Olivier Devillers, Sylvain Pion, and Monique Teillaud. Walking in a triangulation. *Internat. J. Found. Comput. Sci.*, 13:181–199, 2002. http://hal.inria.fr/inria-00102194/.

[24] Olivier Devillers and Monique Teillaud. Perturbations for Delaunay and weighted Delaunay 3d triangulations. *Computational Geometry: Theory and Applications*, 44:160–168, 2011. http://hal.archives-ouvertes.fr/inria-00560388/.

[25] Luc Devroye, Ernst Peter Mücke, and Binhai Zhu. A note on point location in Delaunay triangulations of random points. *Algorithmica*, 22:477–482, 1998. http://cg.scs.carleton.ca/~luc/devroye_mucke_zhu_1998_a_note_on_point_location_in_delaunay_triangulations_of_random_points.pdf.

[26] R. Dwyer. The expected number of k-faces of a Voronoi diagram. *Internat. J. Comput. Math.*, 26(5):13–21, 1993. http://www.sciencedirect.com/science/article/pii/0898122193900687.

[27] H. Edelsbrunner and E. P. Mücke. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. *ACM Trans. Graph.*, 9(1):66–104, 1990. http://dl.acm.org/citation.cfm?id=77639.

[28] I. Emiris and J. Canny. A general approach to removing degeneracies. *SIAM J. Comput.*, 24:650–664, 1995. http://epubs.siam.org/sicomp/resource/1/smjcat/v24/i3/p650_s1.

[29] Jeff Erickson. Dense point sets have sparse Delaunay triangulations or "... but not too nasty". *Discrete & Computational Geometry*, 33:83–115, 2005. http://www.springerlink.com/content/91ukk7qdwvpx77d1/.

[30] S. J. Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica*, 2:153–174, 1987. http://www.springerlink.com/content/n88186t1165168rw/.

[31] C. O. Frederick, Y. C. Wong, and F. W. Edge. Two-dimensional automatic mesh generation for structural analysis. *Internat. J. Numer. Methods Eng.*, 2:133–144, 1970. http://onlinelibrary.wiley.com/doi/10.1002/nme.1620020112/abstract.

[32] Stefan Funke, Kurt Mehlhorn, and Stefan Näher. Structural filtering: A paradigm for efficient and exact geometric programs. *Comput. Geom. Theory and Appl.*, pages 179–194, 2005. http://www.sciencedirect.com/science/article/pii/S0925772104001348.

[33] 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. http://www.cse.ust.hk/~golin/pubs/SCG_02.pdf.

[34] 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. http://www.cse.ust.hk/~golin/pubs/3D_Voronoi_I.pdf.

[35] Leonidas J. Guibas, D. E. Knuth, and Micha Sharir. Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica*, 7:381–413, 1992. http://www.springerlink.com/content/p8377h68j8216860/.

[36] M. Held. Vroni: An engineering approach to the reliable and efficient computation of Voronoi diagrams of points and line segments. *Comput. Geom. Theory Appl.*, 18:95–123, 2001. http://www.sciencedirect.com/science/article/pii/S0925772101000037.

[37] C. L. Lawson. Software for C^1 surface interpolation. In J. R. Rice, editor, *Math. Software III*, pages 161–194. Academic Press, New York, NY, 1977. http://www.mendeley.com/research/software-for-c1-interpolation/.

[38] D. T. Lee. *Proximity and reachability in the plane*. PhD thesis, Coordinated Science Lab., Univ. Illinois, Urbana, Ill., 1978. http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA069764.

[39] Ernst P. Mücke, Isaac Saias, and Binhai Zhu. Fast randomized point location without preprocessing in two- and three-dimensional Delaunay triangulations. In *Proc. 12th Annu. Sympos. Comput. Geom.*, pages 274–283, 1996. http://dl.acm.org/citation.cfm?id=237396.

[40] V. Ph. Nguyen. Automatic mesh generation with tetrahedron elements. *Internat. J. Numer. Methods Eng.*, 18:273–289, 1982.

[41] R. Seidel. The nature and meaning of perturbations in geometric computing. *Discrete Comput. Geom.*, 19:1–17, 1998. http://www.springerlink.com/content/px52xh005cxxvdku/.

[42] Jonathan Richard Shewchuk. Adaptive precision floating-point arithmetic and fast robust geometric predicates. *Discrete Comput. Geom.*, 18(3):305–363, 1997. http://www.springerlink.com/content/gfepd5qjykp9mk1f/.

[43] K. Sugihara and M. Iri. Construction of the Voronoi diagram for 'one million' generators in single-precision arithmetic. *Proc. IEEE*, 80(9):1471–1484, September 1992. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=163412&tag=1.

[44] D. F. Watson. Computing the n-dimensional Delaunay tesselation with applications to Voronoi polytopes. *Comput. J.*, 24(2):167–172, 1981. http://comjnl.oxfordjournals.org/content/24/2/167.short.

[45] C. K. Yap and T. Dubé. The exact computation paradigm. In D.-Z. Du and F. K. Hwang, editors, *Computing in Euclidean Geometry*, volume 4 of *Lecture Notes Series on Computing*, pages 452–492. World Scientific, Singapore, 2nd edition, 1995. http://cs.nyu.edu/cs/faculty/yap/papers/paradigm.ps.

Geometric Intersection Graphs: Old Problems, New Approaches (and Vice Versa)

Jan Kratochvíl
Charles University Prague

Abstract

Intersection graphs of geometrical objects have been in the focus of interest of graph theorists and computer scientists for quite a while. Many of such classes of graphs arise from real world applications, they allow elegant characterizations, as well as fast algorithms for optimization problems NP-hard on general graphs. At the same time this area is still a rich source of interesting and challenging open problems, though many of such problems have been recently solved.

We will survey the progress and recent results, comment on the few remaining old open problems and present new approaches, ideas and problems that have emerged in the last years.

The topics covered will include sizes of representations, a concept closely related to the NP-membership versus PSPACE-hardness question for some of the classes under consideration, the newly developing approach of extending partial representations, and more.

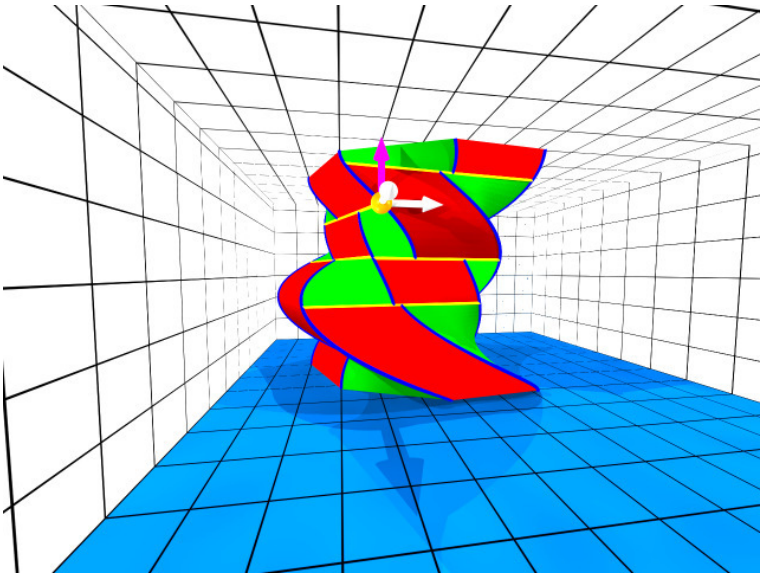
Motion Planning for a Rigid Robot in the Plane

Günter Rote
Freie Universität Berlin

Abstract

We consider the motion of a rigid polygonal robot among polygonal obstacles. After introducing the standard configuration space approach, we will see that the contact surfaces in configuration space are patches of helicoids and cylinders. We then consider various precise and approximate approaches for approaching the problem. This includes techniques based on precise algebraic computation with rational functions of a single variable, which are combined with various sampling-based approaches and approaches that approximate the contact surfaces geometrically. These approximations provide error guarantees.

This talk is based on work by and with Oren Salzman, Michael Hemmer, Barak Raveh, Dan Halperin (Tel Aviv University), and Dror Atariah and Sunayana Ghosh (Freie Universität Berlin) within the EU-funded project Computational Geometric Learning (CGL) (2010-2013).



Covering spaces and Delaunay triangulations of the 2D flat torus

Mikhail Bogdanov* Monique Teillaud† Gert Vegter‡

Abstract

A previous algorithm was computing the Delaunay triangulation of the flat torus, by using a 9-sheeted covering space [5]. We propose a modification of the algorithm using only a 8-sheeted covering space, which allows to work with 8 periodic copies of the input points instead of 9. The main interest of our contribution is not only this result, but most of all the method itself: this new construction of covering spaces generalizes to Delaunay triangulations of surfaces of higher genus.

1 Introduction

The Delaunay triangulation is a widely used structure in Computational Geometry. Delaunay triangulations of the *flat torus* [13] are used in different domains of science, like astronomy [10, 11, 14]. More references can be found in [5, 6, 3].

The Delaunay triangulation of points on a 2D flat torus was previously obtained from a triangulation of the convex hull of nine periodic copies of the points in the plane $\mathbb{E}^2$, laid in a 3x3 pattern [9, 8]. A more recent algorithm directly computes the Delaunay triangulation in a flat torus [5, 4], thus providing all adjacency relations between triangles. The algorithm is incremental, it resorts to a 9-sheeted covering space of the torus, and switches to computing in the initial torus as soon as possible.

After recalling some background on orbit spaces in Section 2, we propose in Section 3 a construction of a family of 2^k sheeted-covering spaces of the flat torus, $k > 0$. In Section 4 we show that the Delaunay triangulation of any finite set of $n > 1$ points in the 8-sheeted covering space of this family is well defined, we give conditions ensuring that the Delaunay triangulation can be defined in the 2- and 4-sheeted covering spaces of the family, and we propose a modified incremental algorithm computing with covering spaces of no more than eight sheets. Section 5 finally gives hints about the extension of the construction to surfaces of higher genus.

*INRIA Sophia Antipolis - Méditerranée, France, Mikhail.Bogdanov@inria.fr

†INRIA Sophia Antipolis - Méditerranée, France, Monique.Teillaud@inria.fr

‡Department of Mathematics and Computing Science, University of Groningen, The Netherlands, G.Vegter@rug.nl

2 Background

The flat torus. The unit sphere $\mathcal{S}^1$ can be mapped into $\mathbb{E}^2$ by the map $c : \mathbb{E} \to \mathbb{E}^2$, defined by $c(s) = (\cos s, \sin s)$. Similarly, the torus $\mathbb{T}^2$, being homeomorphic to the Cartesian product $\mathcal{S}^1 \times \mathcal{S}^1$, can be mapped into $\mathbb{E}^4$ by the map $(s,t) \mapsto (\cos s, \sin s, \cos t, \sin t)$. The corestriction of this map to its image yields a map $\pi : \mathbb{E}^2 \to \mathbb{T}^2$. Since this torus, endowed with the ambient metric of $\mathbb{E}^4$, has zero Gaussian curvature, it is called the *flat torus* (as opposed to the 'standard' torus of revolution in $\mathbb{E}^3$, which has regions of positive and negative Gaussian curvature). The periodicity of the maps c and π corresponds to an action of the groups $\mathbb{Z}$ and $\mathbb{Z}^2$, respectively. Such group actions are crucial in our approach, so we first provide the necessary background.

Covering Spaces. In this paper, all topological spaces are assumed to be connected. A *covering map* is a continuous surjective map $\rho : \mathbb{Y} \to \mathbb{X}$ from a topological space $\mathbb{Y}$ to a topological space $\mathbb{X}$, such that each point $x \in \mathbb{X}$ is evenly covered, i.e., there is a an open neighborhood V of x such that $\rho^{-1}(V)$ is the disjoint union of a family $\{U_\alpha\}$ of open subsets of $\mathbb{Y}$ such that $\rho|_{U_\alpha}$ is a homeomorphism for each α. One of our key examples is the map $\pi : \mathbb{E}^2 \to \mathbb{T}^2$ introduced in the previous paragraph. If a point in $\mathbb{X}$ has a finite number k of pre-images under the covering map ρ, then the connectedness of $\mathbb{X}$ implies that each point of $\mathbb{X}$ has k pre-images. In this case $\mathbb{Y}$ is called a k-sheeted cover of $\mathbb{X}$. Taking $\mathbb{X} = \mathbb{T}^2$ and $\mathbb{Y} = \mathbb{T}^2$, the map $\sigma : \mathbb{Y} \to \mathbb{X}$, mapping the point $(\cos s, \sin s, \cos t, \sin t) \in \mathbb{Y}$ to the point $(\cos 3s, \sin 3s, \cos 3t, \sin 3t) \in \mathbb{X}$, is an example of a nine-sheeted covering map of the torus (by another torus). If, moreover, $\mathbb{Y}$ is simply connected, i.e., if every closed curve in $\mathbb{Y}$ can be contracted to a point, then $\mathbb{Y}$ is called the *universal cover* of $\mathbb{X}$. It covers all (connected) covers of $\mathbb{X}$ in the sense that for every covering map $\sigma : \mathbb{M} \to \mathbb{X}$ there is a map $f : \mathbb{Y} \to \mathbb{M}$ such that $\pi = \sigma \circ f$. In our earlier example of the nine-sheeted covering map $\sigma : \mathbb{T}^2 \to \mathbb{T}^2$ the map $f : \mathbb{E}^2 \to \mathbb{T}^2$ is given by $f(s,t) = (\cos \frac{1}{3}s, \sin \frac{1}{3}s, \cos \frac{1}{3}t, \sin \frac{1}{3}t)$.

Group Actions and Orbit Spaces. If u and v are orthogonal vectors in $\mathbb{E}^2$ of equal length, then the map $\mathbb{Z}^2 \times \mathbb{E}^2 \to \mathbb{E}^2$, given by $(m,n) * p \mapsto p + mu + nv$, defines an action of the group $\mathbb{Z}^2$ on $\mathbb{E}^2$. In general, an *action* of a group $\mathcal{G}$ on a topological space $\mathbb{X}$ is a map $\mathcal{G} \times \mathbb{X} \to \mathbb{X} : (g,x) \mapsto g * x$ such that

This is an extended abstract of a presentation given at EuroCG 2012. 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 in a conference with formal proceedings and/or in a journal.

(i) for $g \in \mathcal{G}$, the map $\mathbb{X} \to \mathbb{X} : x \mapsto g * x$, is a homeomorphism on $\mathbb{X}$;
(ii) the identity element $e \in \mathcal{G}$ corresponds to the identity on $\mathbb{X}$, i.e., $e * x = x$, for $x \in \mathbb{X}$;
(iii) $(hg) * x = h * (g * x)$, for $g, h \in \mathcal{G}$ and $x \in \mathbb{X}$.
Each element $g \in \mathcal{G}$ correponds to a homeomorphism $x \mapsto g * x$, and this correspondence is an isomorphism according to (ii) and (iii). In our case, $(m, n) \in \mathbb{Z}^2$ corresponds to the translation $a_u^m b_v^n$, where a_u and b_v are the translations over u and v, respectively.
The *orbit* of a point $x \in \mathbb{X}$ under the action of a group $\mathcal{G}$ is the set $\mathcal{G}x = \{g * x \mid g \in \mathcal{G}\}$. Properties (ii) and (iii) imply that the orbits form a partition of $\mathbb{X}$. In other words, the group action induces an equivalence relation on $\mathbb{X}$, given by $x \sim y$ if $x = g * y$ for some $g \in \mathcal{G}$. The *orbit space* $\mathbb{X}/\mathcal{G}$ is the set of all orbits of $\mathbb{X}$ under the action of $\mathcal{G}$. A *fundamental region* is a subset of $\mathbb{X}$ which contains at least one point of each $\mathcal{G}$-orbit with at most one point of each orbit in its interior.
The orbit space of the $\mathbb{Z}^2$-action introduced at the beginning of this paragraph is the torus. Each orbit forms a lattice in $\mathbb{E}^2$. The minimal distance by which $\mathcal{G} = \mathbb{Z}^2$ moves a point of $\mathbb{E}^2$ is $\delta(\mathcal{G}) = \| u \| = \| v \|$, where $\| \cdot \|$ denotes the Euclidean norm. Any closed square of edge length $\delta(\mathcal{G})$ whose edges are parallel to u and v is a fundamental domain of $\mathcal{G}$. For a point $p \in \mathbb{E}^2$, the half-open square $\mathcal{D}(p) = \left\{p + t_u u + t_v v \mid t_u, t_v \in [-\frac{1}{2}, \frac{1}{2})\right\}$ contains exactly one representative of each element of $\mathbb{T}^2$. We call $\mathcal{D}(O)$ the *original domain*, where O denotes the origin.
Group Actions and Covering Spaces. In all our examples, the group actions have the property that each $x \in \mathbb{X}$ has a neighborhood in which each point belongs to a different $\mathcal{G}$-orbit. (Technically, this follows from the fact that the action is *discontinuous* and *fixed point free*; See [12] for details.) If, moreover, $\mathbb{X}$ is a *metric space* with metric d, as in our examples, then the orbit space $\mathbb{M} := \mathbb{X}/\mathcal{G}$ can also be endowed with a metric $d_{\mathbb{M}}$ given by $d_{\mathbb{M}}(\mathcal{G}x, \mathcal{G}y) = \min\{d(x', y') \mid x' \in \mathcal{G}x, y' \in \mathcal{G}y\}$. The orbit space is *locally isometric* to the space $\mathbb{X}$. In this case, the quotient map $\varrho : \mathbb{X} \to \mathbb{X}/\mathcal{G}$ is a covering map, which is a local isometry. In particular, if $\mathbb{X} = \mathbb{E}^2$, as in our examples, orbit spaces are *flat*.
A normal subgroup $\mathcal{H}$ of $\mathcal{G}$ has index k if it has k distinct right cosets in $\mathcal{G}$, i.e., there are k elements $e = g_1, g_2, \ldots, g_k$ of $\mathcal{G}$ such that $\mathcal{G}$ is the disjoint union of $g_1\mathcal{H}, g_2\mathcal{H}, \ldots, g_k\mathcal{H}$. The $\mathcal{H}$-action on $\mathbb{X}$ induced by the $\mathcal{G}$-action gives rise to an orbit space $\mathbb{X}/\mathcal{H}$. This space is a k-sheeted cover of the orbit space $\mathbb{X}/\mathcal{G}$, since the natural map $\rho : \mathbb{X}/\mathcal{H} \to \mathbb{X}/\mathcal{G} : \mathcal{H}x \mapsto \mathcal{G}x$ is k-to-1. Indeed, each $\mathcal{G}$-orbit is partitioned into k orbits of $\mathcal{H}$: $\mathcal{G}x = g_1\mathcal{H}x \cup g_2\mathcal{H}x \cup \cdots \cup g_k\mathcal{H}x$. This is the key construction of covering spaces with finite covering degree.

3 Some finitely-sheeted covering spaces of the flat torus

Let a, b denote translations of the same length along the x- and y-axis in $\mathbb{E}^2$, respectively. We denote as $\mathcal{G}_1$ the group $< a, b >$ generated by a and b. Since $\mathcal{G}_1$ is Abelian, any $x \in \mathcal{G}_1$ can be uniquely written as $x = a^\alpha b^\beta$, where $\alpha, \beta \in \mathbb{Z}$. The *length* $\lambda_1(x)$ of x can thus be defined as the sum $|\alpha| + |\beta|$. We are going to study some covering spaces of the flat torus $\mathbb{T}_1^2 := \mathbb{E}^2/\mathcal{G}_1$. The original domain of $\mathcal{G}_1$ is denoted by $\mathcal{D}_1$. The edge length of $\mathcal{D}_1$ is denoted by $l = \delta(\mathcal{G}_1) = \|a\| = \|b\|$.

In the sequel, the inverse x^{-1} of an element x of $\mathcal{G}_1$ will alternatively be denoted by $\overline{x}$.

All subgroups are normal, since $\mathcal{G}_1$ is Abelian. Let us consider the subgroup $\mathcal{G}_2$ consisting of elements of $\mathcal{G}_1$ of even length. It is easy to see that $\mathcal{G}_2 = \langle \overline{a}b, a\overline{b} \rangle$.

Lemma 1 *$\mathcal{G}_2$ is a subgroup of index 2 in $\mathcal{G}_1$.*

Proof. Let $\varphi : \mathcal{G}_1 \to \mathbb{Z}_2$ be the group homomorphism defined by $\varphi(x) = \lambda_1(x) \mod 2$. The subgroup $\mathcal{G}_2$ is the kernel of φ. According to the First Isomorphism Theorem (see, e.g., [1]) $\ker\varphi$ is a normal subgroup of $\mathcal{G}_1$, and $G/\ker\varphi \cong \phi(G)$. Therefore, $\mathcal{G}_1/\mathcal{G}_2 \cong \mathbb{Z}_2$. □

In an inductive way, for $k > 1$, if $\mathcal{G}_{2^{k-1}}$ is a subgroup of index 2^{k-1} of $\mathcal{G}_1$ generated by two elements, i.e., $\mathcal{G}_{2^{k-1}} = \langle g, h \rangle$, $g, h \in \mathcal{G}_1$, we construct the subgroup $\mathcal{G}_{2^k}$ of $\mathcal{G}_{2^{k-1}}$ as follows: $\mathcal{G}_{2^k} = \langle \overline{g}h, gh \rangle$. The proof of the following lemma is similar to that of Lemma 1.

Lemma 2 *For any $k > 0$, $\mathcal{G}_{2^k}$ is a subgroup of index 2^k in $\mathcal{G}_1$.*

By construction, for any $k \geq 0$, the generators of $\mathcal{G}_{2^k}$ are two translations whose vectors are orthogonal and of equal length $\delta(\mathcal{G}_{2^k}) = \sqrt{2}^k l$. Following Section 2, the orbits of $\mathcal{G}_{2^k}$ are isomorphic to $\mathbb{Z}^2$ and the original domain $\mathcal{D}_{2^k}$ of $\mathcal{G}_{2^k}$ is a half-open square of edge length $\delta(\mathcal{G}_{2^k})$ (See Figure 1). For simplicity, x both denotes an element x of $\mathcal{G}_1$ and the image xO of the origin O by x.

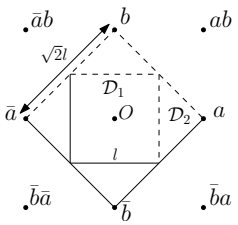


Figure 1: The original domains $\mathcal{D}_1$ and $\mathcal{D}_2$ of $\mathcal{G}_1$ and $\mathcal{G}_2$ respectively.

For any $k > 0$, $\mathbb{T}_{2^k}^2 = \mathbb{E}^2/\mathcal{G}_{2^k}$ is a flat torus, with the corresponding projection map $\pi_{2^k} : \mathbb{E}^2 \to \mathbb{T}_{2^k}^2$. $\mathbb{T}_{2^k}^2$

is a covering space of $\mathbb{T}_1^2$ together with the covering map $\rho_k := \pi_1 \circ \pi_{2^k}^{-1}$. Since the index of $\mathcal{G}_{2^k}$ in $\mathcal{G}_1$ is 2^k, we get the following result:

Proposition 3 *For any $k > 0$, $\mathbb{T}_{2^k}^2$ is a 2^k-sheeted covering space of $\mathbb{T}_1^2$.*

4 Delaunay triangulations via 2^k-sheeted covering spaces, $k > 0$

As in [5, 6], we stick to the definition of a *triangulation* of a topological space $\mathbb{Y}$ as a geometric simplicial complex K such that $K = \cup_{\sigma \in K} \sigma$ is homeomorphic to $\mathbb{Y}$. A triangulation defined by a point set $\mathcal{P}$ is a triangulation whose set of vertices is identical to $\mathcal{P}$.

The Delaunay triangulation of a point set $\mathcal{S}$ in $\mathbb{E}^2$ is a triangulation of the convex hull of $\mathcal{S}$ such that the circumscribing disk of any triangle does not contain any point of $\mathcal{P}$ in its interior. It can be defined uniquely even in denegerate cases [7].

We use the following definition of a Delaunay triangulation of a flat torus $\mathbb{T}^2 = \mathbb{E}^2/\mathcal{G}$ defined by a set of points $\mathcal{S} \in \mathbb{E}^2$.

Definition 1 ([5]) (Delaunay triangulation of $\mathbb{T}^2$). *Let $DT(\mathcal{GS})$ be the Delaunay triangulation of $\mathcal{GS}$ in $\mathbb{E}^2$. If $\pi(DT(\mathcal{GS}))$ is a simplicial complex in $\mathbb{T}^2$, then we call it the* Delaunay triangulation *of $\mathbb{T}^2$ defined by $\mathcal{S}$ and we denote it by $DT_{\mathbb{T}}(\mathcal{S})$.*

As shown in [5] this Delaunay triangulation is not always defined. However, there are always some covering spaces of the torus in which the Delaunay triangulation can be defined and computed.

Let $\Delta(\mathcal{S})$ denote the diameter of the largest disk in $\mathbb{E}^2$ that does not contain any point of a set $\mathcal{S}$ in its interior. Note that for any $p \in \mathbb{E}^2, \Delta(\mathcal{G}_{2^k} p) = \sqrt{2}^{k+1} l$.

Proposition 4 ([5]) *If $\Delta(\mathcal{GS}) < \frac{\delta(\mathcal{G})}{2}$, then $\pi(DT(\mathcal{GS}'))$ is a triangulation of $\mathbb{T}^2$ for any finite $\mathcal{S}' \supseteq \mathcal{S}$.*

Proposition 5 *If $\Delta(\mathcal{G}_1\mathcal{S}) < \frac{1}{2}\delta(\mathcal{G}_{2^k})$, then $\pi_{2^k}(DT(\mathcal{G}_1\mathcal{S} \cup \mathcal{G}_{2^k}\mathcal{T}))$ is a triangulation of $\mathbb{T}_{2^k}^2$ for any finite point set $\mathcal{T}$ in $\mathbb{E}^2$. In particular, then $\pi_{2^k}(DT(\mathcal{G}_1\mathcal{S}'))$ is a triangulation of $\mathbb{T}_{2^k}^2$ for any finite $\mathcal{S}' \supseteq \mathcal{S}$, $\mathcal{S}' \in \mathbb{E}^2$.*

Proof. Let $\mathcal{S}_{2^k}$ denote the set $\mathcal{G}_1\mathcal{S} \cap \mathcal{D}_{2^k}$. By Proposition 4, if $\Delta(\mathcal{G}_{2^k}\mathcal{S}_{2^k}) < \frac{1}{2}\delta(\mathcal{G}_{2^k})$, then $\pi_{2^k}(DT(\mathcal{G}_{2^k}\mathcal{S}_{2^k} \cup \mathcal{G}_{2^k}\mathcal{T}))$ is a triangulation. Let us note that $\mathcal{G}_{2^k}\mathcal{S}_{2^k} = \mathcal{G}_1\mathcal{S}$.

For $\mathcal{S}' \supseteq \mathcal{S}$, $\mathcal{G}_1\mathcal{S}' = \mathcal{G}_1\mathcal{S} \cup \mathcal{G}_1(\mathcal{S}' \setminus \mathcal{S}) = \mathcal{G}_1\mathcal{S} \cup \mathcal{G}_{2^k}(\mathcal{G}_1(\mathcal{S}' \setminus \mathcal{S}) \cap \mathcal{D}_{2^k})$. It remains to take $\mathcal{T} := \mathcal{G}((\mathcal{S}' \setminus \mathcal{S}) \cap \mathcal{D}_{2^k})$. □

By Proposition 5, if the maximum empty disk diameter $\Delta(\mathcal{G}_1\mathcal{S})$ is smaller than $\frac{1}{2}\delta(\mathcal{G}_4) = l$, then $\pi_4(DT(\mathcal{G}_1\mathcal{S} \cup \mathcal{G}_4\mathcal{T})))$ is a simplicial complex for any finite $\mathcal{T}$ in $\mathbb{E}^2$. If it is smaller than $\frac{1}{2}\delta(\mathcal{G}_2) = \frac{\sqrt{2}}{2}l$, then $\pi_2(DT(\mathcal{G}_1\mathcal{S} \cup \mathcal{G}_2\mathcal{T})))$ is a simplicial complex.

Note that, for $k > 3$, $\frac{1}{2}\delta(\mathcal{G}_{2^k}) = \frac{1}{2}\sqrt{2}^k l > \sqrt{2}l = \Delta(\mathcal{G}_1 p)$, for any $p \in \mathcal{S}$. Therefore, by Proposition 5, $\pi_{2^k}(DT(\mathcal{G}_1\mathcal{S}))$ is a simplicial complex. Let us now consider the case $k = 3$.

Corollary 6 *For any finite set $\mathcal{P} \subset \mathcal{D}_1$ of $n > 1$ points, $\pi_8(DT(\mathcal{G}_1\mathcal{P}))$ is a simplicial complex.*

Proof. The maximum empty disk diameter $\Delta(\mathcal{G}_1 p)$, for any $p \in \mathcal{P}$, is $\sqrt{2}l$, that is equal to $\frac{1}{2}\delta(\mathcal{G}_8)$ (See Figure 2). We are going to show that $\Delta(\mathcal{G}_1\mathcal{P})$ is strictly less than $\sqrt{2}l$.

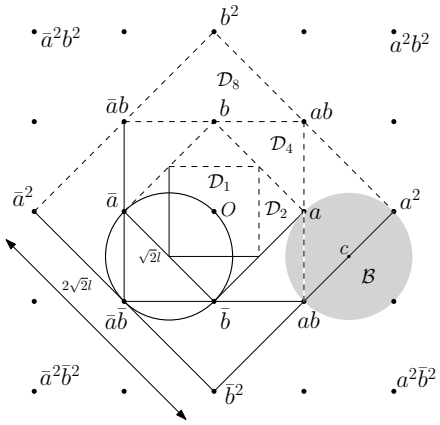


Figure 2: The original domain $\mathcal{D}_8$. Maximum empty disk with diameter $\Delta(\mathcal{G}_1 O)$.

Let us suppose that $\mathcal{B}$ is a disk with diameter $\sqrt{2}l$ centered at some point $c \in \mathbb{E}^2$ and containing no point of $\mathcal{G}_1\mathcal{P}$ in its interior (See Figure 2). $\mathcal{B}$ contains $\mathcal{D}_1(c)$. Since $\mathcal{D}_1(c)$ is a half-open square, there is only one point of $\mathcal{D}_1(c)$ on the boundary of $\mathcal{B}$. For p and $q \neq p, q \in \mathcal{P}$, there is a representative of $\mathcal{G}_1 p$ and a representative of $\mathcal{G}_1 q$ in $\mathcal{D}_1(c)$. At least one of these representatives lies in the interior of $\mathcal{B}$. This contradicts that the interior of $\mathcal{B}$ is empty.

If we add more points, the diameter of the largest empty disk cannot become larger. By Proposition 5, $\pi_8(DT(\mathcal{G}_1\mathcal{P}))$ is a simplicial complex. □

Algorithm. For a finite set of points $\mathcal{P} \subset \mathcal{D}_1$, the algorithm of [5], inspired by the standard incremental algorithm [2], computes $DT_{\mathbb{T}}(\mathcal{G}_1\mathcal{P})$ via a 9-sheeted covering space. As soon as the condition of Proposition 4 is fulfilled upon insertion of a point, it switches to computing in $\mathbb{T}^2$.

Assuming that $\mathcal{P}$ contains at least two points, we modify this algorithm and use the covering spaces $\mathbb{T}_8^2, \mathbb{T}_4^2$, and $\mathbb{T}_2^2$ using Proposition 5 and Corollary 6.

• The algorithm starts by precomputing $\pi_8(DT(\mathcal{G}_1\{p,q\}))$, for any $\{p,q\} \subset \mathcal{P}$. Then it adds points one by one in the Delaunay triangulation of the 8-sheeted covering space $\mathbb{T}^2_8$.

• If after insertion of some set of points $\mathcal{S} \subseteq \mathcal{P}$, the maximum empty disk diameter becomes less than l, then $\pi_4(DT(\mathcal{G}_1\mathcal{S}'))$ is guaranteed to be a triangulation for any finite $\mathcal{S}' \supseteq \mathcal{S}$. So, we can discard all redundant periodic copies of simplices of $\pi_8(DT(\mathcal{G}_1\mathcal{S}))$ and switch to incrementally computing $\pi_4(DT(\mathcal{G}_1\mathcal{S}'))$ in the 4-sheeted covering space $\mathbb{T}^2_4$ for $\mathcal{S} \subseteq \mathcal{S}' \subseteq \mathcal{P}$.

• Similarly, if after some more insertions the maximum empty disk diameter becomes less than $\frac{\sqrt{2}}{2}l$, then we switch to incrementally computing triangulations in the 2-sheeted covering space $\mathbb{T}^2_2$.

• If it becomes less than $\frac{1}{2}l$, then we switch to computing $\pi(DT(\mathcal{G}_1\mathcal{P}))$ in $\mathbb{T}^2$.

For any $\mathcal{S} \subset \mathcal{P}$, $\pi_{2^k}(DT(\mathcal{G}_1\mathcal{S}))$ contains 2^k periodic copies of each point of $\mathcal{S}$. Hence, using covering spaces whose number of sheets is as small as possible improves the efficiency of the algorithm, even though the asymptotic complexity is unchanged: it stays randomized worst-case time and space optimal [5].

5 Future work. Covering spaces and Delaunay triangulations of the double torus.

Whereas flat tori studied above, which have one handle, are a Euclidean surfaces, 2-tori, i.e., tori with two handles, are hyperbolic surfaces. A 2-torus can be constructed as the orbit space under action on the hyperbolic plane $\mathbb{H}^2$ of a discrete group generated by four hyperbolic translations. Due to lack of space, we only give here a few hints on how to extend the approach above to this case. Details and proofs will be given in a forthcoming paper.

A major difference with the Euclidean case is that hyperbolic translations do not commute.

Let a, b, c, and d denote four hyperbolic translations and $\overline{a}, \overline{b}, \overline{c}$, and $\overline{d}$ their respective inverse translations. The group denoted as $\mathcal{G}_1^{\mathbb{H}} := \langle a, b, c, d \mid ab\overline{a}\overline{b}cd\overline{c}\overline{d}\rangle$ is defined as the quotient group of $\langle a, b, c, d\rangle$ by its smallest normal subgroup containing the element $ab\overline{a}\overline{b}cd\overline{c}\overline{d}$. For any $p \in \mathbb{H}^2$, $\mathcal{G}_1^{\mathbb{H}}p$ is a lattice. The fundamental domain of $\mathcal{G}_1^{\mathbb{H}}$ is a regular octagon in $\mathbb{H}^2$.

As in the Euclidean case, we can define the subgroup $\mathcal{G}_2^{\mathbb{H}}$ of $\mathcal{G}_1^{\mathbb{H}}$ consisting of the elements of even length. $\mathcal{G}_2^{\mathbb{H}}$ is a normal subgroup of index 2 of $\mathcal{G}_1^{\mathbb{H}}$, and $\mathbb{H}^2/\mathcal{G}_2^{\mathbb{H}}$ is a two-sheeted covering space of the 2-torus $\mathbb{H}^2/\mathcal{G}_1^{\mathbb{H}}$. The surface $\mathbb{H}^2/\mathcal{G}_2^{\mathbb{H}}$ does not have the same genus: it is a 3-torus.

In an inductive way, we define the normal subgroup $\mathcal{G}_{2^k}^{\mathbb{H}}, k > 1$ of index 2^k in $\mathcal{G}_1^{\mathbb{H}}$ as the subgroup of elements of even length in $\mathcal{G}_{2^{k-1}}^{\mathbb{H}}$ (the length is defined in terms of the generators of $\mathcal{G}_{2^{k-1}}^{\mathbb{H}}$). The orbit space $\mathbb{H}^2/\mathcal{G}_{2^k}^{\mathbb{H}}, k > 0$ is a 2^k-sheeted covering space of the 2-torus.

We expect that there exists $k > 1$, such that the Delaunay triangulation of $\mathbb{H}^2/\mathcal{G}_{2^k}^{\mathbb{H}}$ be well defined for any set of points in $\mathbb{H}^2$.

References

[1] M. Artin. *Algebra*. Prentice Hall, 1991.

[2] A. Bowyer. Computing Dirichlet tesselations. *Comput. J.*, 24:162–166, 1981.

[3] M. Caroli. *Triangulating Point Sets in Orbit Spaces*. Thèse de doctorat en sciences, Université de Nice-Sophia Antipolis, France, 2010. http://tel.archives-ouvertes.fr/tel-00552215/.

[4] M. Caroli and M. Teillaud. 3D periodic triangulations. In CGAL Editorial Board, editor, *CGAL User and Reference Manual*. 3.5 – 3.9 edition, 2009 – 2011. http://www.cgal.org/.

[5] M. Caroli and M. Teillaud. Computing 3d periodic triangulations. In *European Symposium on Algorithms*, volume 5757 of *LNCS*, pages 37–48, 2009.

[6] M. Caroli and M. Teillaud. Delaunay triangulations of point sets in closed Euclidean d-manifolds. In *Proc. 27th Annual Symposium on Computational Geometry*, pages 274–282, 2011.

[7] O. Devillers and M. Teillaud. Perturbations for Delaunay and weighted Delaunay 3D triangulations. *Computational Geometry: Theory and Applications*, 44:160–168, 2011.

[8] C. I. Grima and A. Màrquez. *Computational Geometry on Surfaces*. Kluwer Academic Publishers, 2001.

[9] M. Mazón and T. Recio. The computation of Voronoi diagrams on the Euclidean and spherical 2-orbifolds. In *Abstracts 10th EuroCG*, pages 12–15, 1994.

[10] T. Sousbie. The persistent cosmic web and its filamentary structure: I - theory and implementation. *Monthly Notices of the Royal Astronomical Society*, 414:350–383, 2011. Preliminary version available as http://arxiv.org/abs/arXiv:1009.4015.

[11] T. Sousbie, C. Pichon, and H. Kawahara. The persistent cosmic web and its filamentary structure: II - illustrations. *Monthly Notices of the Royal Astronomical Society*, 414:384–403, 2011. Preliminary version available as http://arxiv.org/abs/arXiv:1009.4014.

[12] J. Stillwell. *Geometry of Surfaces*. Universitext. Springer Verlag, Berlin, Heidelberg, New York, 1992.

[13] W. Thurston. *Three-Dimensional Geometry and Topology, Volume 1*. Princeton University Press, New Jersey, 1997.

[14] R. van de Weygaert, G. Vegter, H. Edelsbrunner, B. Jones, P. Pranav, C. Park, W. Hellwing, B. Eldering, N. Kruithof, E. Bos, J. Hidding, J. Feldbrugge, E. ten Have, M. van Engelen, M. Caroli, and M. Teillaud. Alpha, Betti and the Megaparsec universe: On the topology of the cosmic web. In *Transactions on Computational Science XIV*, volume 6970 of *LNCS*, pages 60–101. Springer-Verlag, 2011. http://www.springerlink.com/content/334357373166n902.

Delaunay triangulations on the word RAM: Towards a practical worst-case optimal algorithm*

Okke Schrijvers Frits van Bommel Kevin Buchin

Abstract

The Delaunay triangulation of n points in the plane can be constructed in $o(n \log n)$ time when the coordinates of the points are integers from a restricted range. However, algorithms that are known to achieve such running times had not been implemented so far. We explore ways to obtain a practical algorithm for Delaunay triangulations in the plane that runs in linear time for small integers. For this, we first implement and evaluate variants of an algorithm, BrioDC, that is known to achieve this bound. We find that our implementations of these algorithms are competitive with fast existing algorithms. Secondly, we implement and evaluate variants of an algorithm, BRIO, that runs fast in experiments. Our variants aim to avoid bad worst-case behavior and our squarified orders indeed provide faster point location.

1 Introduction

In general, constructing the Delaunay triangulation (DT) of n points in the plane takes $\Omega(n \log n)$ time. However, if the coordinates of the points are integers from a restricted range $[0, U)$ this bound no longer holds. Nonetheless, for a long time no algorithms that beat this bound were known. The breakthrough came in 2006, when Chan and Pătraşcu [8] presented an algorithm running in $O(n \log n / \log \log n)$ time, and later improved this bound to $n2^{O(\sqrt{\log \log n})}$ [9]. The best asymptotic running time to hope for is the time for sorting integers in the range $[0, U)$. A randomized and a deterministic algorithm achieving this running time in a suitable model were given by Buchin and Mulzer [5] and Löffler and Mulzer [14], respectively.

For small integers the latter two algorithms run in linear time, since we can sort integers with $U = n^{O(1)}$ in this time using radix sort. Radix sort is not only fast in theory but also runs fast in experiments. In contrast, all of the above algorithms for DTs have been only of theoretical interest so far and none of them had been implemented. Many incremental algorithms for DTs like the algorithm by Su and Drysdale [17] use an orthogonal data structure for point location and concepts related to integer sorting to compute these data structures fast. Such algorithms typically run fast in experiments and on random points but have a quadratic worst-case performance.

The goal of our work is to explore ways to obtain a practical algorithm for DTs in the plane that runs in linear time for small integers. Our approach to this is two-fold. First, we implement and evaluate variants of one of the DT algorithms that has the same asymptotic running time as sorting, namely from [5]. Secondly, we consider an algorithm (namely from [1]) and implement and evaluate variants of it, which aim to avoid typical reasons for bad worst-case behavior.

We focus on incremental constructions, more specifically, incremental constructions *con BRIO* [1], where BRIO stands for *biased randomized insertion order*. The points are inserted in random rounds of increasing size and within each round the order of the points can be chosen freely. Amenta, Choi and Rote [1] prove that the expected running time of an incremental construction using such an order is asymptotically bounded by the expected running time of a randomized incremental construction, that is, $O(n \log n)$ if an optimal point location data structure is used[1].

There are various implementations of variants of this algorithm [1, 3, 10, 13, 18]. Most of these variants actually do not use an additional point location data structure and most sort the points of a round along a space-filling curve (SFC). Such variants run in $O(n \log U)$ expected time [4][2], which for small integers is again $O(n \log n)$. In experiments these variants mostly seem to run in linear time, but unfortunately there are point sets for which this bound is tight [4]. Thus if we want an algorithm with a better worst-case performance, we will need to choose a different order.

One weakness of orders based on SFCs seems to be that the construction process does not adapt to the point distribution. In contrast, the CGAL Hilbert curve order [10][3] does. However, this order is likely to introduce some large jumps, that is, large distances between consecutive points in the order. We propose several new orders that overcome this problem and still adapt well to the point distribution. For the

*Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands.

[1] They actually prove a corresponding result in 3d, but their analysis can be extended to two-dimensional DTs (and other configuration spaces) as shown in [3].

[2] In [4] this bound is formulated in terms of the spread of the point set.

[3] CGAL now also provides a regular Hilbert curve order.

This is an extended abstract of a presentation given at EuroCG 2012. 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 in a conference with formal proceedings and/or in a journal.

purpose of comparison we also implemented various traditional SFC orders.

The algorithm by Buchin and Mulzer [5] uses an extension of BRIOs that uses nearest neighbor graphs (NNGs) to find a suitable order. While this algorithm does run in linear time in our experiments, the constant factor in the running time is large. The large constant is due to the worst-case optimal NNG construction. Therefore, we implement an additional variant of this algorithm, where we use several steps that are non-optimal but simpler.

2 Algorithms

In this section we discuss several variants of BRIO and BrioDC algorithms that we implemented.

2.1 BRIO

The difference between a regular *random incremental construction* (RIC) algorithm and BRIO, is that with BRIO the points are inserted in $\log_2 n$ rounds. First, points are added to the final round with independent probability 1/2. Then the remaining points are added with probability 1/2 to the second-to-last round and this process continues until we reach the first round and all remaining points are added. Within a round the insertion order can be chosen freely, and often is chosen as an SFC order. By sorting points in this way, the next point to be inserted is likely to be close to the previous point, hopefully resulting in faster point location.

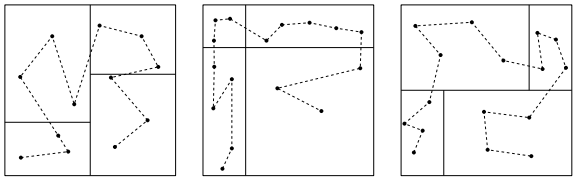
2.1.1 Existing SFCs

There is a plethora of SFCs in the literature and in our experiments, we have used the Peano, Sierpiński and Hilbert curves. In addition, CGAL provides an SFC that is similar to the Hilbert order, but first creates a vertical split on the horizontal median, and then for each half a horizontal split on their vertical medians. The quadrants are handled in Hilbert order, see Figure 1a. Note that if the two vertical medians differ substantially, the jump from the upper left to upper right quadrant can be large.

2.1.2 New SFCs

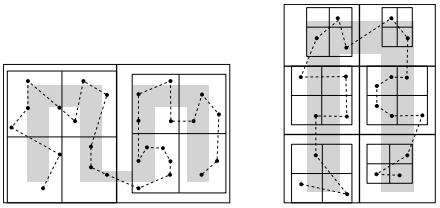
We propose several new SFCs that aim at providing a good mapping between 1-dimensional and d-dimensional space, i.e. no jumps should occur and the total summed distance between consecutive points in the order should be small.

Adaptive Hilbert order. We split the point set by the horizontal and vertical median, as a compromise between the original and CGAL Hilbert orders. This should distribute the points over all quadrants better



(a) CGAL (b) Adaptive (c) HilbertYX

Figure 1: Variants of the Hilbert SFC.



(a) Wide (b) Tall

Figure 2: Squarified Hilbert SFC.

than the original Hilbert order, and remove the jump that was seen in CGAL Hilbert, see Figure 1b.

Hilbert YX. As a variant on the CGAL Hilbert order, if we first split on the vertical median, and then on the two horizonal medians, no jump occurs when going from one quadrant to another, see Figure 1c.

Squarified Hilbert and Peano orders. If the smallest enclosing bounding box of the point set has large aspect ratio, fitting an SFC will either leave part of the domain unused, or stretch one axis. We prevent this from happening by splitting the point set into subsets that have a well-fitting square bounding box, and joining the SFCs over these point sets. This is achieved by either placing multiple curves next to each other, as in Figure 2a, or using more than four subproblems, as in Figure 2b. The bounding box is recomputed for each subproblem, so the individual subproblems may end up being split in a similar way. Squarifying can also be combined with the adaptive orders.

2.1.3 Implementation

BRIO was implemented in C++ and uses CGAL 3.6.1 to generate the DT after the rounds have been determined. It provides a means to sort a point set according to the SFCs defined in Section 2.1.2. We use CGAL for this part of the experiments to obtain a direct comparison to the CGAL Hilbert order, for which we use the original implementation: `hilbert_sort_2`. The code of CGAL was slightly modified in order to gather metrics.

2.2 BrioDC

BrioDC [5], or BRIO with Dependent Choices, reduces the problem of constructing the DT of a point set P to constructing nearest-neighbor graphs

(NNGs). Let $\mathrm{NNG}_{\leq r}$ be the NNG of the points from the first r rounds. Where BRIO assigns points to a round with equal probabilities, BrioDC makes sure that for every point p that is allocated to round $r > 1$, at least one point q in the connected component of $\mathrm{NNG}_{\leq r}$ is added in an earlier round. We can use the location of q to quickly locate the triangle in which point p should be inserted.

2.2.1 Variations

We use two different algorithms for constructing nearest neighbor graphs.

Linear-time algorithm. In order to find the NNG of a point set, we use a series of intermediate data structures. We start by sorting the point in z-order [15] using radix sort, and from this we generate a *compressed quadtree* using the approach of Chan [7]. The compressed quadtree can be used to find the *well-separated pair decomposition (WSPD)* from which we can compute the NNG using the linear time algorithm of Callahan and Kosaraju [6].

$O(n \log U)$-time algorithm. Although each of the steps in the previous paragraph are done in linear time, the constants in the computation are quite high, so alternatively we have used straightforward implementations that run in near-linear time. The compressed quadtree can be constructed top-down and compressed in $O(n \log U)$ time. For computing the NNG from the WSPD, we use the simpler approach by Har-Peled [12] that runs in $O(n(\log n + \log U))$. Since $\log n$ is bounded by $\log U$, this can be simplified to $O(n \log U)$.

2.2.2 Implementation

We have implemented the BrioDC algorithm in C++. For the base case of the DT and for the incremental insertion of points into the triangulation we use the *Triangle* library[4] [16] version 1.6. In Triangles implementation of an incremental construction of DTs, whenever a point is inserted into the triangulation, Triangle searches for the triangle containing that point. It is possible to supply a guiding triangle, which serves as a start point for the point location query. In each round we use the NNG to supply an appropriate triangle. Other than the modifications to gather the metrics, we have made no changes to the library.

3 Results

We have tested all algorithms on point sets with different distributions; namely uniform (in a square), checkers, bivariate normal with independent coordinates, Kuzmin and line singularity distributions. Checkers is a distribution on a checkers board where the 1/8 of the points are distributed uniformly on white squares and 7/8 of the points are distributed uniformly on black squares. The Kuzmin and line singularity distributions are described by Blelloch et al. [2]. For each distribution and n we run 10 tests on different datasets and report the average of the results. We focus on the running time for the comparison of the algorithms.

[4] `http://www.cs.cmu.edu/~quake/triangle.html`

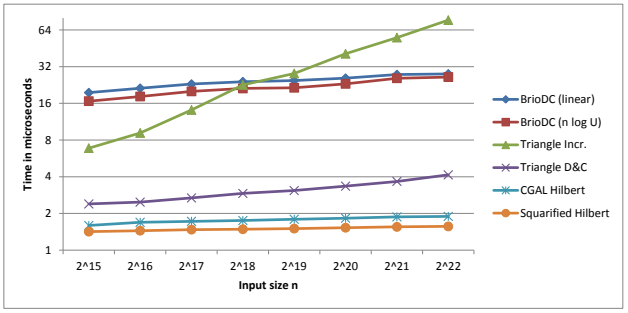


Figure 3: The runtime per point for the different algorithms over all distributions.

The experiments were performed on a 64-bit 16 core Intel Xeon L5520 server running Linux (2.6.35) operating system with 11.7 gigabytes of RAM. Only 1 core was used for the experiments.

We first compare the various insertion orders for incremental constructions con BRIO using the running time per point in microseconds for $n = 2^{22}$ over all distributions. The squarified and original Hilbert orders perform best. In general, the squarified orders (Squarified Hilbert 1.57, Squarified Peano 1.58) are slightly slower than the original orders (Hilbert 1.51, Peano 1.57, Sierpiński 1.52). This can be attributed to the higher construction time. Most remaining orders are slower by 25-80% (Adaptive Hilbert 2.01, Adaptive Peano 2.17, HilbertYX 1.95, CGAL Hilbert 1.89, Squarified Adaptive Hilbert 2.71, Squarified Adaptive Peano 2.78), but this drops to 13-25% if we exclude the line distribution (for which all adaptive curves perform poorly).

Next, we compare our algorithms with existing ones. The BRIO orders fall into two categories and the orders in a category have similar running time. Therefore, from the original and squarified orders we only show Squarified Hilbert and from the Adaptive orders we show CGAL Hilbert. Figure 3 shows the running time per point of the algorithms over all distributions on a log-log-scale. Table 1 shows the running times for the different distributions and input size 2^{20}, 2^{21} and 2^{22}. For all algorithms based on con BRIO (including BrioDC) we would expect a constant, or nearly constant running time per point. This seems to be indeed the case, although it increases very slightly with the input size. The running time of the near-linear $O(n \log U)$ implementation of BrioDC is lower than the linear time implementation, but it

Input Size	DCLin	DCLog	TrInc	D&C	CHil	SqHil	Hil
Uniform							
2^{20}	25.56	21.70	35.73	3.06	1.58	1.51	1.46
2^{21}	26.71	24.43	50.46	3.39	1.61	1.53	1.48
2^{22}	28.36	26.01	74.79	3.98	1.64	1.56	1.50
Normal							
2^{20}	25.57	22.96	30.66	3.06	1.60	1.52	1.47
2^{21}	27.33	25.49	41.30	3.34	1.65	1.55	1.49
2^{22}	27.95	27.02	68.21	3.97	1.65	1.55	1.49
Kuzmin							
2^{20}	25.25	23.81	34.27	3.17	1.63	1.55	1.48
2^{21}	29.02	28.09	49.14	3.53	1.67	1.58	1.51
2^{22}	28.75	27.69	70.31	3.98	1.68	1.59	1.51
Checkers							
2^{20}	25.39	22.25	43.65	3.23	1.57	1.50	1.44
2^{21}	27.58	26.45	59.91	3.58	1.61	1.51	1.46
2^{22}	27.07	25.14	74.19	3.94	1.62	1.53	1.47
Line							
2^{20}	26.85	24.86	59.75	4.22	2.76	1.56	1.51
2^{21}	27.13	23.83	76.19	4.49	2.82	1.59	1.53
2^{22}	27.19	25.65	97.93	4.86	2.87	1.60	1.54

Table 1: The runtime per point in microseconds for BrioDC linear (DCLin) and $O(n \log U)$ (DCLog), Triangle incremental (TrInc) and divide-and-conquer (D&C), CGAL Hilbert (CHil), Squarified Hilbert (SqHil) and Hilbert (Hil).

grows faster. For 2^{19} and more points, BrioDC is faster than the incremental construction algorithm of Triangle. For 2^{22} points BrioDC is only a factor 5.5-7 slower than the divide-and-conquer algorithm of Triangle. Other commonly used $O(n \log n)$-time algorithms come within a factor of 1.5 to 10 of this running time [11]. BrioDC is competitive with these algorithms, and the 5.5-7 factor will decrease further with increasing input sizes. The fastest algorithms in our experiments are the algorithms using a BRIO. The Hilbert order, the squarified Hilbert order and BrioDC show little dependency on the input distribution. Triangle's incremental and divide-and-conquer algorithms, and CGAL Hilbert show a large dependency and considerably slow down for the line distribution.

We have also measured the cost of *point location*, *updating the triangulation* and the running time of the *individual parts* of BrioDC. Point location cost is reduced by approximately 3% by using squarified orders compared to the original Hilbert order. The cost for updating the triangulation is similar for all approaches. For BrioDC, about 84% of the running time is spent on computing the NNGs (18% for the quadtree, 29% for the WSPD, and 37% for the NNG from the WSPD).

Summarizing, we have successfully shown the practicality of an $O(n)$-time algorithm for computing the Delaunay triangulation of points with bounded integer coordinates, as well as improved heuristics on non-optimal but faster algorithms. While currently BrioDC is still outperformed by $O(n \log n)$-time algorithms, our implementation is competitive and could be improved upon with a faster NNG algorithm.

Acknowledgements. The authors would like to thank Tal Milea for her valuable contributions.

References

[1] N. Amenta, S. Choi, and G. Rote. Incremental constructions con BRIO. In *Proc. 19th Annu. ACM Sym. Comp. Geom. (SoCG)*, pages 211–219. ACM, 2003.

[2] G. Blelloch, G. Miller, J. Hardwick, and D. Talmor. Design and implementation of a practical parallel Delaunay algorithm. *Algorithmica*, 24:243–269, 1999.

[3] K. Buchin. *Organizing Point Sets: Space-Filling Curves, Delaunay Tessellations of Random Point Sets, and Flow Complexes.* PhD thesis, Free University Berlin, 2007.

[4] K. Buchin. Constructing Delaunay triangulations along space-filling curves. In *Proc. 17th Annu. European Sympos. Algorithms (ESA)*, pages 119–130. Springer-Verlag, 2009.

[5] K. Buchin and W. Mulzer. Delaunay triangulations in $O(sort(n))$ time and more. *J. ACM*, 58:6:1–6:27, 2011.

[6] P. B. Callahan and S. R. Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *J. ACM*, 42(1):67–90, 1995.

[7] T. M. Chan. Well-separated pair decomposition in linear time? *Inf. Proc. Lett.*, 107(5):138–141, 2008.

[8] T. M. Chan and M. Pătraşcu. Transdichotomous results in computational geometry, I: Point location in sublogarithmic time. *SIAM J. Comput.*, 39(2):703–729, 2009.

[9] T. M. Chan and M. Pătraşcu. Voronoi diagrams in $n2^{O(\sqrt{\lg \lg n})}$ time. In *Proc. 39th Annu. ACM Sym. Theory Comput. (STOC)*, pages 31–39. ACM, 2007.

[10] C. Delage. Spatial sorting. In CGAL Editorial Board, editor, *CGAL User and Reference Manual*. 2007.

[11] O. Devillers. The Delaunay hierarchy. *Int. J. Found. CS*, 13(2):163–180, 2002.

[12] S. Har-Peled. *Geometric Approximation Algorithms.* Mathematical Surveys and Monographs. AMS, 2011.

[13] Y. Liu and J. Snoeyink. A comparison of five implementations of 3d Delaunay tesselation. In *Comb. and Comp. Geom.*, volume 52 of *MSRI Publications*, pages 439–458. Cambridge University Press, 2005.

[14] M. Löffler and W. Mulzer. Triangulating the square: quadtrees and Delaunay triangulations are equivalent. *Proc. 22nd Annu. ACM-SIAM Sym. Discrete Algorithms (SODA)*, pages 1759–1777, 2011.

[15] G. Morton. A computer oriented geodetic data base and a new technique in file sequencing. Technical report, IBM Ltd., Ottawa, Canada, 1966.

[16] J. Shewchuk. Triangle: Engineering a 2d quality mesh generator and delaunay triangulator. In *Appl. Comp. Geom. Towards Geometric Engineering*, volume 1148 of *LNCS*, pages 203–222. Springer, 1996.

[17] P. Su and R. Drysdale. A comparison of sequential Delaunay triangulation algorithms. *Comput. Geom. Theory Appl.*, 7:361–386, 1997.

[18] S. Zhou and C. B. Jones. HCPO: an efficient insertion order for incremental Delaunay triangulation. *Inf. Proc. Lett.*, 93(1):37–42, 2005.

Equating the witness and restricted Delaunay complexes

Jean-Daniel Boissonnat, Ramsay Dyer, Arijit Ghosh, and Steve Y. Oudot

Abstract

It is a well-known fact that the restricted Delaunay and witness complexes may differ when the landmark and witness sets are located on submanifolds of $\mathbb{R}^d$ of dimension 3 or more. Currently, the only known way of overcoming this issue consists of building some crude superset of the witness complex, and applying a greedy sliver exudation technique on this superset. Unfortunately, the construction time of the superset depends exponentially on the ambient dimension, which makes the witness complex based approach to manifold reconstruction impractical. This work provides an analysis of the reasons why the restricted Delaunay and witness complexes fail to include each other. From this a new set of conditions naturally arises under which the two complexes are equal.

1 Introduction

Various subcomplexes of the Delaunay triangulation have been used with success for approximating k-submanifolds of $\mathbb{R}^d$ from finite collections of sample points. Perhaps the most popular one in small dimensions ($k \in \{1, 2\}$ and $d \in \{2, 3\}$) is the so-called *restricted Delaunay complex*, defined as the subcomplex spanned by those Delaunay simplices whose dual Voronoi faces intersect the manifold. Its main attraction is that it can be shown to be a faithful approximation to the manifold underlying the data points, in terms of topology (ambient isotopy), of geometry (Hausdorff proximity), and of differential quantities (normals, curvatures, etc), all this under sampling assumptions which only constrain the sampling density [1]. These qualities explain its success in the context of curve and surface meshing or reconstruction, where it is used either as a data structure for the algorithms, or as a mathematical tool for their analysis, or both — see [6] for a survey.

The story becomes quite different when the data is sitting in higher dimensions, where two major bottlenecks appear: (i) The nice structural properties mentioned above no longer hold when the dimension k of the submanifold is 3 or more. In particular, normals may become arbitrarily wrong [7], and more importantly the topological type of the complex may deviate significantly from the one of the manifold [5]. (ii) It is not known how to compute the restricted Delaunay complex without computing the full-dimensional Delaunay triangulation or at least its restriction to some local d-dimensional neighborhood. The resulting construction time incurs an exponential dependence on the ambient dimension d, which makes the complex a prohibitively costly data structure in practice. Another issue is the high degree of the algebraic operations involved in the construction of Delaunay triangulations. Specifically, we need to evaluate signs of polynomials of degree $d + 2$ in the input variables, which is prohibitive for large d.

To address problem (i), Cheng *et al.* [7] suggested to use *weighted* Delaunay triangulations. The intuition underlying their approach is simple: when the restricted Delaunay complex contains badly shaped simplices, called *slivers*, its behavior in their vicinity may be arbitrarily bad: wrong normals, wrong local homology, and so on. By carefully assigning weights to the data points, one can remove all the slivers from the restricted Delaunay complex and thus have it recover its good structural properties. This idea was carried on in subsequent work [5, 4], and it is now considered a fairly common technique in Delaunay-based manifold reconstruction.

Yet, the question of computing a good set of weights given the input point cloud remains. This question is closely connected to problem (ii) above, since determining which simplices are the slivers to be removed requires that the restricted Delaunay complex be computed first. To address this issue, it has been proposed to build some superset of the restricted Delaunay complex, from which the slivers are removed [5, 7]. After the operation the superset becomes equal to the restricted Delaunay complex, and thus it shares its nice properties. Unfortunately, the supersets proposed so far were pretty crude, and their construction times depended exponentially on the ambient dimension d, which made the approach intractable in practice.

To circumvent the building time issue, Boissonnat and Ghosh [4] proposed to use a different subcomplex of the Delaunay triangulation, called the *tangential complex*, whose construction reduces to computing local Delaunay triangulations in (approximations of) the k-dimensional tangent spaces of the manifold at the sample points. Once these local triangulations have been computed, the tangential complex is assembled by gluing them together. Consistency issues between

This is an extended abstract of a presentation given at EuroCG 2012. 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 in a conference with formal proceedings and/or in a journal.

the local triangulations may appear, which are solved once again by a careful weight assignment over the set of data points to remove slivers. The benefit is that the slivers to be removed are determined directly from the complex, not from some superset, so the complexity of the sliver removal phase reduces to a linear dependence on the ambient dimension d, while keeping an exponential dependence on the intrinsic dimension k. This makes the approach tractable under the common assumption that the data points live on a manifold with small intrinsic dimension, embedded in some potentially very high-dimensional space. The obtained complex is not the restricted Delaunay complex in general, and the question of whether the latter can be effectively retrieved remains open. Also, the algebraic degree of the polynomials involved in the construction of the tangential complex is $k+2$, which is a neat improvement over $d+2$ but still limits the practical use of the tangential complex to manifolds of small dimensions.

Enter the witness complex. In light of the apparent hardness of manifold reconstruction, researchers have turned their focus to the somewhat easier problem of inferring some topological invariants of the manifold without explicitly reconstructing it. Their belief was that more lightweight data structures would be appropriate for this simpler task, and it is in this context that Vin de Silva introduced the *witness complex* [8]. Given a point cloud W, his idea was to carefully select a subset L of landmarks on top of which the complex would be built, and to use the remaining data points to drive the complex construction. More precisely, a point $w \in W$ is called a *witness* for a simplex $\sigma \in 2^L$ if no point of L is closer to w than w is to the vertices of σ, i.e. if there is a ball centered at w that includes the vertices of σ, but no other points from L. The witness complex is then the largest abstract simplicial complex that can be assembled using only witnessed simplices. The geometric test for being a witness can be viewed as a simplified version of the classical Delaunay predicate, and its great advantage is to require a mere comparison of (squared) distances (hence the evaluation of the signs of polynomials of degree 2 in the input variables). As a result, witness complexes can be built in arbitrary metric spaces, and the construction time is bound to the size of the input point cloud rather than to the dimension d of the ambient space.

Since its introduction, the witness complex has attracted interest [2, 5, 9, 10], which can be explained by its close connection to the Delaunay triangulation and restricted Delaunay complex. In his seminal paper [8], de Silva showed that the witness complex is always a subcomplex of the Delaunay triangulation $\mathrm{Del}(L)$, provided that the data points lie in some Euclidean space or more generally in some Riemannian manifold of constant sectional curvature. With applications to reconstruction in mind, Attali *et al.* [2] and Guibas and Oudot [10] considered the case where the data points lie on or close to some k-submanifold of $\mathbb{R}^d$, and they showed that the witness complex is equal to the restricted Delaunay complex when $k=1$, and a subset of it when $k=2$. Unfortunately, the case of 3-manifolds is once again problematic, and it is now a well-known fact that the restricted Delaunay and witness complexes may differ significantly (no respective inclusion, different topological types, etc) when $k \geq 3$ [11]. To overcome this issue, Boissonnat, Guibas and Oudot [5] resorted to the sliver removal technique described above on some superset of the witness complex, whose construction incurs an exponential dependence on d. The state of affairs as of now is that the complexity of witness complex based manifold reconstruction is exponential in d, and whether it can be made only polynomial in d (while still exponential in k) remains an open question.

2 Terminology and notation

We write $\tau \leq \sigma$ to indicate that τ is a (not necessarily proper) face of simplex σ. Let $L \subset \mathbb{R}^d$ be a finite set and let $X \subset \mathbb{R}^d$ be an arbitrary set.

Definition 2.1 *A point $x \in X$ is a* witness *for $\sigma \subset L$ if*

$$\|p-x\| \leq \|q-x\| \quad \forall p \in \sigma \text{ and } q \in L \setminus \sigma.$$

A Delaunay centre *for σ is a point x that satisfies*

$$\|p-x\| \leq \|q-x\| \quad \forall p \in \sigma \text{ and } q \in L. \qquad (1)$$

If a simplex σ has a witness, we say that it is witnessed.

Note that a Delaunay centre is also a witness; the relaxed qualification on q in Equation (1) simply serves to demand that $\|p-x\| = \|p'-x\|$ for all $p, p' \in \sigma$. The formulation in terms of an inequality is convenient in the subsequent developments.

Definition 2.2 *The* witness complex *of $L \subset \mathbb{R}^d$ with respect to X is the abstract simplicial complex, $\mathit{Wit}(L,X)$, on L defined by*

$$\sigma \in \mathit{Wit}(L,X) \iff \text{every subsimplex } \tau \leq \sigma \text{ has a witness in } X.$$

The Delaunay complex of L restricted to X *is the abstract simplicial complex, $\mathit{Del}(L,X)$, on L defined by*

$$\sigma \in \mathit{Del}(L,X) \iff \text{every subsimplex } \tau \leq \sigma \text{ has a Delaunay centre in } X.$$

We will also use the relaxed notion of the witness complex, which was also introduced by de Silva [8].

Definition 2.3 *Assume $\rho \geq 0$. A point $x \in X$ is a* ρ-witness *for $\sigma \subset L$ if*

$$\|p - x\| \leq \|q - x\| + \rho \quad \forall p \in \sigma, \text{ and } q \in L \setminus \sigma.$$

A ρ-Delaunay centre *for σ is a point x that satisfies*

$$\|p - x\| \leq \|q - x\| + \rho \quad \forall p \in \sigma, \text{ and } q \in L.$$

Using ρ-witnesses we obtain a superset of the witness complex

Definition 2.4 *The* ρ-witness complex, *$Wit^{\rho}(L, X)$, is the abstract simplicial complex on L defined by*

$$\sigma \in Wit^{\rho}(L, X) \iff \text{every subsimplex } \tau \leq \sigma \text{ has a } \rho\text{-witness in } X.$$

The ρ-Delaunay complex of L restricted to X *is the abstract simplicial complex, $Del^{\rho}(L, X)$, on L defined by*

$$\sigma \in Del^{\rho}(L, X) \iff \text{every subsimplex } \tau \leq \sigma \text{ has a } \rho\text{-Delaunay centre in } X.$$

de Silva [8] also showed that $\mathrm{Wit}^{\rho}(L, \mathbb{R}^d) = \mathrm{Del}^{\rho}(L, \mathbb{R}^d)$. Our motivation for introducing $\mathrm{Wit}^{\rho}(L, W)$ is that we want a discrete set of witnesses W representing an embedded manifold $\mathbb{M}$. We would like to be able to relax the Delaunay condition without changing the Delaunay complex. For this we introduce the protection idea:

Definition 2.5 *A point $x \in X$ is a* δ-protected witness *for $\sigma \subset L$ if*

$$\|p - x\| < \|q - x\| - \delta \quad \forall p \in \sigma, \text{ and } q \in L \setminus \sigma.$$

If, in addition, x is a ρ-Delaunay centre for σ, then we say x is a (δ, ρ)-Delaunay centre for σ.

Protection is the opposite of relaxation, but a (δ, ρ)-Delaunay centre combines both notions.

A (δ, ρ)-Delaunay centre for σ is the centre of a closed ball $\overline{B}$ which contains all the vertices of σ and these lie within a distance ρ from $\partial\overline{B}$, and furthermore there are no points in $L \setminus \sigma$ within a distance of δ from $\partial\overline{B}$. Observe in particular that a (δ, ρ)-Delaunay centre for σ is a witness for σ, but it is not a Delaunay centre for σ. We will require $\delta > \rho$.

For p a vertex of simplex σ, we denote by σ_p the face opposite p. For a geometric simplex $\sigma \subset \mathbb{R}^d$, the *altitude* of p in σ is $D(p, \sigma) = d_{\mathbb{R}^d}(p, \mathrm{aff}(\sigma_p))$. Poorly shaped simplices are problematic for our purposes; we need to avoid them. Such simplices can be characterized by the existence of a relatively small altitude. The *thickness* of a j-simplex σ is the dimensionless quantity

$$\Upsilon(\sigma) = \begin{cases} 1 & \text{if } j = 0 \\ \min_{p \in \sigma} \frac{D(p,\sigma)}{j\Delta(\sigma)} & \text{otherwise.} \end{cases}$$

We say that σ is Υ_0-thick, if $\Upsilon(\sigma) \geq \Upsilon_0$. The constant Υ_0 that bounds the thickness of the simplices plays an important role in our analysis.

We will require some terminology related to sampling of manifolds. Given two subsets W and X of $\mathbb{R}^d$, the symmetric *Hausdorff distance* between them is defined by

$$d_H(W, X) = \max\{\sup_{w \in W} d_{\mathbb{R}^d}(w, X), \sup_{x \in X} d_{\mathbb{R}^d}(x, W)\}.$$

If W is a finite set, we say it is an ϵ-*sample* of X if $d_H(W, X) < \epsilon$. We say that ϵ is the *sampling radius*, because any ball of radius ϵ centred on X must contain a point of W. Note that W is not required to be a subset of X. We say that W is β-*sparse* if for all $w, w' \in W$, $d_{\mathbb{R}^d}(w, w') \geq \beta$.

Let $\mathbb{M} \subset \mathbb{R}^d$ be a compact manifold. An open ball $B = B_{\mathbb{R}^d}(c; r)$ is a *medial ball* at $p \in \mathbb{M}$ if it is tangent to $\mathbb{M}$ at p, and $B \cap \mathbb{M} = \emptyset$, and it is maximal in the sense that any ball which is centred on the line through p and c, and contains B, either coincides with B or intersects $\mathbb{M}$. The *local reach* at p is the infimum of the radii of the medial balls at p, and the *reach* of $\mathbb{M}$, denoted $\mathrm{rch}(\mathbb{M})$, is the infimum of the local reach over all points of $\mathbb{M}$. In order to approximate the geometry and topology with a mesh, manifolds with small reach require a higher sampling density than those with a larger reach. As is typical, an upper bound on our sampling radius will be proportional to $\mathrm{rch}(\mathbb{M})$. We require $\mathbb{M}$ to have positive reach.

3 Equating the restricted Delaunay and witness complexes

Given a smooth compact k-manifold, $\mathbb{M} \subset \mathbb{R}^d$, we establish conditions on the finite set $W \subset \mathbb{R}^d$ of *witnesses*, and the set $L \subset W$ of *landmarks*, which will guarantee the equivalence of $\mathrm{Del}(L, \mathbb{M})$ and $\mathrm{Wit}(L, W)$. These conditions are specified in terms of $\mathrm{rch}(\mathbb{M})$ and three additional parameters by which the sampling radius ϵ of W with respect to $\mathbb{M}$, and the sampling radius λ of L with respect to W are controlled. These parameters are δ, which specifies the protection of the simplices, ρ, which is a relaxation parameter, closely tied to ϵ, and Υ_0, which is a constant that places a lower bound on the thickness of the simplices.

The sampling density of L reflects the resolution of the final simplicial representation of $\mathbb{M}$. It is governed by λ, which is constrained only by Υ_0 and $\mathrm{rch}(\mathbb{M})$. The choice of λ constrains the sampling density of W.

Hypotheses 3.1 *The sampling conditions demanded of L and W are as follows:*

$\mathbf{H}_L$ *$L \subset W$ is a λ-sparse λ-sample of W with $\lambda \leq \frac{\Upsilon_0 rch(\mathbb{M})}{32}$, $\rho \leq \left(\frac{\Upsilon_0^2}{130}\right)\lambda$, $\delta \geq \left(\frac{644}{\Upsilon_0^2}\right)\rho$, and every simplex $\sigma \in Wit^{\rho}(L,W)$ is Υ_0-thick and has a (δ,ρ)-Delaunay centre in W.*

$\mathbf{H}_W$ *$W \subset \mathbb{R}^d$ is an ϵ-sample of $\mathbb{M}$ with*

$$\epsilon \leq \min\{\frac{\rho}{2}, \frac{2\lambda^2}{rch(\mathbb{M})}\}.$$

Looking past the parameters and equations, the essential demand of $\mathbf{H}_L$ is that every $\sigma \in \mathrm{Wit}^{\rho}(L,W)$ is thick and has a (δ,ρ)-Delaunay centre $x \in \mathbb{R}^d$. As mentioned above, de Silva [8] has demonstrated that every simplex in $\mathrm{Wit}^{\rho}(L,W)$ already has a ρ-Delaunay centre. Also a thickness or equivalent quality constraint on the simplices is standard and is known to be a requirement [7, 11] for the restricted Delaunay triangulation to be an accurate representation of $\mathbb{M}$. Thus the principal novelty in our criteria is the protection demand, and it yields our main structural result [3]:

Theorem 3.2 *If $\mathbf{H}_W$ and $\mathbf{H}_L$ are satisfied, then*

$$Wit(L,W) = Del(L,\mathbb{M}).$$

The protection requirement is not an unreasonable demand: As we demonstrate [3], our criteria imply that each Delaunay simplex has a Delaunay centre that has some amount of protection. It is not difficult to verify that if there is a simplex in $\mathrm{Del}(L,\mathbb{R}^d)$ that does not have a protected Delaunay centre, then there is a $(d+1)$-simplex in $\mathrm{Del}(L,\mathbb{R}^d)$. In other words, a Delaunay simplex without any protected Delaunay centre only occurs in a point set that is not in "general position". Although de Silva [8] makes no such demand, it is customary, when dealing with Delaunay triangulations, to assume that the vertex set is in general position. Thus this standard assumption is already demanding that there is some $\delta > 0$ such that the Delaunay simplices each have a δ-protected centre. The difference is that the general position assumption allows δ to be arbitrarily small, but we demand that δ be bounded below by ρ, which is in turn bounded below by ϵ. However, we may let ϵ be arbitrarily small.

Thus if ϵ is small enough we should expect that if we generate a λ-sparse λ-sample L of W, then (once non-thick simplices are exuded) it will probably meet our criteria. In this sense our sampling criteria are not unreasonable, yet when they are met we guarantee that the witness complex coincides with the restricted Delaunay complex.

References

[1] N. Amenta and M. Bern. Surface reconstruction by Voronoi filtering. *Discrete and Comp. Geom.*, 22(4):481–504, 1999.

[2] D. Attali, H. Edelsbrunner, and Y. Mileyko. Weak witnesses for Delaunay triangulations of submanifolds. In *Proc. ACM Sympos. on Solid and Physical Modeling*, pages 143–150, 2007.

[3] J.-D. Boissonnat, R. Dyer, A. Ghosh, and S. Oudot. Equating the witness and restricted Delaunay complexes. Technical Report CGL-TR-24, Computational Geometric Learning, 2011. http://cgl.uni-jena.de/Publications/WebHome.

[4] J.-D. Boissonnat and A. Ghosh. Manifold reconstruction using tangential Delaunay complexes. In *ACM SoCG*, pages 324–333, 2010.

[5] J.-D. Boissonnat, L. Guibas, and S. Oudot. Manifold reconstruction in arbitrary dimensions using witness complexes. *Discrete and Comp. Geom.*, 42(1):37–70, 2009.

[6] F. Cazals and J. Giesen. Delaunay triangulation based surface reconstruction. In J.D. Boissonnat and M. Teillaud, editors, *Effective Computational Geometry for Curves and Surfaces*, pages 231–273. Springer, 2006.

[7] S.-W. Cheng, T. K. Dey, and E. A. Ramos. Manifold reconstruction from point samples. In *ACM-SIAM SODA*, pages 1018–1027, 2005.

[8] V. de Silva. A weak characterisation of the Delaunay triangulation. *Geometriae Dedicata*, 135:39–64, 2008.

[9] V. de Silva and G. Carlsson. Topological estimation using witness complexes. In *Proc. Sympos. Point-Based Graphics*, pages 157–166, 2004.

[10] L. J. Guibas and S. Y. Oudot. Reconstruction using witness complexes. *Discrete and Comp. Geom.*, 40:325–356, 2008.

[11] S. Y. Oudot. On the topology of the restricted Delaunay triangulation and witness complex in higher dimensions. Technical report, Stanford University, November 2006. LANL arXiv:0803.1296v1 [cs.CG].

Probabilistic Lower Bounds on the Length of a Longest Edge in Delaunay Graphs of Random Points in a d-Ball *

Miguel A. Mosteiro†

1 Introduction

In this paper, we complete the study of the length of a longest Delaunay edge for points randomly distributed in multidimensional Euclidean spaces carried out in [2]. The Delaunay graph is defined over a set of n points distributed uniformly at random in a d-dimensional body of unit volume, assuming that the probability that those points are not in general position is negligible [7]. The motivation to study this problem comes from its application in energy-efficient geometric routing and flooding in wireless sensor networks (see, e.g., [6, 8, 9, 10]).

Given that the length of the longest Delaunay edge is strongly influenced by the boundary of the enclosing region, in [2] we study the problem for the surface of a d-sphere and the volume of a d-ball. The results presented in that work include upper and lower bounds for d-dimensional bodies with and without boundaries, that hold for a parametric error probability ε. Lower bounds for a d-ball were proved only for $d \leq 3$ leaving open the question for higher dimensions. We answer this question in the present work presenting a lower bound that holds in general for any $d > 1$, and we also compute the particular cases $d = 2$ and $d = 3$ to obtain the precise bound on the euclidean distance. These lower bounds are proved showing that a configuration that yields a Delaunay edge of a certain length occurs with probability at least a parameter ε. These lower bounds are computed up to constants and are asymptotically tight for $e^{-cn} \leq \varepsilon \leq n^{-c}$, for any constant $c > 0$, and $d \in O(1)$. To the best of our knowledge, the present work and [2] are the first comprehensive study of this problem.

In the following section, we overview some previous work. In Section 3 we introduce some necessary notation. The analysis is left to Section 4.

2 Related Work.

The present work completes the study of the length of a longest Delaunay edge for points randomly distributed in multidimensional Euclidean spaces in [2]. Matching (asymptotically) upper and lower bounds for points distributed over bodies with and without boundaries were proved there. For bodies with boundaries, only $d \leq 3$ was considered leaving the question for higher dimensions open until the present work.

Kozma, Lotker, Sharir, and Stupp [8] show that the asymptotic length of a longest Delaunay edge depends on the sum, σ, of the distances to the boundary of its endpoints. More specifically, their bounds are $O(\sqrt[3]{(\log n)/n})$ if $\sigma \leq ((\log n)/n)^{2/3}$, $O(\sqrt{(\log n)/n})$ if $\sigma \geq \sqrt{(\log n)/n}$, and $O((\log n)/(n\sigma))$ otherwise. Kozma et al. also show, in the same setting, that the expected sum of the squares of all Delaunay edge lengths is $O(1)$. In [4] the authors consider the Delaunay triangulation of an infinite random (Poisson) point set in d dimensional space. In particular, they study different properties of the subset of those Delaunay edges completely included in a cube $[0, n^{1/d}] \times \cdots \times [0, n^{1/d}]$. For the maximum length of a Delaunay edge in this setting, they observe that in expectation is in $\Theta(\log^{1/d} n)$.

Interest in bounding the length of a longest Delaunay edge in two-dimensional spaces has grown out of extensive algorithmic work [5, 3, 1] aimed at reducing the energy consumption of geographically routing messages in Radio Networks. Multidimensional Delaunay graphs are well studied in computational geometry from the point of view of efficient algorithms to construct them (see [7] and references therein), but only limited results are known regarding probabilistic analysis of Delaunay graphs in higher dimensions [11].

3 Preliminaries

The following notation will be used throughout. We restrict attention to Euclidean (L_2) spaces. A *d-ball* of radius r is the set of all points in a d-dimensional space that are located at distance *at most* r (the *radius*) from a given point c (the *center*). The *volume* of a d-ball (in d-space) is its d-dimensional volume. We refer to a *unit ball* as a ball of unit volume[1].

An *spherical cap of U* is the intersection of a d-ball U with a closed halfspace, bounded by a hyperplane

*This research was partially supported by Spanish MICINN grant TIN2008-06735-C02-01, Comunidad de Madrid grant S2009TIC-1692, and the National Science Foundation (CCF-0937829, CCF-1018388).

†Computer Science Department, Rutgers University, USA, and LADyR, GSyC, Universidad Rey Juan Carlos, Spain, mosteiro@cs.rutgers.edu

[1] This is in contrast with some definitions of a "unit" ball/sphere as a unit-radius ball/sphere; we find it convenient to standardize the volume/area to be 1 in all dimensions.

h, in d-space; the *base* of U is the $(d-1)$-ball B that is the intersection of h with the ball U. Let $V_d(x)$ be the d-volume of a ball cap of base diameter x, of a d-ball of volume 1. For any pair of points a, b, let $d(a,b)$ be the Euclidean distance between a and b, i.e. $d(a,b) = ||\overrightarrow{ab}||_2$. Let $D(P)$ be the Delaunay graph of a set of points P defined as follows [7, 2].

Definition 1 *Let P be a generic set of points in a d-ball B.*

(i) A set $F \subseteq P$ of $d+1$ points define the vertices of a Delaunay face *of $D(P)$ if and only if there is a d-ball B' such that F is contained in the boundary, $\partial B'$, of B' and no points of P lie in the interior of B'.*

(ii) Two points $a, b \in P$ form a Delaunay edge*, an arc of $D(P)$, if and only if there is a d-ball B' such that $a, b \in \partial B'$ and no points of P lie in the interior of B'.*

4 Lower Bounds

Theorem 1 *For any $d > 1$, let*

$$\alpha = \left(1 - e^{-\kappa_1(d)/\kappa_2(d)}\right)\left(1 - e^{-\kappa_1(d)/(2\kappa_2(d)(2d-2))}\right)$$

$$\kappa_1(d) = \frac{1}{d-1}\sum_{i=0}^{d-2}\left(\left(\frac{d}{\sqrt{d^2-1}}\right)^i - \frac{\sqrt{d^2-1}}{d}\right)$$

$$\kappa_2(d) = \left(1 + \left(\frac{2d-1}{d-1}\right)^{d-1}\frac{d}{d-1}\right).$$

For any $n > 1$ and $0 < \varepsilon \leq \alpha/e$, given the Delaunay graph $D(P)$ of a set P of n points distributed uniformly and independently at random in a unit d-ball, with probability at least ε, there is an edge $(a,b) \in D(P)$, $a, b \in P$, such that $d(a,b) \geq \rho_1/\sqrt{d-1}$, where

$$V_d(\rho_1) = \frac{\ln(\alpha/\varepsilon)}{\kappa_2(d)\,(n - 2 + \ln(\alpha/\varepsilon))}.$$

Proof. Throughout the proof, we refer to a body and its set of space points with the same name indistinctly. Let $V(X)$ be the volume of a body (or a set of space points) X. Let the unit ball where points are sampled be called U. Consider two spherical caps of U, concentric on a line ℓ, called S_1 and S_2, with bases B_1 and B_2 of diameters ρ_1 and ρ_2, and heights h_1 and h_2 respectively. Inside $S_2 \setminus S_1$, consider the following d-dimensional bodies of height $h_2 - h_1$: a cylinder C with base B_1; a cone K of base B_2; and a frustum F of bases B_2 and B_1.

Consider the body $F \setminus (C \cup K)$ evenly divided by $d-1$ planes (of $d-1$ dimensions) containing ℓ. Consider two of the bodies defined by those hyperplanes that are opposite with respect to ℓ, call them B_a and B_b.

Then, for any pair of points $a \in B_a$ and $b \in B_b$: *(i)* the points a and b are separated a distance at least $\rho_1/\sqrt{d-1}$; and *(ii)* there exists an spherical cap S that contains the points a and b in its base of diameter ρ such that $V_d(\rho) \leq V_d(\rho_2)$. Property *(i)* holds because $\rho_1/\sqrt{d-1}$ is the side of a $(d-1)$-cube inscribed in B_1. To see why property *(ii)* holds, consider the following. Without loss of generality assume that the point a is closer to B_2 than b. Then, consider a $d-1$ hyperplane h containing the line ℓ and the point a and the projection of b on h. On h, the point closest to B_2 is located above the projection of K. Hence, the property follows.

If S is void of points, the configuration described implies the existence of an empty d-ball of infinite radius with a and b in its surface which proves that $(a,b) \in D(P)$. In the following, we show that such configuration occurs with big enough probability.

Let ρ_1 be such that $V_d(\rho_1)$ is as defined in the statement of the theorem. Let h_2 be such that $V(C) = dV_d(\rho_1)/(d-1)$. Let $q = \rho_2/\rho_1$. First, we prove upper and lower bounds on q to be used later.

Claim 1 $d/\sqrt{d^2-1} \leq q \leq (2d-1)/(d-1)$.

Proof. From the volume of C, we know that $h_2/h_1 = 1 + V(C)/(h_1 V(B_1))$. Consider a cone with the same volume and base as S_1. The height of such cone, which is bigger than h_1, is $dV_d(\rho_1)/V(B_1)$. That is, $h_1 < dV_d(\rho_1)/V(B_1)$. Consider also a cylinder with the same volume and base as S_1. The height of such cylinder, which is smaller than h_1, is $V_d(\rho_1)/V(B_1)$. That is, $h_1 > V_d(\rho_1)/V(B_1)$. Replacing those bounds and using that $V(C) = dV_d(\rho_1)/(d-1)$, we get

$$\frac{d}{d-1} \leq \frac{h_2}{h_1} \leq \frac{2d-1}{d-1}. \tag{1}$$

Consider a 2-dimensional projection of the configuration described, on a plane orthogonal to B_1 (hence, B_2). Let R be the radius of U. Then, using Pythagoras' theorem, $R^2 = (\rho_2/2)^2 + (R-h_2)^2 = (\rho_1/2)^2 + (R-h_1)^2$. Subtracting,

$$\begin{aligned} q^2 &= 1 + \frac{(R-h_1)^2 - (R-h_2)^2}{(\rho_1/2)^2} \\ &\geq 1 + \left(\frac{h_2}{h_1} - 1\right)\left(1 - \frac{1}{2 - h_1/h_2}\right) \\ &= \frac{h_2}{h_1(2 - h_1/h_2)}. \end{aligned}$$

Using Inequality 1,

$$q^2 \geq \frac{d}{d-1} \cdot \frac{1}{2 - (d-1)/d} = \frac{d^2}{d^2-1}.$$

Which proves the lower bound. For the upper bound, consider the cones K_1 and K_2 inscribed in

S_1 and S_2 respectively. It can be seen that

$$V(K_1 \cup F) > V(K_2). \tag{2}$$

The volumes of K_1 and K_2 are

$$V(K_1) = \frac{h_1 V(B_1)}{d} = \frac{h_1 C(d-1)\rho_1^{d-1}}{d2^{d-1}}$$
$$V(K_2) = \frac{h_2 V(B_2)}{d} = \frac{h_1 C(d-1)\rho_1^{d-1}}{d2^{d-1}}.$$

Replacing in 2, the following inequality holds,

$$\rho_2^{d-1}\left(\rho_2 - \frac{h_2}{h_1}\rho_1\right) < \rho_1^{d-1}\left(\rho_2 - \frac{h_2}{h_1}\rho_1\right).$$

Given that $\rho_2^{d-1} > \rho_1^{d-1}$ it must be $\rho_2 < \rho_1 h_2/h_1$. Using Inequality 1, it is, $q < (2d-1)/(d-1)$.

□

For any $d > 1$, let $C(d) = \pi^{d/2}/\Gamma(1+d/2)$, where $\Gamma(\cdot)$ is the Gamma function. We compute the volume of $F \setminus (C \cup K)$ as $V(F) - V(C \cup K)$.

$$V(F) = C(d-1)\int_0^{h_2-h_1}\left(\rho_1/2 + \frac{\rho_2/2 - \rho_1/2}{h_2-h_1}z\right)^{d-1} dz$$
$$= \frac{V(C)}{d}\cdot\frac{q^d-1}{q-1}.$$

$$V(C \cup K) = C(d-1)\Bigg((\rho_1/2)^{d-1}\int_0^{\rho_1(h_2-h_1)/\rho_2} dz + \int_{\rho_1(h_2-h_1)/\rho_2}^{(h_2-h_1)} r_K(z)^{d-1}dz\Bigg)$$
$$= C(d-1)\Bigg((\rho_1/2)^{d-1}\int_0^{\rho_1(h_2-h_1)/\rho_2} dz + \left(\frac{\rho_2/2}{h_2-h_1}\right)^{d-1}\int_{\rho_1(h_2-h_1)/\rho_2}^{(h_2-h_1)} z^{d-1}dz\Bigg)$$
$$= V(C)\frac{1}{q}\left(1 + \frac{1}{d}\left(q^d-1\right)\right).$$

Replacing,

$$V(F \setminus (C \cup K)) = \frac{V(C)}{d}\sum_{i=0}^{d-2}\left(q^i - \frac{1}{q}\right). \tag{3}$$

Using Claim 1 and that $V(C) = dV_d(\rho_1)/(d-1)$ in Equation 3, $V(F \setminus (C \cup K)) \geq \kappa_1(d)V_d(\rho_1)$. Given that $\varepsilon \leq \alpha/e$, we know that $V_d(\rho_1) \geq 1/(\kappa_2(d)n)$, then $V(F \setminus (C \cup K)) \geq \kappa_1(d)/(\kappa_2(d)n)$. Then, the probability that $F \setminus (C \cup K)$ is void of points of P is at least $1-(1-\kappa_1(d)/(\kappa_2(d)n))^n \geq 1 - e^{-\kappa_1(d)/\kappa_2(d)}$. Consider the body $F \setminus (C \cup K)$ evenly divided by $d-1$ planes (of $d-1$ dimensions) containing ℓ. For any of these sections of $F \setminus (C \cup K)$, the probability that it is void of points of $P \setminus \{a\}$, for some $a \in P$, is at least $1-(1-\kappa_1(d)/(\kappa_2(d)n(2d-2)))^{n-1} \geq 1 - e^{-\kappa_1(d)/(2\kappa_2(d)(2d-2))}$, because the intersecting hyperplanes define $2d-2$ sections on $F \setminus (C \cup K)$. Conditioned on the existence of two points $a, b \in P$ located in opposite sections of $F \setminus (C \cup K)$, let S be a spherical cap of base B (of diameter ρ) such that B contains a and b and $S \subset S_2$. Such cap exists as shown before. The probability that S is void of points of P is lower bounded by upper bounding its volume. We know that $V(S) \leq V(S_2)$, and $V(S_2)$ can be upper bounded considering S_1 and $S_2 \setminus S_1$ separately, which we do as follows.

$$V(S_2) - V(S_1) \leq C(d-1)(\rho_2/2)^{d-1}(h_2-h_1)$$
$$\leq C(d-1)\left(\frac{2d-1}{2(d-1)}\rho_1\right)^{d-1}(h_2-h_1)$$
$$= \left(\frac{2d-1}{d-1}\right)^{d-1}\frac{d}{d-1}V_d(\rho_1).$$

Then $V(S) \leq \kappa_2(d)V_d(\rho_1)$. Thus, the probability that S is empty is at least

$$(1-\kappa_2(d)V_d(\rho_1))^{n-2} \geq \exp\left(-\frac{\kappa_2(d)V_d(\rho_1)(n-2)}{1-\kappa_2(d)V_d(\rho_1)}\right).$$

Replacing, we get

$$Pr\left((a,b) \in D(P)\right) \geq \alpha\exp\left(-\frac{\kappa_2(d)V_d(\rho_1)(n-2)}{1-\kappa_2(d)V_d(\rho_1)}\right) = \varepsilon.$$

□

Corollary 1 *For any $n > 1$ and $0 < \varepsilon \leq \alpha/e$, where $\alpha = \left(1-e^{-(2-\sqrt{3})/14}\right)\left(1-e^{-(2-\sqrt{3})/56}\right)$, given the Delaunay graph $D(P)$ of a set P of n points distributed uniformly and independently at random in a unit circle, with probability at least ε, there is an edge $(a,b) \in D(P)$, $a, b \in P$, such that*

$$d(a,b) \geq 2\sqrt[3]{\frac{\ln(\alpha/\varepsilon)}{14\sqrt{\pi}\,(n-2+\ln(\alpha/\varepsilon))}}.$$

Proof. Instantiating Theorem 1 in $d = 2$, we know that with probability at least ε there is an edge $(a,b) \in D(P)$, such that $d(a,b) \geq \rho_1$, where

$$V_2(\rho_1) = \frac{\ln(\alpha/\varepsilon)}{7\,(n-2+\ln(\alpha/\varepsilon))}.$$

We upper bound the area of the circular segment of chord ρ_1 with the area of the rectangle circumscribing it.

$$V_2(\rho_1) \leq \rho_1\left(\frac{1}{\sqrt{\pi}} - \sqrt{\frac{1}{\pi} - \frac{\rho_1^2}{4}}\right).$$

Hence,

$$\sqrt{\frac{\rho_1^2}{\pi} - \frac{\rho_1^4}{4}} \leq \frac{\rho_1}{\sqrt{\pi}} - V_2(\rho_1).$$

Given that $\rho_1/\sqrt{\pi} \geq V_2(\rho_1)$, we can square both sides getting

$$\begin{aligned}\rho_1^4 &\geq 4\left(2\frac{\rho_1}{\sqrt{\pi}} - V_2(\rho_1)\right)V_2(\rho_1)\\ &\geq 4\frac{\rho_1}{\sqrt{\pi}}V_2(\rho_1), \text{ because } V_2(\rho_1) \leq \rho_1/\sqrt{\pi}.\end{aligned}$$

Then we get $\rho_1/2 \geq \sqrt[3]{V_2(\rho_1)/(2\sqrt{\pi})}$ and replacing $V_2(\rho_1)$ the claim follows.

□

Corollary 2 *For any $n > 1$ and $0 < \varepsilon \leq \alpha/e$, where $\alpha = \left(1 - e^{-\kappa_1(3)/\kappa_2(3)}\right)\left(1 - e^{-\kappa_1(3)/(8\kappa_2(3))}\right)$, $\kappa_1(3) = 1/2 - 7/(6\sqrt{8})$, and $\kappa_2(3) = 10 + 3/8$, given the Delaunay graph $D(P)$ of a set P of n points distributed uniformly and independently at random in a unit ball in $\mathbb{R}^3$, with probability at least ε, there is an edge $(a, b) \in D(P)$, $a, b \in P$, such that*

$$d(a,b) \geq \sqrt{2}\sqrt[4]{\frac{\sqrt[3]{48/\pi^4}\ln(\alpha/\varepsilon)}{\kappa_2(3)\left(n - 2 + \ln(\alpha/\varepsilon)\right)}}.$$

Proof. Instantiating Theorem 1 in $d = 3$, we know that with probability at least ε there is an edge $(a, b) \in D(P)$, $a, b \in P$, such that $d(a, b) \geq \rho_1/\sqrt{2}$, where

$$V_3(\rho_1) = \frac{\ln(\alpha/\varepsilon)}{\kappa_2(3)\left(n - 2 + \ln(\alpha/\varepsilon)\right)}.$$

We upper bound the volume of the spherical cap of base diameter ρ_1 with the volume of the cylinder circumscribing it.

$$V_3(\rho_1) \leq \frac{\pi\rho_1^2}{4}\left(\sqrt[3]{\frac{3}{4\pi}} - \sqrt{\left(\frac{3}{4\pi}\right)^{2/3} - \frac{\rho_1^2}{4}}\right).$$

Hence,

$$\sqrt{\left(\frac{\pi}{4}\sqrt[3]{\frac{3}{4\pi}}\right)^2\rho_1^4 - \frac{\pi^2}{64}\rho_1^6} \leq \frac{\pi\rho_1^2}{4}\sqrt[3]{\frac{3}{4\pi}} - V_3(\rho_1).$$

Given that $\pi\rho_1^2/4\sqrt[3]{3/(4\pi)} \geq V_3(\rho_1)$, we can square both sides getting

$$\begin{aligned}\frac{\pi^2}{64}\rho_1^6 &\geq \left(2\frac{\pi\rho_1^2}{4}\sqrt[3]{\frac{3}{4\pi}} - V_3(\rho_1)\right)V_3(\rho_1)\\ &\geq \frac{\pi\rho_1^2}{4}\sqrt[3]{\frac{3}{4\pi}}V_3(\rho_1), \text{ because } V_3(\rho_1) \leq \pi\rho_1^2/4\sqrt[3]{3/(4\pi)}.\end{aligned}$$

Then we get $\rho_1/2 \geq \sqrt[4]{\sqrt[3]{48/\pi^4}V_3(\rho_1)}$ and replacing $V_3(\rho_1)$ the claim follows.

□

References

[1] F. Araújo and L. Rodrigues. Single-step creation of localized Delaunay triangulations. *Wireless Networks*, 15(7):845–858, 2009.

[2] E. M. Arkin, A. Fernández Anta, J. S. B. Mitchell, and M. A. Mosteiro. Probabilistic bounds on the length of a longest edge in delaunay graphs of random points in d-dimensions. In *Proceedings of the 23rd Canadian Conference on Computational Geometry*, pages 163–168, 2011.

[3] C. Avin. Fast and efficient restricted Delaunay triangulation in random geometric graphs. *Internet Mathematics*, 5(3):195–210, 2008.

[4] M. Bern, D. Eppstein, and F. Yao. The expected extremes in a delaunay triangulation. *International Journal of Computational Geometry and Applications*, 1(1):79–91, 1991.

[5] P. Bose, P. Carmi, M. H. M. Smid, and D. Xu. Communication-efficient construction of the plane localized Delaunay graph. In *Proc. of the 9th Latin American Theoretical Informatics Symposium*, pages 282–293, 2010.

[6] P. Bose and P. Morin. Online Routing in Triangulations. In *Proc. of the 10th International Symposium on Algorithms and Computation*, page 113. Springer Verlag, 1999.

[7] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, second edition, 2000.

[8] G. Kozma, Z. Lotker, M. Sharir, and G. Stupp. Geometrically aware communication in random wireless networks. In *Proc. 23rd Ann. ACM Symp. on Principles of Distributed Computing*, pages 310–319, 2004.

[9] E. Kranakis, H. Singh, and J. Urrutia. Compass routing on geometric networks. In *Proc. of the 11th Canadian Conference on Computational Geometry*, 1999.

[10] E. Lebhar and Z. Lotker. Unit disk graph and physical interference model: Putting pieces together. In *Proc. of the 23rd International Symposium on Parallel & Distributed Processing*, pages 1–8. IEEE, 2009.

[11] C. Lemaire and J. Moreau. A probabilistic result on multi-dimensional Delaunay triangulations, and its application to the 2D case. *Comput. Geom.*, 17(1-2):69–96, 2000.

Tighter Bounds on the Size of Optimal Meshes

Donald R. Sheehy

Abstract

The theory of optimal size meshes gives a method for analyzing the output size of a Delaunay refinement mesh in terms of the integral of a sizing function over the space. The input points define a maximal such sizing function called the feature size. Integrating the feature size function over input domain is not easy, and historically, was not deemed necessary. Matching upper and lower bounds in terms of this integral seemed sufficient. However, a new analysis of the feature size integral [9] led to linear-size Delaunay meshes [9], the Scaffold Theorem relating surface and volume meshes [8], and a time-optimal output-sensitive point meshing algorithm [10]. The key idea is to consider the pacing of an ordered point set, a measure of the rate of change in the feature size as points are added one at a time. In previous work, Miller et al. showed that if an ordered point set has pacing ϕ, then the number of vertices in an optimal mesh will be $O(\phi^d n)$, where d is the input dimension. We give a new analysis of this integral showing that the output size is only $O(n \log \phi)$. The new analysis tightens all of the previous results mentioned above and provides matching lower bounds.

1 Introduction

Delaunay refinement is the adding of new vertices to an initial starting set, so that the Delaunay simplices all have a bounded circumradius-to-shortest-edge ratio (see Figure 1). A celebrated result of Ruppert [13] showed how to construct such meshes with an optimal number of vertices. The straightforward generalization of Ruppert's work to $\mathbb{R}^d$ says that the number of vertices in an optimal mesh of a domain $\Omega \subset \mathbb{R}^d$ starting with a point set P is bounded by the **feature size integral**:

$$\Theta\left(\int_\Omega \frac{dx}{\mathbf{f}_P(x)^d}.\right)$$

The function $\mathbf{f}_P : \mathbb{R}_d \to \mathbb{R}_{\geq 0}$ is the **feature size** induced by the input set P and is defined as

$$\mathbf{f}_P(x) = \min\{r : |P \cap \mathbf{ball}(x, r)| \geq 2\}.$$

Clearly, $\mathbf{f}_P$ is 1-Lipschitz. Here and throughout, the asymptotic bounds suppress constants that are singly exponential in d.

Such high dimensional meshes are ideal for geometric and topological inference as they provide a nice basis for a space of smooth functions graded according to the density of the input points [7, 14]. Such functions, such as the distance function to the input or the distance to the empirical measure, can be used to recover the homology of the underlying space from which the input was sampled [3, 2].

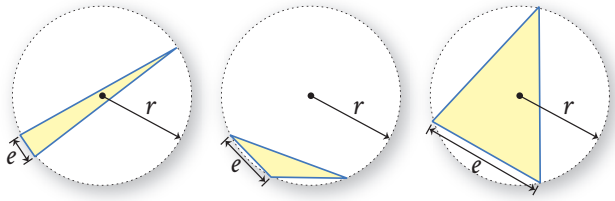


Figure 1: Three examples of triangles with different circumradius to shortest edge ratios. Delaunay refinement attempts to produce a mesh in which all simplices have this ratio bounded by a constant.

The tight bounds on mesh sizing from the Ruppert bounds are nice, but they are not very informative because they depend on the feature size integral. They do not, for example, tell when the output size will be $O(n^2)$ or $O(n)$ or any other tidy function of n. Moreover, they do not give an easy way to evaluate the amortized change in the output size as a result of adding a single new point.

In this paper, we give a new analysis of the feature size integral that provides tight upper and lower bounds. Our analysis makes it clear exactly when an input will yield an optimal mesh of $O(n)$ size. Moreover, the analysis, gives a tight bound on the influence of a single new point, which may have implications for future algorithms, particularly in dynamic meshing.

If we order the input set $P = \{p_1, \ldots, p_n\}$, we can define the ith **prefix** of the ordering to be $P_i = \{p_1, \ldots, p_i\}$. For any point p_i in the ordering ($i \geq 3$), the **pacing** is defined as the ratio ϕ_i of the feature sizes at p_i induced by P_{i-1} and P_i:

$$\phi_i = \frac{\mathbf{f}_{P_{i-1}(p_i)}}{\mathbf{f}_{P_i(p_i)}}.$$

Let $\phi_P = \max_i \phi_i$ denote the **pacing of the ordering**.

In previous work, it was shown that the feature size integral is at most $O(n\phi_P^d)$ [9]. Theorem 1 (be-

This is an extended abstract of a presentation given at EuroCG 2012. 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 in a conference with formal proceedings and/or in a journal.

low) implies a bound of $O(n \log \phi_P)$, eliminating the exponential dependence on d. This tightens the previous results using this method, moving them from theoretically novel to practically useful.

We need a few definitions in order to state the main result. We will use $|\cdot|$ to denote the Euclidean norm for points and cardinality for sets. Let $\mathbf{ball}(c, r)$ denote the closed Euclidean ball of radius r centered at c. Let $\mathbb{V}_d$ denote the volume of the unit ball $\mathcal{B} = \mathbf{ball}(0, 1)$, so $d\mathbb{V}_d$ is the $(d-1)$-dimensional volume of the sphere bounding $\mathcal{B}$.

Theorem 1 *Let $P = \{p_1, \ldots, p_n\}$ be a set of n points in $\mathbb{R}^d$ such that p_1 and p_2 are the farthest pair. Let Ω be a subset of $\mathbb{R}^d$ such that*

$$\mathbf{ball}(p_1, 2|p_1 - p_2|) \subseteq \Omega \subseteq \mathbf{ball}(p_1, c|p_1 - p_2|),$$

for some constant $c \geq 2$. Then,

$$\int_\Omega \frac{dx}{\mathbf{f}_P(x)^d} = \mathbb{V}_d \Theta\left(n + \sum_{i=3}^{n} \log \phi_i\right).$$

The proof will be broken up into two parts: the upper bound in Theorem 4 and the lower bound in Theorem 6.[1]

2 Related Work

The spread Δ_P of a point set P is the ratio of the largest to smallest interpoint distances. A standard bad case for meshing illustrated on the left in Figure 2 occurs when there is high spread and a large empty annulus around the close points. On the right of Figure 2, the point set has been refined. Approximately a constant number of points appear in each of the geometrically growing annuli. Thus the number of points added is roughly the log of the ratio of the inner and outer radius. Observe that other than the inner most annuli, the number of points added is not heavily affected by the number of points on the inside. As a result, previous methods to characterize the feature size locally such as the gap ratio introduced by Talmor [15] can dramatically overestimate the feature size integral because each interior point pays for all of the refinement. In the worst case, such an analyses lead to an $O(n \log \Delta_P)$ upper bound when the true answer is $\Theta(n)$.

Erickson also used the spread to bound the complexity of Delaunay triangulations in the absence of refinement [5, 6]. However, even point sets with exponentially large spread can yield linear size meshes. Figure 3 illustrates a point set with geometrically growing spread, but the pacing is small, thus the mesh size will be linear.

[1] The asymptotic version of the lower bound as stated in Theorem 1 also depends on the trivial $\Omega(n)$ lower bound from the Ruppert bounds.

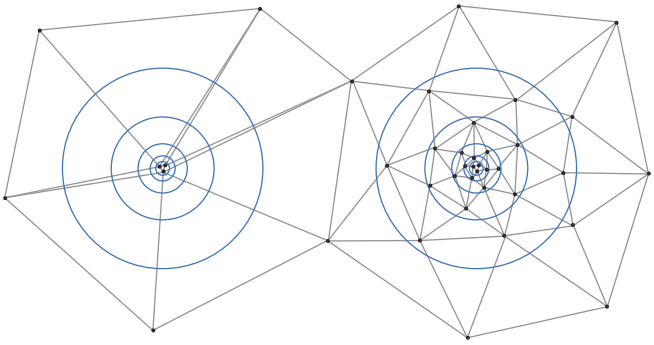


Figure 2: On the left is a typical bad example for meshing, a small set of points inside a large empty annulus. On the right shows a similar example after refinement.

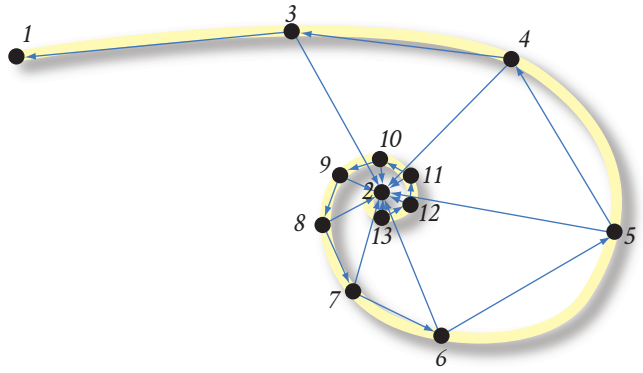


Figure 3: Point sets with bounded pacing can still have high spread, i.e. the ratio of largest to smallest interpoint distances can be $2^{O(n)}$. In the figure, each point has arrows directed at its two nearest predecessors.

The use of feature size integrals can also be used in the anisotropic setting [1]. Pav also gave anisotropic bounds on mesh size in terms of the smallest angles allowed[12].

There is a strong connection between the present work and the linear cost of *balancing* a quadtree [11]. The corners of quadtree may be viewed as a well-paced set of points (with pacing = 1). The balancing of the quadtree refines it, adding only a linear number of points.

3 Upper Bound

The proof of the upper bound on the feature size integral will follow a simple pattern. First, we prove a bound for inputs consisting of only two points (Lemma 2). Then, we bound the change in the integral upon adding a single new point (Lemma 3). Finally, we apply this Lemma inductively to get the final bound (Theorem 4).

Lemma 2 (Just two points) *If $P = \{p, q\}$ and $\Omega \subseteq \mathbf{ball}(p, c|p-q|)$ for some constant $c > 2$, then*

$$\int_\Omega \frac{dx}{\mathbf{f}_P(x)^d} \leq \mathbb{V}_d(1 + d\ln(2c)).$$

Proof. For all $x \in \mathbb{R}^d$, $\mathbf{f}_P(x) \geq \max\{\frac{1}{2}|p-q|, |x-p|\}$. So, we can rewrite the integral in polar coordinates (centered at p) and bound it as follows.

$$\begin{aligned}\int_\Omega \frac{dx}{\mathbf{f}_P(x)^d} &\leq d\mathbb{V}_d \left(\int_0^{\frac{1}{2}|p-q|} \frac{r^{d-1}dr}{(\frac{1}{2}|p-q|)^d} + \int_{\frac{1}{2}|p-q|}^{c|p-q|} \frac{r^{d-1}dr}{r^d} \right) \\ &\leq d\mathbb{V}_d \left(\frac{1}{d} + \ln\left(\frac{2c|p-q|}{|p-q|} \right) \right) \\ &= \mathbb{V}_d(1 + d\ln(2c)).\end{aligned}$$

□

We now bound the change in the feature size integral induced by the addition of a single point.

Lemma 3 (One more point upper bound) *Let P be a point set and let $P' = P \cup \{q\}$. If $\phi = \frac{\mathbf{f}_P(q)}{\mathbf{f}_{P'}(q)}$ then*

$$\int_\Omega \left(\frac{1}{\mathbf{f}_{P'}(x)^d} - \frac{1}{\mathbf{f}_P(x)^d} \right) dx \leq \mathbb{V}_d \left(1 + d\ln(3d\phi)\right).$$

Proof. Let U be the subset of $\mathbb{R}^d$ where $\mathbf{f}_P \neq \mathbf{f}_{P'}$. Clearly, the integral is 0 outside U, so we can restrict our attention to U. Let $R = \mathbf{f}_{P'}(q)$; this is the distance from q to the nearest point of P. This implies that $R\phi = \mathbf{f}_P(q)$. For all points x in the ball $B = \mathbf{ball}(q, \frac{R}{2})$, $\mathbf{f}_{P'}(x) \geq \frac{R}{2}$, so

$$\int_B \left(\frac{1}{\mathbf{f}_{P'}(x)^d} - \frac{1}{\mathbf{f}_P(x)^d} \right) dx \leq \int_B \left(\frac{2}{R} \right)^d dx = \mathbb{V}_d.$$

The definitions imply the following bounds for any $x \in U$:

$$\begin{aligned}\mathbf{f}_P(x) &\leq |x-q| + R\phi, \\ \mathbf{f}_{P'}(x) &\geq |x-q|.\end{aligned}$$

The upper bound follows because $\mathbf{f}_P$ is 1-Lipschitz. The lower bound follows because q must be one of the two nearest neighbors of x if $\mathbf{f}_P(x) \neq \mathbf{f}_{P'}(x)$. We apply these bounds as follows.

$$\begin{aligned}&\int_{U\setminus B} \left(\frac{1}{\mathbf{f}_{P'}(x)^d} - \frac{1}{\mathbf{f}_P(x)^d} \right) dx \\ &\quad\leq \int_{U\setminus B} \left(\frac{1}{|x-q|^d} - \frac{1}{(|x-q| + R\phi)^d} \right) dx \\ &\quad\leq d\mathbb{V}_d \int_{R/2}^\infty \left(\frac{1}{r^d} - \frac{1}{(r+R\phi)^d} \right) r^{d-1} dr \\ &\quad< d\mathbb{V}_d \ln(3d\phi).\end{aligned}$$

We have extended the integral over all of $\mathbb{R}^d \setminus B$ (the function is nonnegative) and rewrote it in polar coordinates. The final inequality follows from a straightforward calculus exercise (the full proof may be found in Lemma 7 below). To bound the integral over all of Ω, we simply add the bounds on the integral over B and $U \setminus B$. □

Theorem 4 (Upper bound) *Let $P = \{p_1 \ldots, p_n\}$ be an ordered set of points such that $|p_1 - p_2| = \mathbf{diameter}(P)$. Let $\Omega \subseteq \mathbf{ball}(p_1, c\mathbf{diameter}(P))$ for some constant $c > 1$ be the bounding region. Then,*

$$\int_\Omega \frac{dx}{\mathbf{f}_P(x)^d} < \mathbb{V}_d \left(1 + d\ln(2c) + \sum_{i=3}^n (1 + d\ln(3d\phi_i)) \right).$$

Proof. We rewrite the integral as a telescoping sum:

$$\int_\Omega \frac{dx}{\mathbf{f}_P(x)^d} = \int_\Omega \frac{dx}{\mathbf{f}_{P_2}(x)^d} + \sum_{i=3}^n \left(\int_\Omega \frac{dx}{\mathbf{f}_{P_i}(x)^d} - \int_\Omega \frac{dx}{\mathbf{f}_{P_{i-1}}(x)^d} \right).$$

The bounds from Lemmas 2 and 3 complete the proof. □

4 Lower Bound

The proof of the lower bound will be similar to the proof of the upper bound in that we will use the pacing of a single new point to bound the change in the feature size integral.

Lemma 5 (One more point lower bound) *Let P be a set of at least 2 points and let $P' = P \cup \{q\}$ for some $q \in \mathbb{R}^d$. Let $\Omega \subset \mathbb{R}^d$ be a set containing $\mathbf{ball}(q, \mathbf{diameter}(P'))$. If $\phi = \frac{\mathbf{f}_P(q)}{\mathbf{f}_{P'}(q)}$ then*

$$\int_\Omega \left(\frac{1}{\mathbf{f}_{P'}(x)^d} - \frac{1}{\mathbf{f}_P(x)^d} \right) dx \geq \frac{\mathbb{V}_d}{2^d} \left(d\ln\frac{\phi}{3} - 1 \right)$$

Proof. The bound is trivial if $\phi \leq 3$, so we may assume that $\phi > 3$. Let $R = \mathbf{f}_{P'}(q)$. Since $\mathbf{f}_P \geq \mathbf{f}_{P'}$, it will suffice to prove a lower bound on the change in the feature size integral over the subset $U = \{x : R \leq |x-q| \leq \frac{R\phi}{3}\} \subseteq \Omega$. For all $x \in U$,

$$\begin{aligned}\mathbf{f}_P(x) &\geq \frac{2R\phi}{3}, \text{ and} \\ \mathbf{f}_{P'}(x) &\leq R + |x-q| \leq 2|x-q|.\end{aligned}$$

The lower bound follows because there is at most one point of P in the interior of $\mathbf{ball}(q, R\phi)$. The upper bound follows because $\mathbf{f}_{P'}(x)$ is 1-Lipschitz. We apply

these bounds and convert to polar coordinates:

$$\begin{aligned}
&\int_U \left(\frac{1}{\mathbf{f}_{P'}(x)^d} - \frac{1}{\mathbf{f}_P(x)^d} \right) dx \\
&\geq \int_U \left(\frac{1}{(2|x-q|)^d} - \left(\frac{3}{2R\phi} \right)^d \right) dx \\
&> \left(d\mathbb{V}_d \int_R^{\frac{R\phi}{3}} \frac{r^{d-1}}{(2r)^d} dr \right) - \frac{\mathbb{V}_d}{2^d} \\
&= \frac{\mathbb{V}_d}{2^d} \left(d \ln \frac{\phi}{3} - 1 \right)
\end{aligned}$$

□

When we apply the preceding lemma to a set of n points, we get the following lower bound.

Theorem 6 (Lower bound) *Let $P = \{p_1 \ldots, p_n\}$ be an ordered set of points. If $\Omega \subset \mathbb{R}^d$ is a set containing* $\mathbf{ball}(p, 2\mathbf{diameter}(P))$ *for some $p \in P$, then*

$$\int_\Omega \frac{dx}{\mathbf{f}_P(x)^d} > \frac{\mathbb{V}_d}{2^d} \sum_{i=3}^{n} \left(d \ln \frac{\phi_i}{3} - 1 \right).$$

5 Some directions for future work

The work of Ruppert on optimal meshing has been extended to feature size functions that also take into account input features beyond just point sets, including piecewise linear or even piecewise smooth complexes [4]. One direction for future work is to extend these methods for bounding the feature size to these settings as well.

It would also be interesting to extend this approach to anisotropic case, such as in [1]. In that setting, it is not known how to relax the quality constraints to guarantee a linear size mesh.

Moreover, since Theorem 1 describes the cost of adding a single point, it makes sense to apply these analytic techniques to dynamic meshing problems.

References

[1] Jean-Daniel Boissonnat, Camille Wormser, and Mariette Yvinec. Anisotropic Delaunay Mesh Generation. Rapport de recherche RR-7712, INRIA, August 2011.

[2] Frédéric Chazal, David Cohen-Steiner, and Quentin Mérigot. Geometric inference for probability measures. *Foundations of Computational Mathematics*, 11:733–751, 2011.

[3] Frédéric Chazal and André Lieutier. Topology guaranteeing manifold reconstruction using distance function to noisy data. In *Proceedings of the 22nd ACM Symposium on Computational Geometry*, 2006.

[4] Tamal K. Dey and Joshua A. Levine. Delaunay meshing of piecewise smooth complexes without expensive predicates. *Algorithms*, 2(4):1327–1349, 2009.

[5] Jeff Erickson. Nice point sets can have nasty Delaunay triangulations. In *Proceedings of the 17th ACM Symposium on Computational Geometry*, pages 96–105, 2001.

[6] Jeff Erickson. Dense point sets have sparse delaunay triangulations or "... but not too nasty". *Discrete & Computational Geometry*, 33:83–115, 2005.

[7] Benoît Hudson, Gary L. Miller, Steve Y. Oudot, and Donald R. Sheehy. Topological inference via meshing. In *Proceedings of the 26th ACM Symposium on Computational Geometry*, pages 277–286, 2010.

[8] Benoît Hudson, Gary L. Miller, Todd Phillips, and Donald R. Sheehy. Size complexity of volume meshes vs. surface meshes. In *SODA: ACM-SIAM Symposium on Discrete Algorithms*, 2009.

[9] Gary L. Miller, Todd Phillips, and Donald R. Sheehy. Linear-size meshes. In *CCCG: Canadian Conference in Computational Geometry*, pages 175–178, 2008.

[10] Gary L. Miller, Todd Phillips, and Donald R. Sheehy. Beating the spread: Time-optimal point meshing. In *SOCG: Proceedings of the 26th ACM Symposium on Computational Geometry*, pages 321–330, 2011.

[11] Doug Moore. The cost of balancing generalized quadtrees. In *SMA '95: Proceedings of the Third Symposium on Solid Modeling and Applications*, pages 305–312, 1995.

[12] Steven E. Pav. An anisotropic cardinality bound for triangulations. In *CCCG: The Canadian Conference on Computational Geometry*, pages 95–98, 2004.

[13] Jim Ruppert. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *J. Algorithms*, 18(3):548–585, 1995.

[14] Donald R. Sheehy. *Mesh Generation and Geometric Persistent Homology.* PhD thesis, Carnegie Mellon University, 2011.

[15] Dafna Talmor. *Well-Spaced Points for Numerical Methods.* PhD thesis, Carnegie Mellon University, 1997.

A A bit of calculus

Lemma 7 *Given positive constants $\phi \geq 1$ and R,*

$$\int_{R/2}^{\infty} \left(\frac{1}{r^d} - \frac{1}{(r+R\phi)^d} \right) r^{d-1} dr < \ln(3d\phi).$$

Proof. We bound this integral using the change of variables $u = \frac{R\phi}{r} + 1$ as follows.

$$\begin{aligned}
\int_{R/2}^{\infty} \left(\frac{1}{r^d} - \frac{1}{(r+R\phi)^d} \right) r^{d-1} dr &= \int_1^{1+2\phi} \left(\frac{u^d - 1}{u^d(u-1)} \right) du \\
&= \sum_{i=0}^{d-1} \int_1^{1+2\phi} u^{i-d} du \\
&< \ln(1+2\phi) + \sum_{i=0}^{d-2} \frac{1}{d-i-1} \\
&< \ln(1+2\phi) + \ln d \\
&\leq \ln(3d\phi). \qquad \square
\end{aligned}$$

Augmentability to Cubic Graphs

Alexander Pilz*

Abstract

We consider the problem of adding edges to a given graph in order to make it cubic (i.e., all vertices have degree 3). As a first result, we show that the problem is NP-complete if the resulting graph is required to be simple and planar. In the reduction, we rely on edges whose combinatorial embedding is not determined by the graph. Further, we show that the problem of deciding whether additional edges can be drawn on a given (embedded) plane geometric graph, in a way that the resulting plane geometric graph is cubic, is NP-complete. The problem remains NP-complete even if the initial graph is connected.

1 Introduction

Given a graph $G = (V, E)$, a second graph $G' = (V, E \cup E')$ is called an *(edge) augmentation* of G [2]. Classical augmentation problems involve, e.g., improving the resulting graph's connectivity, minimizing the number of edges added. A recent survey, focusing on augmentation of plane geometric (i.e., straight-line) graphs, is given by Hurtado and Tóth [2].

An extremal variant of augmentation problems on geometric graphs is the one where the initial edge set is empty. García et al. [1] give a partial characterization of point sets in the plane that allow drawing a plane geometric cubic graph (in a cubic graph, all vertices have degree 3). Schmidt and Valtr [5] extend this characterization and give an $O(n^3)$ time algorithm to compute a cubic graph on a given point set, if there exists one.

In this work, we address the problem of adding edges in order to make a graph cubic, both in the general planar and the plane geometric setting. Of course, to obtain a cubic graph, the number of edges that have to be added is already predefined by the initial graph. We ask whether such an augmentation is possible at all, either when requiring the resulting graph to be planar or when a predefined straight-line embedding of the initial graph is given.

Augmentability in plane geometric graphs is constrained by the visibility in the embedding. Deciding augmentability to a planar cubic graph is a different problem since there are no visibility constraints and the combinatorial embedding (i.e., the order of the edges around a vertex, or, equivalently, the incidence of the edges to faces) might not be determined.

In Section 2, the problem of augmenting a graph to a planar cubic graph is shown to be NP-complete. In that setting, the combinatorial embedding of the initial graph is not given. In Section 3, we show that the problem of deciding whether additional edges can be drawn on a given (embedded) plane geometric graph, in a way that the resulting graph is plane geometric and cubic, is NP-complete. The problem remains NP-complete even if the initial graph is connected.

Definition 1 *A vertex of degree 3 is called* full.

2 The General Setting

In this section, we prove NP-completeness of deciding whether a given planar graph can be augmented to a planar cubic graph. The problem is obviously in NP. We show a reduction from Restricted Planar 3-Sat.

Definition 2 (Restricted Planar 3-Sat [3]) *Given a Boolean formula ϕ in Conjunctive Normal Form let $G(\phi)$ be the graph whose vertices are the variables and the clauses of ϕ, and whose edges are defined by the occurrence of a variable in a clause. Given that $G(\phi)$ is planar, that every variable occurs at least twice and at most three times (negated or unnegated), and that every clause contains at least two and at most three literals, the* Restricted Planar 3-Sat *problem asks whether ϕ is satisfiable.*

We use the standard approach of representing the graph defining the problem instance by another graph constructed using gadgets (however, we do not directly use gadgets representing the edges of $G(\phi)$). Note throughout the construction that we rely on the fact that the combinatorial embedding of the initial graph is not fixed. (It is an open problem to decide augmentability for input graphs having a fixed combinatorial embedding.) A valid augmentation of the graph will, however, be 3-connected and therefore have a fixed combinatorial embedding [6]. The different possibilities to embed the initial graph are well-defined by the gadgets. Contrary to similar reductions, the crucial decision for a possible algorithm

*Recipient of a DOC-fellowship of the Austrian Academy of Sciences at the Institute for Software Technology, Graz University of Technology, Austria. apilz@ist.tugraz.at. Part of this work was done while the author was visiting the Work Group Theoretical Computer Science at the Institute of Computer Science, Freie Universität Berlin, Germany.

This is an extended abstract of a presentation given at EuroCG 2012. 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 in a conference with formal proceedings and/or in a journal.

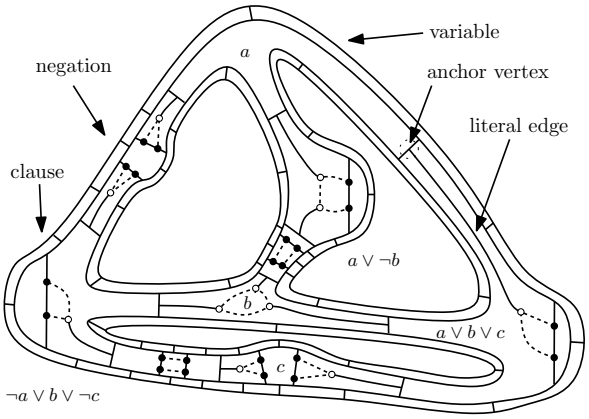


Figure 1: A formula embedded as a graph using the gadgets. An augmentation is, e.g., $a = T, b = c = F$.

is not where to place the additional edges, but how to embed the graph to allow an augmentation. An example for a construction is shown in Figure 1. Full vertices are depicted as the common point of segments only. White and black dots indicate vertices of degree 1 and 2 of the initial graph, respectively. We start with a plane embedding of $G(\phi)$ and draw a new graph G_1 along with $G(\phi)$ by "thickening" the drawing of $G(\phi)$ to a connected bounded region with holes, s.t. it contains the embedding of $G(\phi)$. G_1 is used as "jig" for the whole construction. We separate the variables from the clauses by drawing edges, one for unnegated and two for negated literals. This replaces each face and each node of $G(\phi)$ by a face in G_1, and adds one face for each negation. Now, each *variable face* is neighbored to *clause faces*, possibly via a *negation face*; these incidences reflect which variables occur in which clauses. We continue by replacing the faces of G_1 by gadgets defined later. We give the construction in terms of variants of the graph's embedding. Each edge separating a clause face and a variable face in G_1 is subdivided by a so-called *anchor vertex*. Each anchor vertex is connected to a degree 1 vertex by a *literal edge*. The combinatorial embedding of G_2 is fixed except for literal edges, which can be drawn either in the clause face or the variable face (of G_1). A valid augmentation to a cubic graph (if there exists one) determines the radial order of the neighbors of each anchor vertex. Our construction uses literal edges to transport the truth value of a variable. From a planar cubic augmentation of the graph, one can directly derive the truth assignment of the variables satisfying the formula. Otherwise, such an augmentation is not possible.

A clause gadget (see Figure 2) consists of a face that is incident to the two or three anchors of the corresponding literal edges. A literal is true if its edge is drawn inside the clause face. Two neighbored vertices v and w of degree 2 are also incident to the clause

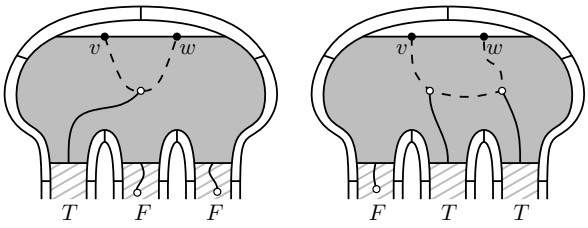


Figure 2: The gadget for a clause and two possible embeddings with augmentation (dashed). The clause face is gray.

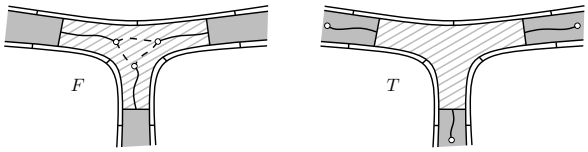


Figure 3: The gadget for a variable with three occurances. The variable face is striped.

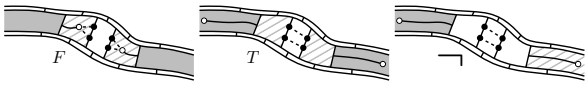


Figure 4: The gadget for a variable occurring twice. The same construction is used to negate a literal.

face. If no literal edge is inside the clause face, there exists no augmentation for the face. Hence, at least one literal has to be set to true. A path between v and w through all end vertices of literal edges inside the clause gives a locally cubic graph. Note that the construction drawn on the outside of the clause face prevents a combinatorially different embedding of the gadget.

For variables of ϕ we distinguish between variables that occur three times (see Figure 3) and twice (see Figure 4). A variable with three occurrences is represented by a variable face incident to the three anchors of its three literal edges. The variable is set to false if all literal edges are inside that face, and true if none of them is. If all literal edges are in the face, a circle through their non-full end vertices makes them full. If only one or two literal edges are inside that face, there is obviously no valid simple augmentation. The construction is different for two-literal variables. Either both the literal edges have to be drawn in a clause face, or they are in the two other faces incident to the anchor vertices. There are two pairs of neighbored vertices of degree 2. Each pair is incident to the face where one could also draw the literal edge, and the two pairs are incident to a common third face. If both literal edges are in their corresponding clause face, two edges in the third face can make the vertices full. If they are not in the clause face, two paths through the literal edges' non-full vertices allow a locally cubic graph. If only one of the literal edges would not be

in the clause face, there exists no valid augmentation. Finally, note that the gadget for two variables shown in Figure 4 can also be used to negate the Boolean value transported by a literal edge if it is placed between a variable and a clause face. This completes the reduction.

Theorem 1 *The problem of deciding whether a graph can be augmented to a simple planar cubic graph is NP-complete.*

3 The Geometric Setting

Requiring the added edges to be drawn as line segments on a given straight-line embedding of the initial graph imposes further constraints to the problem when the resulting drawing should be crossing-free.

We reduce Positive Planar 1-in-3-Sat, which has been proven to be NP-complete in [4], to the problem of augmenting to a plane geometric cubic graph.

Definition 3 (Positive Planar 1-in-3-Sat [4]) *Given a Boolean formula ϕ in Conjunctive Normal Form let $G(\phi)$ be the graph whose vertices are the variables and the clauses of ϕ, and whose edges are defined by the occurrence of a variable in a clause. Given that $G(\phi)$ is planar, that every variable occurs only positively (i.e., each literal in every clause is unnegated), and that every clause has exactly three variables, the* Positive Planar 1-in-3-Sat *problem asks whether there exists a variable assignment such that exactly one variable in each clause is true.*

Mulzer and Rote [4] use this problem to show NP-hardness of the Minimum Weight Triangulation problem. Their reduction from Planar 3-Sat to Positive Planar 1-in-3-Sat maintains the property that $G(\phi)$ can be embedded having all variable vertices on a line, and all edges can be embedded in a rectilinear fashion (i.e., using orthogonal segments and 90° bends). We use the standard technique of replacing such an embedding of $G(\phi)$ by geometric gadgets. We show that the resulting geometric graph can be augmented to a plane geometric cubic graph if and only if the Positive Planar 1-in-3-Sat instance modeled is 1-in-3-satisfiable.

There are three types of gadgets: *wires* that transport the truth assignment of a Boolean variable, *splitters* that replace the variable node in $G(\phi)$ and make sure that the truth assignment of a variable is the same for all its instances, and *clauses* that can be drawn if and only if exactly one of the three literals of the corresponding CNF-clause is true. For all these gadgets, we need a construction element called *pin* to reduce visibility among the vertices and to make vertices full without further effects. Although the gadgets are drawn with many collinear points, none of the constructions depend on collinearity and therefore the result is also applicable for graphs with vertices embedded as a point set in general position.

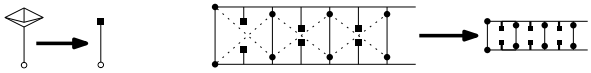


Figure 5: Two basic gadgets and their schematic representation. The pin gadget to the left has exactly one vertex of degree 1, the other vertices are full. The wire gadget to the right transports the truth assignment of a variable by the different directions of the diagonals. The dotted strokes indicate these.

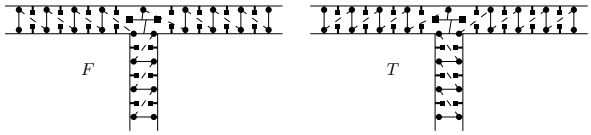


Figure 6: Splitters replace variable nodes and ensure that all the wires of a variable carry the same truth assignment to the different clauses.

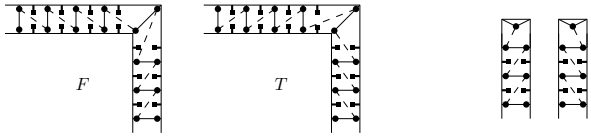


Figure 7: Bends of the edges of the original graph can be modeled as shown to the left. If a variable occurs only once, it can be represented by the gadget to the right.

A pin gadget is shown in Figure 5 (left). This subgraph has exactly one vertex of degree 1, all others are full. In further illustrations, this gadget will be indicated by a line segment and a box. The wire gadget, as shown in Figure 5 (right), transports the state of a variable. In each face of the wire gadget, we can only draw a single diagonal, making full two of the four vertices that have degree 2. Therefore, the only possible augmentation of the graph is by diagonals with a common orientation along a wire. This leaves one of the vertices at the end of each wire non-full. Which of these two vertices is full and which is not after drawing the wire diagonals will be used in combination with the clause gadgets.

A splitter allows having multiple wires carrying the same truth assignment, see Figure 6. It therefore is a representation of a variable, where the corresponding wire ends represent the literals of the variable at the clauses. Several splitters can be used one after the other. The pin construction where the three wires meet reduces the visibility between the non-full vertices and only allows the depicted augmentations.

Bends are constructed in a straight-forward manner (see Figure 7, left). They work the same way as wires.

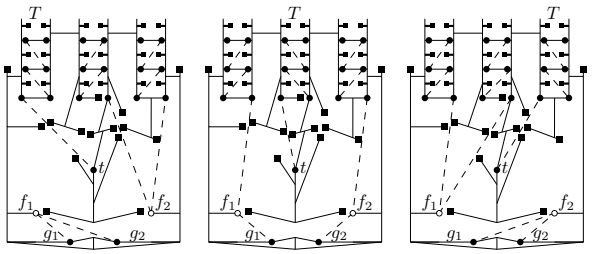


Figure 8: Clause gadgets and their three possible augmentations (dashed segments). Note the special construction of the end part of the middle wire gadget which allows the graph to be connected.

If a literal occurs only once in the formula, we do not need any bends or splitters. The variable can have any truth assignment, we end the wire of the literal with the gadget shown in Figure 7 (right).

Clauses (see Figure 8) are entered by the ends of three wire gadgets. Note the small modification of the end of the middle wire; it allows reducing the visibility by pins while keeping the graph connected. In each clause gadget, there is one designated vertex t of degree 2 that sees exactly three non-full vertices at wire ends. Any of these three vertices being full would determine the diagonal orientation in the corresponding wire. The edge added to t therefore indicates which variable is set to true. The non-full vertices f_1, f_2, g_1, and g_2 allow the construction to be completed if and only if the other two wires carry false.

In Figure 8 (left), the first wire is set to true. We have to draw edges from f_1 to the only two non-full vertices visible to it, i.e., g_1 and g_2, and therefore f_2 becomes a neighbor of the other two wire end vertices, indicating false for these.

In Figure 8 (center), setting the second wire to true leaves f_1 with three visible non-full points. One edge has to go to the end vertex of the first wire as otherwise there is one non-full vertex in that wire. Subsequently, g_1 is the only choice for the second edge since an edge f_1g_2 would keep g_1 from getting full, the edges for f_2 directly follow.

In Figure 8 (right), the third wire is set to true, leaving g_1 and g_2 as the only possible new neighbors for f_2. Since they are now full, f_1 has to set the first and second wire to false.

All gadgets only use a constant number of edges per element of $G(\phi)$ (the wires can be "stretched"). Again, from a plane geometric cubic augmentation of the construction, one can directly derive the truth assignment for ϕ.

Theorem 2 *The problem of deciding whether a plane geometric graph can be augmented to a plane geometric cubic graph is NP-complete. The problem remains NP-complete if the initial graph is connected.*

4 Conclusion

In this work we considered two variants of the problem of adding edges to given graphs in order to make them cubic. We showed that the problem is NP-complete if the resulting graph is required to be planar. In the context of geometric graphs, we showed NP-completeness of the problem in which the resulting graph is required to be a plane geometric (straight-line) graph, with the embedding of the vertices as a point set in the plane given beforehand.

In the non-geometric setting, we relied on the literal edges to transport the truth value of a variable. Our reduction does not work for fixed face-edge incidence. Of course, all these incidences are given for the initial graph in the geometric setting. There, the reduction relies on the constraint that edges need to be straight line segments. It would be interesting whether the problem remains NP-complete if this constraint is dropped. In that setting, only the requirement of simplicity constrains the position of the edges inside a face of the initial graph.

Open Problem 1 *What is the complexity of deciding augmentability of a planar graph having a determined combinatorial embedding?*

Acknowledgements

The author wants to express his gratitude to Jens Schmidt for valuable discussions on the problem setting, as well as to Oswin Aichholzer and Thomas Hackl for helpful suggestions.

References

[1] A. García, F. Hurtado, C. Huemer, J. Tejel, and P. Valtr. On triconnected and cubic plane graphs on given point sets. *Comput. Geom.*, 42(9):913–922, 2009.

[2] F. Hurtado and C. D. Tóth. Plane geometric graph augmentation: a generic perspective. In J. Pach, editor, *Thirty Essays on Geometric Graph Theory*, volume 29 of *Algorithms Combin.* Springer, 2012. To appear.

[3] K. Jansen. One strike against the min-max degree triangulation problem. *Comput. Geom.*, 3(2):107–120, 1993.

[4] W. Mulzer and G. Rote. Minimum-weight triangulation is NP-hard. *J. ACM*, 55(2):1–29, 2008.

[5] J. M. Schmidt and P. Valtr. Cubic plane graphs on a given point set. Submitted.

[6] H. Whitney. Congruent graphs and the connectivity of graphs. *Amer. J. Math.*, 54(1):150–168, 1932.

Book Embedding of N-free posets

Anna Beata Kwiatkowska* Maciej M. Sysło†

Abstract

In this paper we deal with the page number problem for partially ordered sets (posets) restricted to the family of N-free posets. We calculate this number exactly for tree-structured N-free posets and for tree-structured N-free planar posets. Some other results are also mentioned. For details, see [10].

1 Introduction

The book embedding of graphs was introduced by Bernhart and Kainen in [3] and the study of such graph embeddings has been stimulated by the conjecture, that the page number is unbounded in the family of planar graphs.

In some applications, the book embedding problem for graphs reduces to that for covering digraphs of posets over the set of all their linear extensions. The research in this direction has been initiated by Nowakowski [14], then continued independently in [15] and in [20] (see also [12], [13], and [9]).

A book embedding of a planar poset can be regarded as the upward planar drawing of the covering digraph of the poset, studied in computational geometry, see e.g. [4], [5].

In this paper we investigate the book embedding problem for N-free posets. Our approach is based on characterizations of N-free posets as those whose covering digraphs are line digraphs. Using graph-theoretic results we attempt to find which posets admit 2-page book embedding and what is the page number of some planar posets.

This paper is based on the Ph.D. Thesis of the first author [10].

2 The book embedding of graphs

A *book embedding of a graph* $G = (V, E)$ consists of two assignments: the vertices of G to the spine of a book in some order and the edges of G to the pages (infinite half-planes) of the book so that each edge lies on only one page and the edges assigned to the same page do not intersect. The objective is to minimize the number of pages used. The *page number of a graph* G, denoted by $p(G)$, is the smallest number k such that G has a book embedding on k pages over all possible permutations of the vertex set V on the spine. It is known [3] that:

- $p(G) = 1$ *if and only if* G *is outerplanar,*
- $p(G) \leq 2$ *if and only if* G *is a subgraph of a planar Hamiltonian graph*; hence deciding whether a (planar) graph G has page number 2 is an NP-complete problem.

Bernhart and Kainen conjectured that the page number of planar graphs is unbouded, however Yannakakis [21] proved that $p(G) \leq 4$ for every planar graph G.

3 The book embedding of posets

Let $(P, \leq)$ be a partially ordered set, simply called a *poset* and denoted by P. Let $H(P)$ denote the *covering digraph* of P, known also as the *Hasse diagram* of P and let $G(P)$ denote the undirected copy of $H(P)$, known as the covering graph of P

A *book embedding of a poset* P consists also of two assignments: the elements of P to the spine of the book according to a linear extension L of P and the arcs of $H(P)$ to the pages of the book so that each arc lies on only one page and the arcs assigned to the same page do not intersect. The objective is to minimize the number of pages used. Let $p(P, L)$ denote the smallest number k such that poset P has a book embedding on k pages with the elements of P placed on the spine according to a linear extension L of P. The *page number of a poset* P, denoted by $p(P)$, is the minumum $p(P, L)$ over all linear extesions L of P.

The page number for posets has been introduced by Nowakowski [14]. Then Nowakowski and Parker [15] have developed some bounds for the page number in general and also for planar posets. Syslo [20] has provided some other bounds, in particular investigating the relation between the page number and the jump number of a poset (which is the minimum number of breaks between consecutive chains in a linear extension of a poset). It is known ([15], [20]) that:

- $p(P) = 1$ *if and only if* $G(P)$ *has no cycle.*

No characterization of two page posets is known. Planar postes which require more than two pages can

*Faculty of Mathematics and Computer Science, Nicolaus Copernicus University, Chopin 12-18, 87-100 Toruń, Poland, aba@mat.uni.torun.pl

†Faculty of Mathematics and Computer Science, Nicolaus Copernicus University, Chopin 12-18, 87-100 Toruń, Poland, syslo@ii.uni.wroc.pl

be found in [15] and [20]. It was conjectured that the page number is unbounded for planar posets. One such conjecture has been disproved by Le Tu [12] who has also constructed a planar poset which requires four pages. Another such conjecture appeared in [1].

If $p(P) = 2$ then it is clear that the covering digraph of P contains a cycle. In [9] we study a family of posets which are combinations of edge-disjoint cycles and classify such posets according to their page number which is either 2 or 3.

4 N-free posets

An N is a poset consisting of four elements a, b, c, d such that $a < c$, $b < c$, $b < d$, $b \not< a$, $a \not< d$, $d \not< c$, see Figure 1. This poset plays an important role in studying several algorithmic problems on posets [16], see also [19].

A poset P is *N-free* if its covering digraph $H(P)$ contains no subdigrah isomorphic to $H(N)$, the covering digraph of N. N-free posets have been introduced by Grillet [6], see also [11].

Figure 1: An N poset.

An N-free poset P (in general, an N-free digraph) can be also defined as a line digraph. The *line digraph of a multidigraph* E is the digraph $L(E)$ having a vertex $l(a)$ for each arc a of E and an arc $(l(a_1), l(a_2))$ for each pair of arcs a_1 and a_2 in E which are of the form $a_1 = (u, v)$ and $a_2 = (v, w)$. A digraph D is a *line digraph* if there exists a multidigraph E such that $D = L(E)$; E is called a *root digraph* of D.

There exist many characterizations of N-free posets (see [7], [10]). We make use here mainly of property 4 below.

Theorem 1 *Let $D = (P, A)$ denote the covering digraph of a poset. The following statements are equivalent:*

1. *P has the C. A. C. property, i.e. each maximal chain meets each maximal antichain, [6], [11].*
2. *P is N-free*
3. *D is a line digraph.*
4. *There exist two improper partitions $\{P_i\}_I$ and $\{Q_i\}_I$ of P (improper means that some P_i and Q_i may be empty) such that $A = \bigcup_{i \in I} P_i \times Q_i$, where $\times$ denotes the Cartesian product of sets.*
5. *For every two elements $p, q \in P$, if $N(p) \cap N(q) \neq \emptyset$ then $N(p) = N(q)$, where $N(p)$ denotes the set of all elements of P which cover p in P.*

A linear time algorithm to recognize a line digraph and output its root digraph has been presented in [17]. Therefore, N-free posets can be also recognized in time linear in the number of poset elements and the number of poset comparabilities.

Let us assume that for a given N-free poset P, represented by its covering digraph $D = (P, A)$, we have two improper partitions $\{P_i\}_I$ and $\{Q_i\}_I$ of P such that $A = \bigcup_{i \in I} P_i \times Q_i$. If both sets P_i and Q_i are nonempty then $P_i \times Q_i$ is a complete bipartite subdigraph of D. Therefore the arc set A of D can be partitioned into complete bipartite subdigraphs $B_i = P_i \times Q_i$ ($i \in I$) of D – we call P_i *a lower set* of B_i and Q_i *an upper set* of B_i. For two subdigraphs B_i and B_j, $i \neq j$, they lower sets are disjoint and also they upper sets are disjoint, however they lower and upper sets may have some elements in common. We have:

Theorem 2 *Let D be a line digraph and let E denote its root digraph. We have the following properties of D and E:*

1. *A digraph D has no loops if and only if its root digraph E has no loops.*
2. *If E is acyclic (i.e. has no directed cycles) and E has no loops, then the digraph D is acyclic, contains no loops and has no transitive arcs.*
3. *If E has no multiple (parallel) arcs then*

$$| Q_i \cap P_j | \leq 1$$

for each pair i and j, such that $i \neq j$.

Therefore, if a poset P is N-free, then the root digraph of the covering digraph of P contains no loops and no cycles. Moreover, in what follows we assume that it contains also no multiple arcs, therefore $| Q_i \cap P_j | \leq 1$ for each pair i and j, such that $i \neq j$.

5 Page number of N-free posets

We present here some results on the page number problem for N-free posets and for N-free planar posets. Te details are included in [10].

5.1 Arbitrary N-free posets

Let P be a poset and (p_i, q_i), $i = 1, 2, ..., k$ be a set of k arcs in its covering digraph $H(P)$. If $p_1 < p_2 < ... < p_k < q_1 < q_2 < ... < q_k$ in a linear extension L of P, then evidently we have $k \leq p(P, L)$. Hence:

Theorem 3 (Syslo [20]) *For a complete bipartite poset $P_{k,l}$ with k elements in its lower set and l elements in its upper set, we have $p(P_{k,l}) = min\{k, l\}$.*

We generalize this observation to a tree-like combination of complete bipartite posets. An N-free poset P has a *tree-like structure*, if for every element $p \in P$, which is neither minimal nor maximal in P, the digrapf $H(P) - \{p\}$ is not connected. Equivalently, an N-free poset P has a tree-like structure if a root digraph of its covering digraph is a tree. We have the following theorem:

Theorem 4 *If a poset P is N-free and has a tree-like structure, then we have*

$$p(P) = max_{i \in I} min\{\mid P_i \mid, \mid Q_i \mid\}$$

.

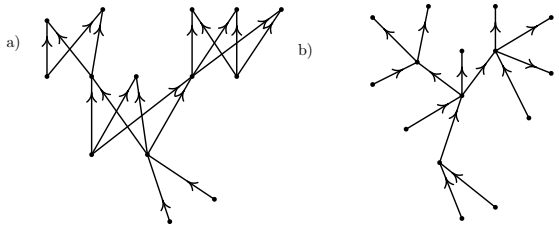


Figure 2: a) The covering digraph of N-free tree-like structured poset and b) its root digraph

The proof of this theorem is constructive – an efficient recursive algorithm finds a book embedding for a poset P which is N-free and has a tree-like structure with the optimal number of pages.

For an arbitrary N-free poset P we provide an upper bound to its page number. It follows from the fact that to destroy all cycles in the root digraph T of the covering digraph $H(P)$ of P we have to remove $c(T)$ arc from T, where $c(T)$ denotes the *cyclomatic number* of T, i.e. the number of independent cycles in T. The cyclomatic number of a connected digraph is equal to the number of its arcs minus the number of its vertices plus 1.

Theorem 5 *Let P be an N-free poset and T denotes the root digraph of its covering digraphs. Then we have:*

$$p(P) \leq max_{i \in I} min\{\mid P_i \mid, \mid Q_i \mid\} + m(T) - n(T) + 1,$$

where $m(T)$ is the number of arcs in T and $n(T)$ is the number of vertices in T. Equivalently

$$p(P) \leq max_{i \in I} min\{\mid P_i \mid, \mid Q_i \mid\} + \mid P \mid - \mid I \mid + 1.$$

In general this bound can not be improved and on the other hand it can be arbitrarily bad.

When a poset P is not N-free, we can transform P to another poset P' which is N-free and the relations among the original elements of P are preserved. Such a transformation may depend on subdividing some arcs in the covering digraph of P – this transformation resuults in topological embedding of P. Another such transformation – *a general subdivision* – has been proposed in [18]. Once a poset P is transformed to an N-free poset, we can apply Theorem 5. However this result is only of algorithmic nature since there is no bound to the number of subdivisions or general subdivisions which transform a poset to an N-free poset with the same comparabilities among elements of P.

5.2 Planar N-free posets

A poset is *planar* if its covering digraph is *upward planar*, i.e. it admits an upward planar drawing. The study of upward planarity has been very active in the last years from both, theoretical (see [8]) and algorithmic point of view (see [4], [5]).

Testing planarity of ordered sets is equivalent to testing upward planarity of digraphs since a digraph D is upward planar if and only if the ordered set with covering digraph D' obtained from D by sudividing all its arcs, is planar. Since the digraph D' is N-free and the problem of testing upward planarity of digraphs is NP-complete, we obtain the following conclusion:

Corollary 6 *Planarity testing of N-free posets is NP-complete.*

It is interesting to observe that the line digraph of an upward planar digraph may not be planar but for an upward planar line digraph there exists a root digraph which is planar, see [10] for details.

The first result on the page number problem for N-free posets appeared in [2] (a *covering four-cycle* is isomorphic to 2×2 bipartite complete digraph).

Theorem 7 (Alzohairi [2]) *The page number of an N-free planar poset which contains no covering four-cycle is at most two.*

It follows from another result in [2] that the page number of an N-free planar lattice is at most two.

Now we look at the structure of a planar N-free poset using its covering digraph $H(P)$ as a line digraph. By Theorem 1 the set of arcs of $H(P)$ can be decomposed into complete bipartite subdiraphs $\{B_i\}_{i \in I}$. Since $H(P)$ is planar, each $B_i = (P_i, Q_i)$ satisfies one of the conditions: $s = 1$ or $t = 1$ or $s = 2$ and $t \geq 2$ or $s \geq 2$ and $t = 2$, where $s = \mid P_i \mid$ and $t = \mid Q_i \mid$, see Figure 3. We denote these bipartite subdigraphs by $K_{1,t}$, $K_{s,1}$, $K_{2,t}$, $K_{s,2}$, respectively.

Notice that in Theorem 7, subdigraphs $K_{2,t}$ and $K_{s,2}$ are not allowed since they contain a four-cycle.

Taking into account all possible emeddings of these complete bipartite subdigraphs we obtain the following theorem.

Theorem 8 *If $H(P)$ is the covering digraph of an N-free planar poset P then each of its complete bipartite subdigraphs has one of the embeddings shown in Figure 3, where the circles attached to vertices correspond to complete bipartite subdigraphs and the filled circles correspond to complete bipartite subdigraphs attached at a cut vertex.*

Finally we get the following result:

Theorem 9 *If in the covering digraph $H(P)$ of an N-free planar poset P each vertex which belongs to two complete bipartite subdigraphs is a cut vertex, then*

$$p(P) \leq 2$$

and the corresponding emebedding of P into two pages can be found in time linear in the number of verices and the number of arcs in $H(P)$.

It is still an open question whether N-free planar posets need more than two pages in a book embedding.

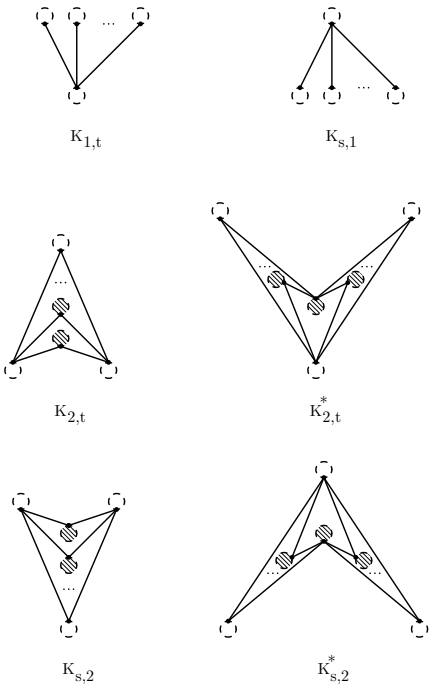


Figure 3: Plane configurations of complete bipartite subdigraphs in the covering digraph of an N-free planar poset (* denotes different plane embeddings)

References

[1] Alzohairi M., *The Pagenumber of Ordered Sets*, Ph. D. Thesis, University of Ottawa, Ottawa 1996.

[2] Alzohairi M., N-free planar ordered sets that contain no covering four-cycle have page number two, *Arab. J. Sci. Eng.* **31**(2006), 1-8.

[3] Bernhart F., Kainen P.C., The book thickness of a graph, *J. Combin. Theory* **B 27**(1979), 320-321.

[4] Garg A., Tamassia R., Upward planarity testing, *Order* **12**(1995), 109-133.

[5] Giordano F., Liotta G., Mchedlidze T., Symvonis A., Computing upward topological book embeddings of upward digraphs, in: *ISAAC 2007*, LNinCS vol. **4835**, Springer-Verlag 2007, 172-183.

[6] Grillet P.A., Maximal chains and antichains, *Fund. Math.* **65**(1969), 157-167.

[7] Hemminger R.L., Beineke L.W., Line graphs and line digraphs, in: Beineke L.W., Wilson R.J. (eds.), *Selected Topics in Graph Theory*, Academic Press. London 1978, 271-305.

[8] Kelly D., Rival I., Fundamentals of planar ordered sets, *Discrete Math.*, **63**(1987), 197-216.

[9] Kwiatkowska A.B., Syslo M.M., On Posets of Page Number 2, *Electronic Notes In Discrete Mathemtics* **22**(2005), 549-552.

[10] Kwiatkowska A.B., On Page Number of Ordered Sets, Faculty of Mathematics and Computer Science, Nicolaus Copernicus University, Toruń, in preparation.

[11] Leclerc B., Monjardet B., Orders C. A. C., *Fund. Math.* **79**(1973), 11-22.

[12] Le Tu Quoc Hung, A planar poset which requires four pages, *Ars Combin.* **35**(1993), 291-302.

[13] Le Tu Quoc Hung, *Numbering problems in graphs and in posets*, Ph.D. Thesis, Institute of Computer Science, University of Wroclaw, 1994.

[14] Nowakowski R., Problems 3.7 and 3.8, in: I. Rival (ed.), *Algorithms and Order*, Kluver 1989, 478.

[15] Nowakowski R., Parker A., Ordered sets, pagenumbers and planarity, *Order* **6**(1989), 209-218.

[16] Rival I., Stories about order and the letter N, in: *Combinatorics and Odered Sets*, Contemp. Math. 57, AMS, 1986, 263-285.

[17] Syslo M.M., A labeling algorithm to recognize a line digraph and output its root graph, *Information Processing Letters* **15**(1982), 28-30.

[18] Syslo M.M., Optimal constructions of reversible digraphs, *Discrete Applied Math.* **7**(1984), 209-220.

[19] Syslo M.M., Minimizing the jump number for partially ordered sets: a graph-theoretic approach, *Order* **1**(1984), 7-19.

[20] Syslo M.M., Bounds to the page number of partially ordered sets, in: M. Nagel (ed.), *Graph-Theoretic Concepts in Computer Science*, LNCS **411**(1990), Springer-Verlag, Berlin, Heidelberg, 181-195.

[21] Yannakakis M., Embedding planar graphs in four pages, *J. Comp. Syst. Sci.* **38**(1989) 36-67.

A Linear Time Algorithm for the Queue-Numbers of Maximal Outerplanar Graphs *

Toru Hasunuma † Ayane Haruna ‡

Abstract

We present a linear time algorithm for computing the queue-numbers of maximal outerplanar graphs.

1 Introduction

A *k-queue layout* of a graph $G = (V, E)$ consists of a vertex-ordering $\sigma : V(G) \to \{1, 2, \ldots, |V(G)|\}$ and an assignment $\rho : E(G) \to \{1, 2, \ldots, k\}$ of the edges to k queues such that no two edges in the same queue nest, i.e., for each $i \in \{1, 2, \ldots, k\}$ and any two edges $uv, xy \in \rho^{-1}(i)$, it does not hold that $\sigma(u) < \sigma(x) < \sigma(y) < \sigma(v)$. The minimum number of queues for a queue layout of G is the *queue-number* $\mathsf{qn}(G)$ of G. A graph is *k-queue graph* if $\mathsf{qn}(G) \leq k$. The notion of queue layouts was originally introduced by Heath, Leighton, and Rosenberg [4, 5]. Queue layouts of graphs can be applied to fault-tolerant computing [7] and three-dimensional graph drawing [2, 3].

The challenging open problem on queue layouts is the conjecture posed by Heath et al. [4, 5] that every planar graph can be laid out using $O(1)$ queues. Recently, Di Battista, Frati, and Pach [1] proved that every planar graph can be laid out using $O(\log^4 n)$ queues, which improves the previously known bound of $O(\sqrt{n})$. The problem of laying out planar graphs using a constant number of queues still remains open, while if we restrict ourselves to outerplanar graphs, such a problem has been solved, i.e., Heath et al. [4] proved that every outerplanar graph can be laid out using two queues. Heath and Rosenberg [5] characterized a 1-queue graph as an arched leveled planar graph, and also proved that the problem of recognizing a 1-queue graph is NP-complete. The planarity can be efficiently tested. Therefore, the problem of computing the queue-numbers of planar graphs is NP-hard. In this paper, we show that if we restrict ourselves to maximal outerplanar graphs, such a problem can be solved in linear time. That is, we present a linear time algorithm for computing the queue-numbers of maximal outerplanar graphs.

*This research was supported by JSPS KAKENHI 21500017.

†Institute of Socio-Arts and Sciences, The University of Tokushima, hasunuma@ias.tokushima-u.ac.jp

‡Faculty of Integrated Arts and Sciences, The University of Tokushima

2 Preliminaries

Let $G = (V, E)$ be a graph. For $S \subseteq V(G)$, the subgraph of G induced by S is denoted by $\langle S \rangle_G$. For $v \in V(G)$, we denote by $deg_G(v)$ the degree of v in G. Also, let $\Delta(G) = \max_{v \in V(G)} deg_G(v)$.

An *outerplanar graph* is a planar graph such that every vertex can be placed on the boundary of the unbounded face called the *outer face*. A maximal outerplanar graph is an outerplanar graph to which we cannot add a new edge while preserving the outerplanarity, i.e., every face except for the outer face is a triangle. An edge on the boundary of the outer face is a *boundary edge*, while an edge which is not a boundary edge is an *internal edge*. The set of boundary edges in a maximal outerplanar graph induces a Hamilton cycle called the *outer cycle*. We denote by $C(G)$ the outer cycle of a maximal outerplanar graph G. A *caterpillar* is a tree T such that the graph obtained from T by deleting all the leaves is a path.

For a vertex-ordering σ of G and $u, v \in V(G)$, we may write $u <_\sigma v$ instead of $\sigma(u) < \sigma(v)$. Let σ and σ' be two vertex-orderings of G. If there exists an automorphism ϕ of G such that $\sigma(v) = \sigma'(\phi(v))$ for all $v \in V(G)$, then σ and σ' are isomorphic and we write $\sigma \cong \sigma'$.

3 Queue-Numbers of Fan Graphs

Definition 1 *A fan graph F_n, where $n \geq 4$, is a maximal outerplanar graph with n vertices such that there exists a vertex with degree $n-1$. The vertex with degree $n-1$ in F_n is called the center vertex of F_n.*

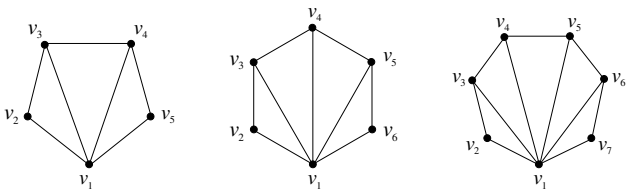


Figure 1: F_5, F_6 and F_7.

Figure 1 illustrates F_5, F_6 and F_7. In Lemma 1, we assume that $V(F_n) = \{v_i \mid 1 \leq i \leq n\}$ and $E(F_n) = \{v_1 v_i \mid 2 \leq i \leq n\} \cup \{v_i v_{i+1} \mid 2 \leq i < n\}$.

Definition 2 *Let $C(G) = (v_1, v_2, \ldots, v_n)$ be the outer cycle of a maximal outerplanar graph G. A vertex-ordering σ of G is a zig-zag ordering if there exists $i \in \{1, 2, \ldots, n\}$ such that for each $1 \leq t \leq n$,*

$$\sigma^{-1}(t) = \begin{cases} i+k-1 & \text{if } t = 2k-1, \\ i-k & \text{if } t = 2k \text{ and } i > k, \\ i-k+n & \text{if } t = 2k \text{ and } i \leq k, \end{cases}$$

or for each $1 \leq t \leq n$,

$$\sigma^{-1}(t) = \begin{cases} i+k & \text{if } t = 2k, \\ i-k & \text{if } t = 2k+1 \text{ and } i > k, \\ i-k+n & \text{if } t = 2k+1 \text{ and } i \leq k. \end{cases}$$

A zig-zag ordering in the first (resp. second) case is called the $[i, i-1]$-zig-zag ordering (resp. $[i, i+1]$-zig-zag ordering) and is denoted by $\sigma_{[i,i-1]}$ (resp. $\sigma_{[i,i+1]}$).

Lemma 1

$$\mathsf{qn}(F_n) = \begin{cases} 1 & \text{if } 4 \leq n \leq 7, \\ 2 & \text{if } n \geq 8. \end{cases}$$

Moreover, any vertex-ordering of a 1-queue layout of F_7, F_6, and F_5 is isomorphic to $\sigma_{[3,2]}$, one of $\sigma_{[3,2]}$ and $\sigma_{[2,3]}$, one of $\sigma_{[3,2]}$, $\sigma_{[2,3]}$, and $\sigma_{[2,1]}$, respectively.

Proof. It can be easily checked that $\mathsf{qn}(F_4) = 1$. Suppose that $n \geq 5$. Let σ be a vertex ordering of F_n such that $\sigma(v_1) = i$. Let $V_L = \{\sigma^{-1}(j) \mid 1 \leq j < i\}$ and $V_R = \{\sigma^{-1}(j) \mid i < j \leq n\}$. Now assume that $\mathsf{qn}(F_n) = 1$. Let $E_{\sigma,n}$ be the set of edges uv, where $u, v \in V(F_n)$, such that uv and $v_1 v_i$ do not nest for $2 \leq i \leq n$. Then, it can be shown that $E_{\sigma,n}$ induces the graph obtained from a star S_L with $|V_L|$ vertices and a star S_R with $|V_R|$ vertices by adding an edge joining a leaf of S_L and a leaf of S_R. Since $F_n - v_1$ is a path, it follows that $|V_L| \leq 3$ and $|V_R| \leq 3$. This indicates that $n \leq 7$.

Suppose that $5 \leq n \leq 7$. In this case, we can define σ so that $|V_L| \leq 3$ and $|V_R| \leq 3$ and obtain a 1-queue layout of F_n. Here, $E_{\sigma,n}$ induces a path with $n-1$ vertices. (For example, see Figure 2 in the case of $n = 7$.) By this property, vertex-orderings for 1-queue layouts of F_n are restricted to zig-zag orderings as follows.

- Case 1: $n = 7$. Since $\sigma(v_1) = 4$, σ is one of the following two vertex-ordering:
  - $v_3 <_\sigma v_2 <_\sigma v_4 <_\sigma v_1 <_\sigma v_5 <_\sigma v_7 <_\sigma v_6$.
  - $v_6 <_\sigma v_7 <_\sigma v_5 <_\sigma v_1 <_\sigma v_4 <_\sigma v_2 <_\sigma v_3$.
- Case 2: $n = 6$. Since $3 \leq \sigma(v_1) \leq 4$, σ is one of the following four vertex-ordering:
  - $v_3 <_\sigma v_2 <_\sigma v_4 <_\sigma v_1 <_\sigma v_5 <_\sigma v_6$.
  - $v_2 <_\sigma v_3 <_\sigma v_1 <_\sigma v_4 <_\sigma v_6 <_\sigma v_5$.

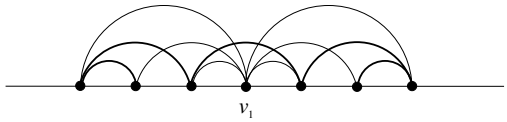


Figure 2: A 1-queue layout of F_7.

  - $v_5 <_\sigma v_6 <_\sigma v_4 <_\sigma v_1 <_\sigma v_3 <_\sigma v_2$.
  - $v_6 <_\sigma v_5 <_\sigma v_1 <_\sigma v_4 <_\sigma v_2 <_\sigma v_3$.
- Case 3: $n = 5$. Since $2 \leq \sigma(v_1) \leq 4$, σ is one of the following six vertex-orderings:
  - $v_3 <_\sigma v_2 <_\sigma v_4 <_\sigma v_1 <_\sigma v_5$.
  - $v_2 <_\sigma v_3 <_\sigma v_1 <_\sigma v_4 <_\sigma v_5$.
  - $v_2 <_\sigma v_1 <_\sigma v_3 <_\sigma v_5 <_\sigma v_4$.
  - $v_4 <_\sigma v_5 <_\sigma v_3 <_\sigma v_1 <_\sigma v_2$.
  - $v_5 <_\sigma v_4 <_\sigma v_1 <_\sigma v_3 <_\sigma v_2$.
  - $v_5 <_\sigma v_1 <_\sigma v_4 <_\sigma v_2 <_\sigma v_3$.

□

Let H_6 be the graph shown in Figure 3. Note that the graph obtained from H_6 by deleting any vertex with degree two is isomorphic to F_5.

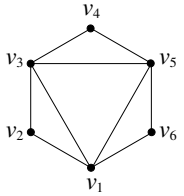


Figure 3: H_6.

The following fact has been shown in [6]. Here we present a proof of this fact which is simpler than that in [6].

Lemma 2 $\mathsf{qn}(H_6) = 2$.

Proof. Assume that $\mathsf{qn}(H_6) = 1$. Since $H_6 - v_4$ and $H_6 - v_6$ are isomorphic to F_5, from Case 3 in the proof Lemma 1, any vertex-ordering of H_6 must be one of the following four vertex-orderings:

- $v_2 <_\sigma v_3 <_\sigma v_1 <_\sigma v_4 <_\sigma v_5 <_\sigma v_6$.
- $v_2 <_\sigma v_1 <_\sigma v_3 <_\sigma v_6 <_\sigma v_5 <_\sigma v_4$.
- $v_4 <_\sigma v_5 <_\sigma v_6 <_\sigma v_3 <_\sigma v_1 <_\sigma v_2$.
- $v_6 <_\sigma v_5 <_\sigma v_4 <_\sigma v_1 <_\sigma v_3 <_\sigma v_2$.

However, each vertex-ordering contradicts all the six possible zig-zag vertex-orderings for 1-queue layout of $H_6 - v_2$. □

From Lemmas 1 and 2, the following useful observation for our algorithm is obtained.

Lemma 3 *Let $T(G)$ be the graph obtained from a maximal outerplanar graph G by deleting all the boundary edges and all the vertices with degree two. If $T(G)$ is not a caterpillar with $\Delta(T(G)) \leq 4$, then $\mathsf{qn}(G) = 2$.*

Proof. If there is a triangle whose all the three edges are internal, i.e., $T(G)$ is not a tree, then G contains H_6 as its subgraph and $\mathsf{qn}(G) = 2$ from Lemma 2.

Suppose that any triangle contains at least one boundary edge. Consider the graph D obtained from the dual graph of G by deleting the vertex corresponding to the outer face. If a triangle in G has two internal edges (resp., one internal edge), then the corresponding vertex in D has degree two (resp., one). Thus, D is a path and it can be shown that $T(G)$ is a caterpillar. Since G contains $F_{\Delta(T(G))+3}$, from Lemma 1, if $\Delta(T(G)) \geq 5$, then $\mathsf{qn}(G) = 2$. □

4 Algorithm

Suppose that $T(G)$ is a caterpillar with $\Delta(T(G)) \leq 4$. Then, G can be represented by a union of k fan graphs $F_{n_1}, F_{n_2}, \ldots, F_{n_k}$, where $5 \leq n_i \leq 7$, $1 \leq i \leq k$, such that two consecutive fan graphs overlap so that their intersection graph is the fan graph with 4 vertices. That is, $G = F_{n_1} \cup F_{n_2} \cup \cdots \cup F_{n_k}$ such that $F_{n_i} \cap F_{n_{i+1}} = F_4$ for $1 \leq i < k$. Figure 4 illustrates an example of such a maximal outerplanar graph with $k = 9$, $n_1 = 6$, and $n_2 = 5$.

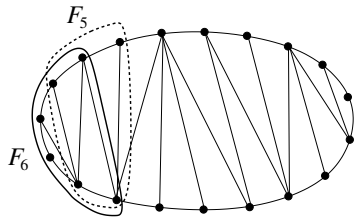


Figure 4: $G = F_6 \cup F_5 \cup F_6 \cup F_7 \cup F_5 \cup F_5 \cup F_6 \cup F_6 \cup F_5$.

Let σ^* be the vertex-ordering of a 1-queue layout of G. Let σ_i^* be the restricted vertex-ordering for F_{n_i}, $1 \leq i \leq k$. Then, each σ_i^* is isomorphic to one of the listed zig-zag orderings in Lemma 1. Since two consecutive fan graphs F_{n_i} and $F_{n_{i+1}}$ overlap at four vertices, σ_i^* and σ_{i+1}^* are related and we can show that σ_{i+1}^* is uniquely (in the sense that two isomorphic vertex-orderings are consider the same) determined by σ_i^*. For example, see Figure 5 which illustrates the case where $n_i = 7$ and $\sigma_i^* \cong \sigma_{[3,2]}$. The vertices v_1, v_5, v_7, v_6 are overlapped by the next fan-graph $F_{d_{i+1}}$ such that the vertex v_6 is the center vertex of $F_{d_{i+1}}$. By the ordering of these vertices, σ_{i+1}^* must be isomorphic to $\sigma_{[3,2]}$. Similarly, we can show Conditions 5–9 in the following theorem. Besides, Conditions 1–3 follow from Lemma 1. Conversely, if there exists a vertex-ordering which satisfies all such conditions, then G can be laid out using one queue.

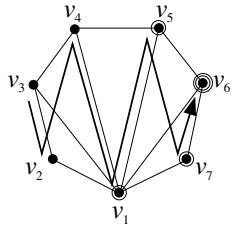


Figure 5: The case where $n_i = 7$ and $\sigma_i^* \cong \sigma_{[3,2]}$.

Theorem 4 *Let G be a maximal outerplanar graph. Let $G = F_{n_1} \cup F_{n_2} \cup \cdots \cup F_{n_k}$ such that $5 \leq d_i \leq 7$ for $1 \leq i \leq k$ and $F_{n_i} \cap F_{n_{i+1}} = F_4$ for $1 \leq i < k$. Then, $\mathsf{qn}(G) = 1$ if and only if there exists a vertex-ordering σ^* so that for all $1 \leq i < k$, the following nine conditions hold:*

1. *If $n_i = 7$, then $\sigma_i^* \cong \sigma_{[3,2]}$.*
2. *If $n_i = 6$, then $\sigma_i^* \cong \sigma \in \{\sigma_{[3,2]}, \sigma_{[2,3]}\}$.*
3. *If $n_i = 5$, then $\sigma_i^* \cong \sigma \in \{\sigma_{[3,2]}, \sigma_{[2,3]}, \sigma_{[2,1]}\}$.*
4. *If $n_i = 7$ and $\sigma_i^* \cong \sigma_{[3,2]}$, then $\sigma_{i+1}^* \cong \sigma_{[3,2]}$.*
5. *If $n_i = 6$ and $\sigma_i^* \cong \sigma_{[3,2]}$, then $\sigma_{i+1}^* \cong \sigma_{[2,3]}$.*
6. *If $n_i = 6$ and $\sigma_i^* \cong \sigma_{[2,3]}$, then $\sigma_{i+1}^* \cong \sigma_{[3,2]}$.*
7. *If $n_i = 5$ and $\sigma_i^* \cong \sigma_{[3,2]}$, then $\sigma_{i+1}^* \cong \sigma_{[2,1]}$.*
8. *If $n_i = 5$ and $\sigma_i^* \cong \sigma_{[2,3]}$, then $\sigma_{i+1}^* \cong \sigma_{[2,3]}$.*
9. *If $n_i = 5$ and $\sigma_i^* \cong \sigma_{[2,1]}$, then $\sigma_{i+1}^* \cong \sigma_{[3,2]}$.*

Conditions 4–9 in Theorem 4 can be represented by a finite automaton with six states (without setting of an initial state), which can be minimized to three states (see Figure 6). Note that the notation in each state indicates the pair(s) of n_i and the zig-zag type of σ_i^*.

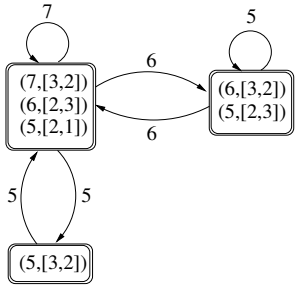


Figure 6: A finite automaton for Conditions 4–9.

Based on Lemma 3 and Theorem 4, we can describe an algorithm for deciding whether the queue-number of a maximal outerplanar graph is one or not as Algorithm Queue-Number with Procedure Queue-Number-Check. Queue-Number-Check corresponds to the finite automaton shown in Figure 6. Note that the procedure uses the degree sequence

Algorithm 1 QUEUE-NUMBER

Require: A maximal outerplanar graph G.
Ensure: The queue-number $\mathsf{qn}(G)$ of G.

Let T be the graph obtained from G by deleting all the boundary edges and all the vertices with degree two.
if T is not a caterpillar with $\Delta(T) \leq 4$ **then**
  $\mathsf{qn}(G) := 2$.
else
  Let $P = (v_1, v_2, \ldots, v_k)$ be the path obtained from T by deleting all the leaves of T.
  Let $(d_1, d_2, \ldots, d_k)$ be the degree sequence of P, where $d_i = deg_T(v_i)$ for $1 \leq i \leq k$.
  if $d_1 = 4$ **then**
    QUEUE-NUMBER-CHECK(1)
  else if $d_1 = 3$ **then**
    QUEUE-NUMBER-CHECK(1)
    if $\mathsf{qncheck} = 2$ **then**
      QUEUE-NUMBER-CHECK(2)
    end if
  else if $d_1 = 2$ **then**
    QUEUE-NUMBER-CHECK(1)
    if $\mathsf{qncheck} = 2$ **then**
      QUEUE-NUMBER-CHECK(2)
    end if
    if $\mathsf{qncheck} = 2$ **then**
      QUEUE-NUMBER-CHECK(3)
    end if
  end if
  $\mathsf{qn}(G) := \mathsf{qncheck}$.
end if

instead of the sequence of fan graphs, i.e., $d_i = n_i - 3$ for each $1 \leq i \leq k$. QUEUE-NUMBER first checks whether $T(G)$ is a caterpillar with $\Delta(T(G)) \leq 4$, and then calls QUEUE-NUMBER-CHECK at most three times since there are at most three possible initial states in the automaton shown in Figure 6. It can be checked in linear time whether $T(G)$ is a tree (which is indeed a caterpillar) with $\Delta(T(G)) \leq 4$. Besides, QUEUE-NUMBER-CHECK clearly runs in linear time. Therefore, we have the following theorem.

Theorem 5 *The problem of computing the queue-numbers of maximal outerplanar graphs can be solved in linear time.*

5 Conclusion

In this paper, we have presented a linear time algorithm for computing the queue-numbers of maximal outerplanar graphs. It remains unknown whether there exists a polynomial time algorithm for computing the queue-numbers of (non-maximal) outerplanar graphs. Besides, it would be interesting to investigate such a problem for k-trees.

Procedure 2 QUEUE-NUMBER-CHECK(state)

$\mathsf{qncheck} := 1$.
$i := 2$.
while $\mathsf{qncheck} = 1$ and $i \leq k$ **do**
  if $\mathsf{state} = 1$ **then**
    if $d_i = 4$ **then**
      $\mathsf{state} := 1$.
    else if $d_i = 3$ **then**
      $\mathsf{state} := 2$.
    else if $d_i = 2$ **then**
      $\mathsf{state} := 3$.
    end if
  else if $\mathsf{state} = 2$ **then**
    if $d_i = 4$ **then**
      $\mathsf{qncheck} := 2$.
    else if $d_i = 3$ **then**
      $\mathsf{state} := 1$.
    else if $d_i = 2$ **then**
      $\mathsf{state} := 2$.
    end if
  else if $\mathsf{state} = 3$ **then**
    if $d_i = 4$ or $d_i = 3$ **then**
      $\mathsf{qncheck} := 2$.
    else if $d_i = 2$ **then**
      $\mathsf{state} := 1$.
    end if
  end if
  $i := i + 1$.
end while

References

[1] G. Di Battista, F. Frati, and J. Pach. On the queue number of planar graphs. Proceedings of 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS), 2010, pp. 365–374.

[2] V. Dujmović, P. Morin, and D.R. Wood. Layout of graphs with bounded tree-width. SIAM J. Comput. 34 (2005) 553–579.

[3] E. Di Giacomo, G. Liotta, and H. Meijer. Computing straight-line 3D grid drawings of graphs in linear volume. Computational Geometry 32 (2005) 26–58.

[4] L.S. Heath, F.T. Leighton, and A.L. Rosenberg. Comparing queues and stacks as mechanisms for laying out graphs. SIAM J. Discrete Math. 5 (1992) 398-412.

[5] L.S. Heath and A.L. Rosenberg. Laying out graphs using queues. SIAM J. Comput. 21 (1992) 927–958.

[6] S. Rengarajan and C.E.V. Madhavan. Stack and queue number of 2-trees. Proceedings of 1st International Conference on Computing and Combinatorics, LNCS vol. 959, pp. 203–212, 1995, Springer.

[7] A.L. Rosenberg. The Diogenes approach to testable fault-tolerant arrays of processors. IEEE Trans. Comput. C-32 (1983) 902–910.

Canonical ordering for triangulations on the cylinder, with applications to periodic straight-line drawings

Luca Castelli Aleardi* Éric Fusy†

Abstract

We extend the notion of canonical orderings (and underlying Schnyder woods) to cylindric triangulations. This allows us to extend the incremental straight-line drawing algorithm of de Fraysseix et al. to this setting. Our algorithm yields in linear time a crossing-free straight-line drawing of a cylindric triangulation T with n vertices on a regular grid $\mathbb{Z}/w\mathbb{Z}\times[0,h]$, with $w \leq 2n$ and $h \leq n(n-3)/2$. As a by-product, we can also obtain in linear time a crossing-free straight-line drawing of a toroidal triangulation with n vertices on a periodic regular grid $\mathbb{Z}/w\mathbb{Z} \times \mathbb{Z}/h\mathbb{Z}$, with $w \leq 2n$ and $h \leq n(n+3)/2$.

1 Introduction

The problem of efficiently computing straight-line drawings of planar graphs has attracted a lot of attention over the last two decades. Two combinatorial concepts for planar triangulations turn out to be the basis of many classical straight-line drawing algorithms: the *canonical ordering* (a special ordering of the vertices obtained by a shelling procedure) and the closely related *Schnyder wood* (a partition of the inner edges of a triangulations into 3 spanning trees with specific incidence conditions). Algorithms based on canonical ordering [6, 8] are typically incremental, adding vertices one by one while keeping the drawing planar. Algorithms based on Schnyder woods [11] are more global, the (barycentric) coordinates of each vertex have a clear combinatorial meaning (typically the number of faces in certain regions associated to the vertex). Algorithms of both types make it possible to draw in linear time a planar triangulation with n vertices on a grid of size $O(n \times n)$. The problem of obtaining planar drawings of higher genus graphs has been addressed less frequently [9, 10, 3, 5, 12], from both the theoretical and algorithmic point of view. Recently some efficient methods for the straight-line planar drawing of genus g graphs have been described in [3, 5] (to apply these methods the graph needs to be unfolded planarly along a *cut-graph*). However it does not yield (at least easily) periodic representations: for example, in the case of a torus, the boundary vertices (on the boundary of the rectangular polygon) might not be aligned, so that the drawing does not give rise to a periodic tiling.

Our main contribution is to generalize the notion of canonical ordering and the incremental straight-line drawing algorithm of de Fraysseix et al [6] to triangulations on the cylinder. The obtained straight-line drawing of a cylindric triangulation T with n vertices is x-periodic, on a regular grid of the form $\mathbb{Z}/w\mathbb{Z}\times[0,h]$, with $w \leq 2n$ and $h \leq n(n-3)/2$. By a reduction to the cylindric case (the reduction is done with the help of a so-called tambourine [1]), we can also obtain a straight-line drawing of a toroidal triangulation T with n vertices on a grid $\mathbb{Z}/w\mathbb{Z} \times \mathbb{Z}/h\mathbb{Z}$, with $w \leq 2n$ and $h \leq n(n+3)/2$. For the toroidal case we mention that a notion of canonical ordering has been introduced in [4] (this actually works in any genus and yields an efficient encoding procedure) but we do not use it here. We also mention that, independently, an elegant periodic straight-line drawing algorithm for toroidal triangulations has been very recently described in [7], based on so-called *toroidal Schnyder woods* and face-counting operations; in their case the area of the periodic grid is in $O(n^4)$.

2 Preliminaries

Graphs embedded on surfaces A *map* of genus g is a graph (cellularly) embedded on the closed surface of genus g. The map is called planar for $g = 0$ (embedding on the sphere) and toroidal for $g = 1$ (embedding on the torus). A *triangulation* is a map with no loops nor multiple edges and with only triangular faces. A *cylindric map* is a planar map with two marked faces B_1 and B_2 whose boundaries $C(B_1)$ and $C(B_2)$ are simple cycles (possibly $C(B_1)$ and $C(B_2)$ share vertices and edges). The faces B_1 and B_2 are called the *boundary-faces*. Boundary edges and vertices (black circles in Fig. 1) are those belonging to $C(B_1)$ or $C(B_2)$; the other ones are called *inner* vertices (white circles in Fig. 1) and edges.

Periodic drawings Here we consider the problem of drawing a cylindric (resp. toroidal) triangulation on the flat cylinder (resp. flat torus). The flat cylinder is a rectangle with a pair of opposite sides identified; the flat torus is a rectangle with both pairs of op-

*LIX, École Polytechnique, amturing@lix.polytechnique.fr
†LIX, École Polytechnique, fusy@lix.polytechnique.fr

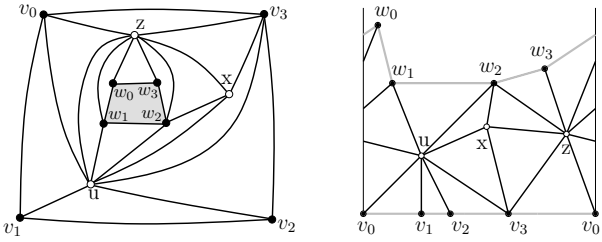


Figure 1: Left: a cylindric triangulation G, with boundary-faces $\{v_0, v_1, v_2, v_3\}$ and $\{w_0, w_1, w_2, w_3\}$. Right: an x-periodic drawing of G.

posite sides identified. We classically require coordinates to be integers. So for a cylindric triangulation we aim at a crossing-free straight-line drawing on a regular (x-periodic) grid of the form $\mathbb{Z}/w\mathbb{Z} \times [0, h]$, and for a toroidal triangulation we aim at a crossing-free straight-line drawing on a regular (x-periodic and y-periodic) grid of the form $\mathbb{Z}/w\mathbb{Z} \times \mathbb{Z}/h\mathbb{Z}$.

3 Canonical ordering for cylindric triangulations

At first we recall the definition of canonical ordering for a planar triangulation G with n vertices, with outer face $\{a, b, c\}$. An ordering $\pi = \{v_1, v_2, \ldots, v_{n-2}\}$ of the vertices of $G\backslash\{a, b\}$ is called a *canonical ordering* if, with G_k the graph induced by vertices $\{a, b, v_1, \ldots, v_k\}$ and C_k the outer face contour of G_k, we have the following conditions:

- The cycle C_k is simple for each $k \geq 1$.
- The vertex v_k belongs to C_k, and all its neighbors in G_{k-1} appear consecutively on C_{k-1}.

Such an ordering is classically computed in decreasing order; at each step k from $n-2$ to 1, one chooses a vertex $v \notin \{a, b\}$ in the current outer contour C such that v is incident to no chord of C (such a vertex v is called *free*), one assigns $v_k \leftarrow v$, and update the figure by deleting v together with its incident edges (note that one must have $v_{n-2} = c$). On the way one can compute the *underlying Schnyder wood*. Precisely, when removing a free vertex v, having neighbors $v_\ell, v_{\ell+1}, \ldots, v_{r-1}, v_r$ on C, perform the following operations:
• assign color 0 to edge (v, v_ℓ), direct it toward v_ℓ;
• assign color 1 to edge (v, v_r), direct it toward v_r;
• assign color 2 to all remaining incident edges, direct them toward v.

We now consider the concept of canonical ordering (and underlying Schnyder wood) in the cylindric case.

Definition 1 *Let G be a cylindric triangulation with boundary-faces B_1 and B_2, and such that the cycle $C(B_1)$ has no chords. An ordering $\pi = \{v_1, v_2, \ldots, v_n\}$ of the vertices of $G\backslash C(B_1)$ is called a (cylindric) canonical ordering if it satisfies:*

- *For each $k \geq 0$ the map G_k induced by $C(B_1)$ and by the vertices $\{v_1, \ldots, v_k\}$ is a cylindric triangulation such that the boundary-face different from B_1 contains B_2 in its interior. The contour of this boundary-face is denoted by C_k.*
- *The vertex v_k lies on C_k, and all its neighbors in G_{k-1} appear consecutively on C_{k-1}.*

We now describe a shelling procedure to compute any canonical ordering of a cylindric triangulation G with boundary-faces B_1, B_2. At each step the graph formed by the remaining vertices is a cylindric triangulation, one boundary face remains B_1 all the way, while the other boundary-face (initially B_2) has its contour, denoted by C^t, getting closer to $C(B_1)$. A vertex $v \in C^t$ is *free* if v is incident to no chord of C^t and if $v \notin C(B_1)$. The shelling procedure goes as follows (n is the number of vertices in $G\backslash C(B_1)$): for k from n to 1, choose a free vertex v on C^t, assign $v_k \leftarrow v$, and then delete v together with all its incident edges. The existence of a free vertex at each step follows from a simple planarity argument.

Similarly as in the planar case [2], this algorithm can be extended in order to assign colors and orientations to the edges of $G\backslash C(B_1)$. When removing a free vertex $v = v_k$, having neighbors $v_\ell, v_{\ell+1}, \ldots, v_{r-1}, v_r$ on C_{k-1}, we perform the following operations:
• assign color 0 to edge (v, v_ℓ), direct it toward v_ℓ;
• assign color 1 to edge (v, v_r), direct it toward v_r;
• assign color 2 to all edges of G_{k-1} incident to v, direct them toward v.

The obtained coloring and orientation of the edges of $G\backslash C(B_1)$ is called the *underlying Schnyder wood* of the canonical ordering. For $i \in \{0, 1, 2\}$, the graph formed by the edges of color i is denoted T_i. It is easy to show that the underlying Schnyder wood S satisfies the following conditions:
• The graph T_2 is an oriented forest spanning all vertices in $G\backslash C(B_2)$, and with all sinks on $C(B_2)$.
• For $i \in \{0, 1\}$, T_i is an oriented forest spanning all vertices in $G\backslash C(B_1)$, and with all sinks on $C(B_1)$.
• Every inner vertex satisfies the local Schnyder wood properties (as in the plane), having exactly 3 outgoing edges (one for each color).
• The edges in cw order around every vertex v lying on B_2 are of the form [1]:
(Seq(In 0), Out 1, Seq(In 2), Out 0, Seq(In 1)).
• The edges in cw order around every vertex lying on B_1 are of the form (e,Seq(In 1), Out 2, Seq(In 0),e'), with e and e' on $C(B_1)$.

Recall that the *dual* of a map G is the map G^* representing the adjacencies of the faces of G, i.e., there is a vertex v_f of G^* in each face f of G, and each edge e of G gives rise to an edge e^* in G^* (if f and f' are

[1] Where $Seq(In\ i)$ denotes a sequence (possibly emtpy) of ingoing edges of color i, and $Out\ i$ denotes an outgoing edge of color i.

the faces on each side of e then e^* connects v_f to $v_{f'}$). Given a cylindric triangulation G with boundary-faces B_1, B_2, the *augmented map* $\widehat{G}$ is obtained from G by adding a vertex b_2 inside B_2 and connecting b_2 to all vertices around B_2 (thus triangulating the interior of B_2). If G is endowed with a canonical ordering π (and the underlying Schnyder wood), let $\widehat{G}^*$ be the dual of $\widehat{G}$. Then the *dual tree* T_2^* for π is the submap of $\widehat{G}^*$ formed by the edges dual to edges that are not in T_2 nor incident to b_2. It can be shown incrementally that T_2^* is a spanning tree of $\widehat{G}^*$ (naturally rooted at b_1 the dual vertex of B_1).

4 Drawing cylindric triangulations

Given a cylindric triangulation G with no chordal edge incident to B_1, we first compute a canonical ordering of G, and then draw G in an incremental way. We start with a cylinder of width $2|C(B_1)|$ and height 0 (i.e., a circle of length $2|C(B_1)|$) and draw the vertices of $C(B_1)$ equally spaced on the circle (space 2 between two consecutive vertices).

Then, we add vertices incrementally as follows: at the k-th step, vertex v_k is added to G_k at position $\mu(P(z_\ell), P(z_r))$, where $P(z_\ell)$ and $P(z_r)$ are respectively the positions of the two neighbors, z_ℓ and z_r, of v_k on the cycle $C(G_k)$, and $\mu(P_0, P_1)$ is the intersecting point of the two half lines, with slopes $+1$ and -1, passing through P_0 and P_1 respectively.

To keep the drawing crossing-free we actually have to stretch the cylinder —increasing its width by 2— just before inserting v_k (this stretching operation is not necessary if all neighbours of v_k are on $C(B_1)$). More precisely, with $z_\ell, z_{\ell+1}, \ldots, z_r$ the neighbours of v_k on C_{k-1}, consider the underlying Schnyder wood restricted to G_{k-1}, and let T_2^* be the dual tree for this Schnyder wood. Let $e_\ell = \{z_\ell, z_{\ell+1}\}$ and $e_r = \{z_{r-1}, z_r\}$. Let P_ℓ (resp. P_r) be the path in T_2^* from e_ℓ^* (resp. e_r^*) to the root of T_2^* (which is the vertex b_1 dual to B_1), see Fig. 2 (bottom-left). We stretch the cylinder by inserting a vertical strip of length 1 along P_ℓ and another along P_r. This means that for each edge dual to an edge in P_ℓ the "x-stretch" of e is increased by 1 and similarly for each edge dual to an edge in P_r (so the x-stretch of an edge dual to an edge in $P_\ell \cap P_r$ is increased by 2).

Fig. 2 shows the execution of the algorithm on an example. The fact that the drawing remains crossing-free relies on an inductive argument similar to the one in the planar case [6].

Proposition 1 *For each cylindric triangulation with n vertices and no chordal edges incident to $C(B_1)$, one can compute in linear time a crossing-free straight-line drawing of G on an x-periodic regular grid $\mathbb{Z}/w\mathbb{Z} \times [0, h]$ where $w \leq 2n$ and $h \leq n(n-3)/2$, such that: every $v \in C(B_1)$ has $y(v) = 0$ (so every edge in $C(B_1)$ has slope 0), and every edge belonging to $C(B_2)\backslash C(B_1)$ has slope ± 1.*

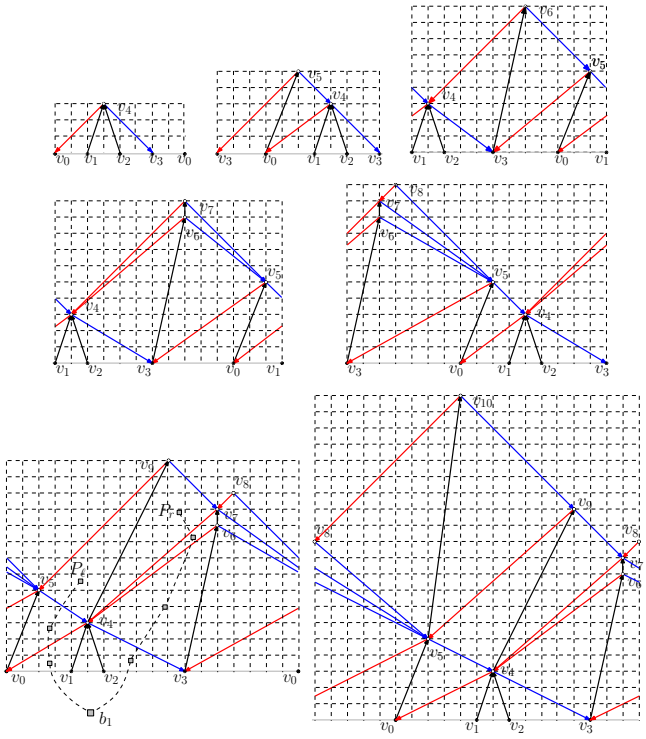


Figure 2: Some steps of the algorithm (Prop. 1) for the x-periodic drawing of a cylindric triangulation (no chordal edges incident to C_1).

We finally explain how to draw a cylindric triangulation allowing chordal edges incident to $C_1 = C(B_1)$; it is good to view B_2 as the top boundary-face and B_1 as the bottom-boundary face (and imagine a standing cylinder). For each chordal edge e of C_1, the *component under* e is the face-connected part of G that lies below e; such a component is a quasi-triangulation (polygonal outer face, triangular inner faces) rooted at the edge e. A chordal edge e of C_1 is *maximal* if the component Q under e is not strictly included in the component under another chordal edge. The *size* of e is defined as $|e| = |V(Q)| - 2$. If we delete the component under each maximal chordal edge (i.e., delete everything from the component except for the chordal edge itself) we get a new bottom cycle C_1' that is chordless, so we can draw the reduced cylindric triangulation G' using the algorithm of Proposition 1. The only difference when drawing G' is in the initial spacing of the vertices of C_1': we space them so that each edge e of $C_1' \cap C_1$ has intial length 2 while each edge e of $C_1'\backslash C_1$ (e is a maximal chordal edge of C_1) has initial length $2|e|$. We denote by $\ell(e)$ the length of e at the end of the execution (the length of e increases during the execution due to stretching operations, so $\ell(e) \geq 2|e|$). Then for each maximal chord e of C_1, we draw the component Q_e under e using the classi-

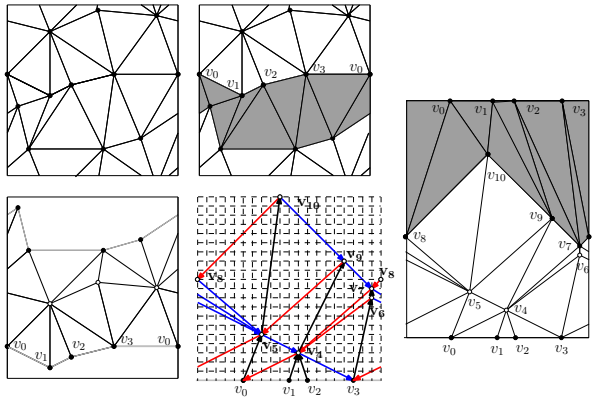


Figure 3: Periodic grid drawing (Theorem 3) of a toroidal triangulation : gray faces represent the so-called tambourine.

cal algorithm by de Fraysseix et al [6]; this drawing has width $2(V(Q_e)-2)$, with e as horizontal bottom edge of length $2(V(Q_e)-2) = 2|e|$ and with the other outer edges of slopes ± 1. We shift the left-extremity of e so that the drawing of Q_e gets width $\ell(e)$, then we rotate the drawing of Q_e by 180 degrees and plug it into the drawing of G'. We obtain:

Theorem 2 *For each cylindric triangulation G with n vertices, one can compute in linear time a crossing-free straight-line drawing of G on an x-periodic regular grid $\mathbb{Z}/w\mathbb{Z}\times[0,h]$, with $w \leq 2n$ and $h \leq n(n-3)/2$. The edges of $C(B_1)$ have slope of absolute value at most 1, the ones not below a chordal edge of $C(B_1)$ having slope 0. The edges of $C(B_2)$ with at least one extremity not on $C(B_1)$ have slope ± 1, the ones with two extremities on $C(B_1)$ have slope 0.*

5 Drawing toroidal triangulations

In order to draw a toroidal triangulation G, we first cut G along a so-called *tambourine* (see [1]), which is a pair of non-contractible oriented cycles C_1 and C_2 such that the edges arriving to C_1 from the right depart at C_2 and the edges arriving to C_2 from the left depart at C_1. Doing this we obtain a cylindric triangulation G' with C_1 and C_2 as the contours of the boundary-faces. We now apply the algorithm of Theorem 2 to G'. If we extend the height h of the drawing by at least $3w/2$, and then wrap the x-periodic grid $\mathbb{Z}/w\mathbb{Z}\times[0,h]$ into a periodic grid $\mathbb{Z}/w\mathbb{Z}\times\mathbb{Z}/h\mathbb{Z}$, and finally insert the edges inside the tambourine as segments, then the slope properties (edges on C_1 and C_2 have slope at most 1 in absolute value while edges inside the tambourine have slope greater than 1 in absolute value) ensure that the resulting drawing is crossing-free (see Fig. 3). We obtain:

Theorem 3 *For each toroidal triangulation G with n vertices, one can compute in linear time a crossing-free straight-line drawing of G on a periodic regular grid $\mathbb{Z}/w\mathbb{Z}\times\mathbb{Z}/h\mathbb{Z}$, with $w \leq 2n$ and $h \leq n(n+3)/2$.*

Acknowledgments

The authors acknowledge the support of the ERC StG 208471 —ExploreMaps. They thank D. Gonçalves and B. Lévêque, and (independently) B. Mohar for interesting discussions about toroidal Schnyder woods, and N. Bonichon for explanations on the computation of a tambourine in a toroidal triangulation.

References

[1] N. Bonichon, C. Gavoille and A. Labourel. Edge partition of toroidal graphs into forests in linear time. In *ICGT*, volume 22, pages 421–425, 2005.

[2] E. Brehm. 3-orientations and Schnyder 3-Tree decompositions. Master's thesis, FUB, 2000.

[3] E. Chambers, D. Eppstein, M. Goodrich, M. Loffler. Drawing graphs in the plane with a prescribed outer face and polynomial area. In *GD*, pages 129–140, 2011.

[4] L. Castelli-Aleardi, E. Fusy, and T. Lewiner. Schnyder woods for higher genus triangulated surfaces, with applications to encoding. *Discr. & Comp. Geom.*, 42(3):489–516, 2009.

[5] C. Duncan, M. Goodrich, S. Kobourov. Planar drawings of higher-genus graphs. *Journal of Graph Algorithms and Applications*, 15:13–32, 2011.

[6] H. de Fraysseix, J. Pach and R. Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10(1):41–51, 1990.

[7] D. Gonçalves and B. Lévêque. Toroidal maps : schnyder woods, orthogonal surfaces and straight-line representation. arXiv:1202.0911, 2012.

[8] G. Kant. Drawing planar graphs using the canonical ordering. *Algorithmica*, 16(1):4–32, 1996.

[9] B. Mohar. Straight-line representations of maps on the torus and other flat surfaces. *Discrete Mathematics*, 15:173–181, 1996.

[10] B. Mohar and P. Rosenstiehl. Tessellation and visibility representations of maps on the torus. *Discrete & Comput. Geom.*, 19:249–263, 1998.

[11] W. Schnyder. Embedding planar graphs on the grid. In *SoDA*, pages 138–148, 1990.

[12] A. Zitnik. Drawing graphs on surfaces. In *SIAM J. Disc. Math*, 7(4):593–597, 1994.

Grid Representations and the Chromatic Number

Martin Balko*

Abstract

A grid drawing of a graph maps vertices to grid points and edges to line segments that avoid grid points representing other vertices. First, we show there is a number of grid points that some line segment of an arbitrary grid drawing of a given graph must intersect. This number is closely connected to the chromatic number. Second, we study how many columns we need to draw a graph in the grid, introducing some new NP-complete problems. Finally, we show that any planar graph has a planar grid drawing where every line segment contains exactly two grid points. This result proves conjectures asked by David Flores-Peñaloza and Francisco Javier Zaragoza Martinez.

1 Introduction

Let $G = (V, E)$ be a simple, undirected and finite graph. A *k-coloring* of G is a function $f\colon V \to C$ for some set C of k colors such that $f(u) \neq f(v)$ for every edge $uv \in E$. If such k-coloring of G exists, then G is *k-colorable*. The *chromatic number* $\chi(G)$ of G is the least k such that G is k-colorable.

For integer $d \geq 2$, a column in the grid $\mathbb{Z}^d$ with *rank* $(x_1, \ldots, x_{d-1}) \in \mathbb{Z}^{d-1}$ is the set $\{(x_1, \ldots, x_{d-1}, x) \mid x \in \mathbb{Z}\}$. Let $\overline{xy}$ denote the closed line segment joining two grid points $x, y \in \mathbb{Z}^d$. The line segment $\overline{xy}$ is *primitive* if $\overline{xy} \cap \mathbb{Z}^d = \{x, y\}$.

Definition 1 *A grid drawing $\phi(G)$ of G in $\mathbb{Z}^d$ is an injective mapping $\phi\colon V \to \mathbb{Z}^d$ such that, for every edge $uv \in E$ and vertex $w \in V$, $\phi(w) \in \overline{\phi(u)\phi(v)}$ implies that $w = u$ or $w = v$.*

2 Complexity of the Grid Drawings

A graph G is said to be *(grid) locatable in $\mathbb{Z}^d$* if there exists a grid drawing of G in $\mathbb{Z}^d$ where every edge is represented by primitive line segment (such drawing is also called *primitive*). Finding a primitive grid drawing of G is called *locating the graph G*. David Flores-Peñaloza and Francisco Javier Zaragoza Martinez showed [7] the following characterization:

Theorem 1 ([7]) *A graph G is locatable in the plane if and only if G is 4-colorable.*

Therefore not all graphs are locatable and every (two-dimensional) grid drawing of any k-colorable graph, where $k > 4$, contains a line segment which intersects at least three grid points. This led us to a generalization of the concept of locatability. Let the number $gp(\phi(G))$ denote the maximal number of grid points any line segment of a grid drawing $\phi(G)$ intersects.

Definition 2 *A graph G is (grid) q- locatable in $\mathbb{Z}^d$, for some positive integer $q \geq 2$, if there exists a grid drawing $\phi(G)$ in $\mathbb{Z}^d$ such that $gp(\phi(G)) \leq q$.*

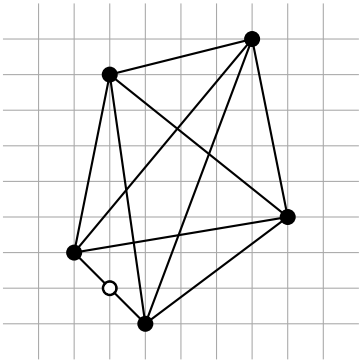


Figure 1: Grid drawing of K_5

The complexity of grid drawings of a graph G is understood as the minimum of $gp(\phi(G))$ among all grid drawings $\phi(G)$. For example, the graph K_5 is five-colorable, thus it is not (two-)locatable. However the grid drawing in Figure 1 shows that K_5 is three-locatable (the third grid point on line segment is denoted by an empty circle). The main result of this section is a much stronger version of Theorem 1.

Theorem 2 *For integers $d, q \geq 2$, a graph G is q^d-colorable if and only if G is q-locatable in $\mathbb{Z}^d$.*

Proof. A trivial but useful observation is that the line segment $\overline{ab}$ between the grid points $a, b \in \mathbb{Z}^d$ intersects exactly the grid points of the form $\left(a_1 + i\frac{b_1 - a_1}{\alpha}, \ldots, a_d + i\frac{b_d - a_d}{\alpha}\right)$ where $0 \leq i \leq \alpha$ and $\alpha = \gcd(|a_1 - b_1|, \ldots, |a_d - b_d|)$.

Using this fact, it is not difficult to prove one implication of Theorem 2. Let $\phi(G)$ be a grid drawing of the graph $G = (V, E)$ in $\mathbb{Z}^d$ having $gp(\phi(G)) \leq q$. Consider the function $f\colon \mathbb{Z}^d \to \mathbb{Z}_q^d$ denoted as $f(x_1, \ldots, x_d) = (x_1 \pmod q, \ldots, x_d \pmod q)$. We use

*Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University in Prague, martin.balko@seznam.cz

This is an extended abstract of a presentation given at EuroCG 2012. 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 in a conference with formal proceedings and/or in a journal.

f as coloring of the grid with q^d colors and we show that it is also a proper vertex coloring of G. Assume to the contrary that $f(\phi(u)) = f(\phi(v))$ for some $uv \in E$. Then $u_1 \equiv v_1, \ldots, u_d \equiv v_d \pmod{q}$ which implies $\gcd(|u_1 - v_1|, \ldots, |u_d - v_d|) \geq q$. According to our observation, there are at least $q+1$ grid points lying on the line segment $\overline{\phi(u)\phi(v)}$. This contradicts the fact that G is q-locatable via the drawing $\phi(G)$.

The proof of the reversed implication is more difficult, but it is constructive and for given coloring of G we can find an appropriate grid drawing in time $O(|V|)$. □

Corollary 3 *A graph is 2^d-colorable if and only if is locatable in $\mathbb{Z}^d$, for $d \geq 2$.*

Corollary 4 *For given $d, q \geq 2$, it is NP-complete to decide whether or not the graph G is q-locatable in $\mathbb{Z}^d$.*

Proof. Clearly, the problem belongs to NP. Theorem 2 shows a reduction of the coloring problem, which asks "Does G admit a proper vertex coloring with q^d colors?", to our problem. We can also ensure that the reduction is polynomial. □

3 Compactness

Another criterion in studying the grid drawings is the minimum number of columns we need to use (see, for example, [9]). We would like to know the grid drawings for a given graph which do not take up too much space. Let $\phi(G)$ be a grid drawing of the graph $G = (V, E)$ in $\mathbb{Z}^d$ such that $\phi(v) = (v_1, \ldots, v_d)$ for every vertex $v \in V$.

Definition 3 *If there exist some $c_1, \ldots, c_l \in \mathbb{Z}^{d-1}$ such that, for every vertex $v \in V$, there is $i \in [l]$ and $(v_1, \ldots, v_{d-1}) = c_i$, then we say that G is embeddable on l columns in $\mathbb{Z}^d$.*

If we can find a grid drawing $\phi(G)$ in $\mathbb{Z}^d$, $d > 2$, on l columns, then we can transfer this drawing on l columns in $\mathbb{Z}^2$. We just take each column of the original grid drawing and transfer its points to an arbitrary free column in the plane. Then we may have to shift some columns higher so that no point representing vertex lies on nonadjacent line segment. This is always possible as the number of vertices in G is finite. Therefore we can assume that the grid is two-dimensional.

By the same trick, we can also assume that there is no unused column between two columns in our drawing. If l is the minimum number of columns on which G can be drawn, then we say this grid drawing of G is *compact*.

Even though we do not care about locatability in this case, some trivial upper bounds hold. A graph embeddable on l columns is l-locatable and every q-locatable graph can be drawn, according to Theorem 2, on q^2 columns. However none of these bound is tight, because, for example, there is a locatable graph with a compact grid drawing on three columns (see Figure 2).

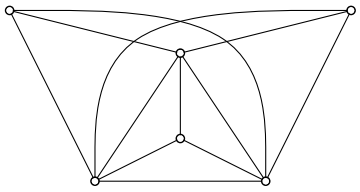


Figure 2: A locatable graph with a compact grid drawing on three columns

The following simple observation characterizes which graphs are embeddable on l lines in the terms of the graph theory.

Observation 1 *A graph $G = (V, E)$ is embeddable on l lines if and only if V can be partitioned into $V_1, V_2, \ldots, V_l$ such that each induced subgraph $G[V_i]$ is isomorphic to a disjoint union of paths.*

Proof. In the grid drawing of G on l columns, the vertices represented by points of a single column define a set V_i. On the other hand, if we have a partition $V_1, V_2, \ldots, V_l$ of V, then each $G[V_i]$ can be drawn on a single column and we can always shift the vertices in such way that the visibility of representing points is guaranteed. □

This observation helps us to show that drawing graphs on the minimal number of columns is NP-complete.

Theorem 5 *For fixed integer $l \geq 2$, it is NP-complete to decide whether or not a graph $G = (V, E)$ is embeddable on l columns of a grid.*

Proof. To prove this theorem, we first use a reduction of Not-All-Equal 3SAT to a problem whether the set V can be partitioned into V_1, V_2 such that $G[V_1]$ and $G[V_2]$ induce disjoint unions of paths (this is a similar method as Hoòng-Oanh Le, Van Bang Lea and Haiko Müller [5] used for a problem of deciding whether V can be partitioned into two induced paths). Then we reduce this new NP-complete problem to the same task with general l. □

If we restrict our attention to only primitive grid drawings, then the situation changes rapidly. According to Theorem 2, only four-colorable graphs have primitive grid drawings in the plane. Also we know, accroding to Corollary 3, that we have to proceed to grid drawings in higher dimensions if we want to

obtain primitive grid drawings of graphs with larger chromatic number.

We also cannot modify a primitive grid drawing by the same trick as before, because shifted line segments could intersect more grid points. Does that mean that some compact primitive grid drawings are necessarily vast and sparse? No, because the proof of the following theorem shows that there are primitive grid drawings with minimal number of columns which take up little space. Also it gives us a characterization of locating similar to Observation 1.

Theorem 6 *For a graph $G = (V, E)$ and integers $d \geq 2$ and l, $2^{d-1} < l \leq 2^d$, the following statements are equivalent:*

1. *G can be located on l columns in $\mathbb{Z}^d$,*
2. *V can be partitioned into $V_1, V_2, \ldots, V_l$ such that $2^d - l$ induced subgraphs $G[V_i]$ induce a disjoint union of paths and the rest induces independent sets.*

Proof. For a graph located on l columns, the desired partition of V can be found by induction on l. On the other hand, if we have such vertex partition, then we can locate G on columns with ranks from the set $\{(r_1, \ldots, r_{d-1}) \in \mathbb{Z}^{d-1} \mid r_1 = 0, 1, 2, 3, r_i = 0, 1\}$. □

Note that if we locate the graph in the smallest possible dimension, according to Corollary 3, then we obtain a grid drawing from this theorem.

We can also characterize graphs which can be located on at most 2^{d-1} columns:

Observation 2 *For a graph $G = (V, E)$ and integers $d \geq 2$ and l, $1 \leq l \leq 2^{d-1}$, the following statements are equivalent:*

1. *G can be located on l columns in $\mathbb{Z}^d$,*
2. *G is embeddable on l columns (in $\mathbb{Z}^2$).*

This observation is somehow intuitive as every grid drawing on two columns is primitive. However we know, according to Theorem 6, that for a larger number of columns this does not hold and locating becomes more restrictive than drawing. Together with Theorem 5, this observation implies the following corollary.

Corollary 7 *For a given $d \in \mathbb{N}$, it is* NP*-complete to decide whether or not a graph G can be located on at most 2^d columns in $\mathbb{Z}^{d+1}$.*

Using this fact we can also prove that locating a graph on minimal number of columns is also a difficult task for general l.

Theorem 8 *For a fixed integer $l \geq 2$, it is* NP*-complete to decide whether or not a graph G can be located on l columns of a grid.*

Proof. Now it suffices to show NP-completeness of this problem only for $2^{d-1} < l \leq 2^d$ where d is the dimension of grid on which we locate G. To do this, we use the characterization of such graphs obtained in Theorem 6 and use the reduction of the coloring problem. □

This result gives an affirmative answer on a question in [4], where the authors ask whether is, in general, the problem of deciding the number of columns needed to locate a graph an NP-hard problem.

Next we show some upper bounds on the minimal number of columns in primitive grid drawings of graphs with known maximal degree, generalizing some known statements about situations in the plane.

In order to prove the next theorem, we need an auxiliary lemma proven by László Lovász.

Lemma 9 ([6]) *Let $G = (V, E)$ be a graph and let $k_1, k_2, \ldots, k_m$ be nonnegative integers with $k_1 + k_2 + \cdots + k_m \geq \Delta(G) - m + 1$. Then V can be partitioned into $V_1, V_2, \ldots, V_m$ so that $\Delta(G[V_i]) \leq k_i$, for all $i \in [m]$.*

Theorem 10 *Let $G = (V, E)$ be a graph with $\Delta(G) \leq 2^{d+1} - 1$, for $d \in \mathbb{N}$. Then G can be located on 2^d columns in $\mathbb{Z}^{d+1}$.*

Proof. According to Observation 2, it suffices to prove that G is embeddable on 2^d columns in the plane. To prove this, we apply Observation 1. So eventually, we show by induction on d that the assumption in our theorem implies that V can be partitioned into $V_1, V_2, \ldots, V_{2^d}$ such that every induced subgraph $G[V_i]$ is isomorphic to a disjoint union of paths. As the basis of the induction we use the proof of a weaker theorem proven in [4] for $d = 1$.

Now we do the inductive step. Let the maximal degree of G be at most $2^{d+1} - 1$. Then, accroding to Lemma 9, V can be partitioned into V_1 and V_2 such that $G[V_1] \leq 2^d - 1$ and $G[V_2] \leq 2^d - 1$, if we set $m = 2$ and $k_1 = k_2 = 2^d - 1$. It follows from the inductive assumption that the vertices of each of the graphs $G[V_1]$, $G[V_2]$ can be partitioned into 2^{d-1} required sets. Together these partitions give the partiton of V into $2^{d-1} + 2^{d-1} = 2^d$ sets. □

Note that the reversed implication does not hold, as every star graph can be located on two columns in the plane and its maximal degree does not have to be bounded.

4 Planar Grid Drawings

Although Theorem 2 and the Four Color Theorem imply that every planar graph is locatable, the drawings obtained by this approach do not have to be planar. On the other hand, De Fraysseix, Pach, and Pollack [2], Schnyder [8], and Chrobak and Nakano [1] proved that any planar graph on n vertices has a planar grid drawing which can be realized in grids of sizes $(2n-4) \times (n-2)$, $(n-2) \times (n-2)$ and $\lfloor 2(n-1)/3 \rfloor \times (4\lfloor 2(n-1)/3 \rfloor - 1)$, respectively. Unfortunately, these drawings are not primitive.

Definition 4 *A primitive planar grid drawing is said to be proper.*

In this section we show that the Four Color Theorem together with Fáry's theorem imply the existence of a proper grid drawing for every planar graph.

Theorem 11 *There exists a proper grid drawing of G for every planar graph G.*

Proof. The main idea is to map a drawing obtained from Fáry's theorem to a grid such that no line segment contains more than two grid points. To find convenient coordinates we use the Four Color Theorem. □

This result gives an affirmative answer to the conjecture asked by Peñaloza and Martinez [7]. The authors point out that proof of this statement would yield an alternate proof of the Four Color Theorem. However we used this theorem as one of the assumptions in the proof. If we use the Five Color Theorem instead, then we get three-locatable planar grid drawings.

In the proof we can choose coordinates of the points in such way that the function $g(a_1, a_2) = (a_1 \pmod 2, a_2 \pmod 2)$ is a vertex coloring of G with $\chi(G)$ colors. Hence we also proved a stronger conjecture from [7], namely the following:

Corollary 12 *Any planar graph has a proper grid drawing such that the function g is a coloring of G that uses exactly $\chi(G)$ colors.*

5 Conclusion

We studied grid drawings from three points of views. First, we showed a connection between the chromatic number of the graph G and the maximal number of grid points that must appear on a line segment of a grid drawing of G. This led to a new classification of graphs according to so called locatability.

Second, we showed that it is NP-complete to find the minimal number of columns on which a graph can be drawn. If we consider only primitive grid drawings, then we have to move to higher dimensions as the chromatic number grows. We also characterized the graphs which can be located on l columns in d-dimensional grid and showed that locating graphs is also NP-complete. Natural question is what happens if we consider grid drawings with both width and height bounded. Such problem is also connected to "No-three-in-line problem" [3].

In the last section we proved that there exist primitive planar grid drawings of an arbitrary planar graph. However the proof of this statement uses a strong result, namely the Four Color Theorem. Perhaps the most intriguing question left open is whether there is a proof of this statement without using the Four Color Theorem. Such proof would yield an alternate proof of this classical result in graph theory.

Acknowledgments

I would like to thank my supervisor Pavel Valtr for his time and for all the provided advice.

References

[1] M. Chrobak and S. Nakano. Minimum-width grid drawings of plane graphs. *Graph Drawing (Proc. GD '94), volume 894 of Lecture Notes in Computer Science*, 10:104–110, 1995.

[2] H. de Fraysseix, J. Pach, and R. Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10(1):41–51, 1990.

[3] R. Guy and P. Kelly. *The no-three-in-line problem.* Research paper. University of Calgary, Dept. of Mathematics, 1968.

[4] J. Cáceres, C. Cortés, C. I. Grima, M. Hachimori, A. Márquez, R. Mukae, A. Nakamoto, S. Negami, R. Robles, and J. Valenzuela. Compact grid representation of graphs. In *XIV Spanish Meeting on Computational Geometry*, pages 121–124, 2011.

[5] H.-O. Le, V. B. Le, and H. Müller. Splitting a graph into disjoint induced paths or cycles. *Discrete Appl. Math.*, 131:199–212, September 2003.

[6] L. Lovász. On decomposition of graphs. *SIAM J Algebraic and Discrete Methods*, 3(1):237–238, 1966.

[7] D. F. Peñaloza and F. J. Z. Martinez. Every four-colorable graph is isomorphic to a subgraph of the visibility graph of the integer lattice. In *Proceedings of the 21st Canadian Conference on Computational Geometry (CCCG2009)*, pages 91–94, 2009.

[8] W. Schnyder. Embedding planar graphs on the grid. In *Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*, SODA '90, pages 138–148, Philadelphia, PA, USA, 1990. Society for Industrial and Applied Mathematics.

[9] D. R. Wood. Grid drawings of k-colourable graphs. Technical report, Vida Dujmović and Attila Pór, 2005.

Memory-Constrained Algorithms for Simple Polygons*

Tetsuo Asano† Kevin Buchin‡ Maike Buchin‡ Matias Korman§ Wolfgang Mulzer¶
Günter Rote¶ André Schulz‖

Abstract

A constant-work-space algorithm has read-only access to an input array and may use only $O(1)$ additional words of $O(\log n)$ bits, where n is the input size. We show how to triangulate a plane straight-line graph with n vertices in $O(n^2)$ time and constant work-space. We also consider the problem of preprocessing a simple n-gon P for shortest path queries, where P is given by the ordered sequence of its vertices. For this, we relax the space constraint to allow s words of work-space. After quadratic preprocessing, the shortest path between any two points inside P can be found in $O(n^2/s)$ time.

1 Introduction

In algorithm development and computer technology, we observe two opposing trends: On the one hand, there are vast amounts of computational resources at our fingertips. On the other hand, more and more specialized tiny devices with limited memory or power supply become available. The first trend leads to software that is written without regard to resources; with this approach, today's latest equipment, be it workstations or hand-held devices, will soon appear unacceptably slow. It is the second trend on which we will focus here: we want to process data with a limited amount of memory. In particular, we consider the setting where the input data is given in a read-only data structure and we have only s words of memory at our disposal, for a parameter $s = o(n)$.

The input of our problem is a polygon P of n vertices in a read-only data structure. We assume that we can access the x- and y-coordinates of any polygon vertex in constant time. We also assume that we can obtain the (clockwise and counterclockwise) neighbor vertex in constant time. We count the storage in terms of the additional number s of *cells* or *words* that an algorithm uses. As usual, a word is large enough to contain either an input item (such as a point coordinate) or an index into the data (of $\log n$ bits). We also assume that basic operations on the input (such as determining if a point is above a given line) take constant time.

Our Results. First, we show how to triangulate a plane straight-line graph (and hence a simple polygon) with constant work-space in $O(n^2)$ time (Section 3). Then, in Section 4, we apply this result to the construction of memory-adjustable data structures for shortest path queries. That is, given a polygon P and a parameter s, we construct a data structure of size $O(s)$ for P. We then use this structure to compute the shortest path between any two points inside P in $O(n^2/s)$ time using $O(s)$ work-space.

Related Work. Given their many applications, a significant amount of research has been dedicated to memory-constrained algorithms, even as early as in the 1980s [6]. A classic example with a geometric flavor is the well-known gift-wrapping algorithm (also known as Jarvis march [7]). It computes the convex hull of a planar n-point set in $O(nh)$ time and $O(1)$ work-space, where h is the number of convex hull vertices. The systematic study of constrained memory in a geometric context was initiated by Asano et al. [2]. Among other results, they describe how to construct well-known geometric structures (such as Delaunay triangulations, the Voronoi diagrams and minimum spanning trees) with a constant number of variables. Recently, Barba et al. [3] gave a constant-work-space algorithm for computing the visibility region of a point inside a simple polygon.

Although we know of no algorithm that triangulates a simple polygon with $o(n)$ work-space, it is known how to find an ear of a given simple polygon P in linear time and with constantly many variables [5]. However, since the input cannot be modified, there seems to be no easy way to extend this method in order to obtain a complete triangulation of P. We also note that there exists a method for triangulating planar point sets [2]. The same paper also contains an

*This work was initiated at the Dagstuhl Workshop on Memory-Constrained Algorithms and Applications, November 21–23, 2011. We are deeply grateful to the organizers as well as the participants of the workshop for helpful discussions during the meeting.

†JAIST, Japan, t-asano@jaist.ac.jp

‡TU Eindhoven, Netherlands, {k.a.buchin,m.e.buchin}@tue.nl MB is supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 612.001.106.

§ULB Brussels, Belgium, mkormanc@ulb.ac.be

¶Freie Universität Berlin, Germany, {mulzer,rote}@inf.fu-berlin.de

‖WWU Münster, Germany, andre.schulz@uni-muenster.de

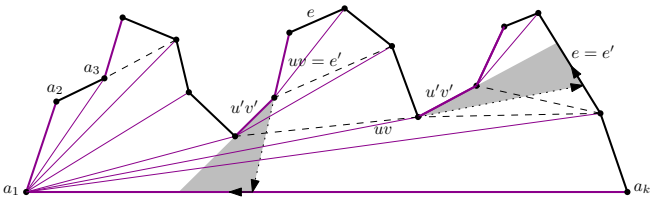


Figure 1: The shortest path tree SPT from a_1 to all other vertices (in purple). Additional triangulation edges generated in our algorithm are dotted. The figure illustrates a forward step (right shaded area) and a backward step (middle area).

algorithm for finding the shortest path between any two points inside a simple polygon. This algorithm uses constant work-space and runs in quadratic time.

For space reasons, we can only give a sketchy idea of the algorithms here. A more complete description is available in the full version [1].

2 Triangulating a Mountain

A *monotone mountain* (or *mountain* for short) is a polygon H whose vertices $a_1, a_2, \ldots, a_k$ have increasing x-coordinates. The edge a_1a_k is called the *base* of H. In this section, we describe how to triangulate an explicitly given mountain, while using only one scan over the boundary of H. This will allow us to extend the algorithm in the next section in order to triangulate a mountain that is provided implicitly as part of a decomposition of a plane straight-line graph.

For the algorithm, we need the *shortest path tree* SPT from a_1 to all other vertices of H. We make the following observations, see Figure 1.

Proposition 1 *SPT has the following properties:*
(i) SPT makes only upward bends.
(ii) Each face f of SPT is bounded by the shortest paths from a_1 to two consecutive vertices a_i and a_{i+1}. (This holds in any simple polygon.)
(iii) Each face f of SPT is a pseudotriangle, bounded from below by an SPT edge ua_{i+1}, from the right by an edge a_ia_{i+1} of H, and from above by a concave chain of SPT edges (as seen from inside).
(iv) The face f can be triangulated uniquely by connecting the rightmost vertex a_{i+1} with the reflex vertices on the upper boundary.

The algorithm traverses SPT in depth-first order, visiting the children of each vertex in *counterclockwise* order. Since there is no space for a stack, this must be done in a "stateless" manner. As in an Euler tour, we visit each non-leaf edge twice, once in forward and once in backward direction. However, a backward step immediately followed by a forward step is considered as a single *sideways* step (this corresponds to the case where the tour moves from a node to its sibling in SPT). We call a vertex *finished* if it has been visited by the traversal and will not be visited again. Otherwise, the vertex is *unfinished*. In an Eulerian traversal of SPT, the vertices of H become finished in order from right to left.

Our algorithm maintains two edges: (i) the current edge of the tour uv, with v lying to the right of u; and (ii) the edge $e = a_ia_{i+1}$ of H such that $\{a_{i+1}, a_{i+2} \ldots, a_k\}$ are the finished vertices of the tour. In each step we distinguish three different cases, and we accordingly perform a step as follows.

Case 1: *v is not incident to e.* We perform a *forward* step into the subtree rooted at v.

Case 2: *$v = a_i$, but uv is a chord of H.* We perform a *sideways* step to the next child of u that follows v in counterclockwise order.

Case 3: *$v = a_i$, and uv is an edge $a_{i-1}a_i$ of H.* We do a *backward* step and return to the parent of u.

We start the algorithm with a sideways step from $uv := e := a_1a_k$ (as an exception to the above rules). The algorithm continues until all vertices are finished and it tries to make a backward step from $e = a_1a_2$. The implementation of the three steps is straightforward. The only information about H that is required by the algorithm is one sequential scan of the sequence of vertices $a_1, a_k, a_{k-1}, \ldots, a_2, a_1$. Thus, the algorithm can also be applied if H is given implicitly and if it takes $O(n)$ time to advance from one edge to the next, as in the next section.

Theorem 2 *Let H be a mountain with k vertices given as part of a larger plane straight-line graph with n vertices. Then we can output the triangles of a triangulation of H in time $O(nk)$ and with constant work-space.*

For an explicitly given mountain of k vertices, we get a running time of $O(k^2)$ as a special case.

3 Triangulating a Plane Straight-Line Graph

Let G be a plane straight-line graph. In order to apply Theorem 2 to the problem at hand, we decompose the convex hull of G into mountains by computing the *vertical decomposition* (or *trapezoidation*) of G and by inserting a chord between any two non-adjacent vertices of G contained in the same trapezoid (cf. Chazelle and Incerpi [4]). These edges partition the convex hull into mountains: Since every vertex (except for the left- and the rightmost ones) has at least one incident edge to either side, the faces must be monotone polygons f. By definition of the decomposition, if there were a face f with vertices on both the upper and the lower boundary, there would be a trapezoid with a vertex at the upper and at the lower boundary. In this case, however, we would have inserted a diagonal.

An edge e of G or of the convex hull is a lower base of a mountain if and only if it is incident to more than one trapezoid above it. This can be checked in linear time. (An inserted chord is never the base of a mountain.) Once the base of a mountain H is at hand, we can visit the edges of H in counterclockwise order by enumerating the trapezoids incident to each vertex of H. This takes linear time per trapezoid. Now, in order to triangulate G, we consider each edge of G and each convex hull edge and determine whether it is a base of a mountain H. If so, we triangulate H using Theorem 2. The convex hull edges can be found in linear time per edge through Jarvis march [7]. Thus, our algorithm needs $O(n^2)$ time plus the time for triangulating the mountains. Since the total size of all mountains is $O(n)$, we get the following result.

Theorem 3 *Given a plane straight-line graph G with n vertices, we can output all triangles in a triangulation of G in $O(n^2)$ time with constant work-space.*

4 Memory-Adjustable Data Structures

Generally, the purpose of a *data structure* D for some set S of n objects is to answer certain types of *queries* on the set S efficiently. For this, D stores some useful information about S. Ideally, D has linear size, and the query algorithm searches within D with only $O(1)$ work-space (more precisely, a work-space of $O(\log n)$ bits). In the classical setting, the whole input data is contained in the data structure, so the storage must be at least as large as the input.

Here, we take a different approach: recall that our input is read-only and cannot be modified. Thus, our approach is to preprocess the data and to store some additional information in a data structure of size s (for some parameter $s = o(n)$). The objective is to design an algorithm that uses this additional information in a way that allows for efficient query processing.

In the following, we use this model for computing the shortest path between two points inside a simple polygon P. In our allotted space, we store a partition of P with $O(s)$ chords into $O(s)$ subpolygons with $\Theta(n/s)$ vertices each. The boundary of each subpolygon is given by subsequences of the original polygon boundary and some chords (hereafter called *cut edges*) associated with it. Additionally, we store the adjacencies between the subpolygons across the cut edges. The total space to represent these subpolygons is $O(s)$.

Theorem 4 *Let P be an n-gon, and let $s \in \{1, \ldots, n\}$. There are $O(s)$ pairwise non-intersecting chords that partition P into $O(s)$ subpolygons, each having $\Theta(n/s)$ vertices. The chords can be found in $O(n^2)$ time using constant work-space.*

We now describe the query algorithm. Given two points $p, q \in P$, we would like to compute the geodesic π_{pq} between them—putting to use the precomputed polygon decomposition. The general idea is to apply the constant work-space method of Asano et al. [2] and to concatenate the resulting paths. Let us thus first give a quick overview of this method.

The algorithm of [2] uses the following invariant: we have partially reported the shortest path from p up to an intermediate point v (where either $v = p$ or v is a reflex vertex of P). Furthermore, we store a *visibility cone* (i.e., a search direction) in which we know that π_{pq} must go. The visibility cone is represented by a wedge emanating from v. The algorithm then shoots a ray inside the wedge that partitions the polygon into two parts. Depending on which part the target q is in, we might obtain a new intermediate point, or a portion of the polygon will be discarded.

Our algorithm uses a similar strategy, but it restricts itself to a subpolygon whenever possible. We start by locating the subpolygons P_p, P_q containing p and q. If $P_p = P_q$, we apply the constant work-space method within that subpolygon. Otherwise, consider the tree associated with the polygon partition and find the path between P_p and P_q. Every edge on that path corresponds to a polygon chord that π_{pq} must cross, in the same order. On the other hand, edges of the tree that were not on the path will not be crossed by π_{pq}, so the corresponding cut edges are treated as obstacles.

We first introduce some notation: let P_{curr} be the polygon containing the current vertex v. For any subpolygon P_i that is traversed by π_{pq}, we define the *entrance* as the cut edge that π_{pq} must cross to enter P_i. Analogously, we define the *exit* of P_i.

In the *standard situation*, everything is confined within one subpolygon P_{curr}. In this case, we can apply the search strategy of the constant-work-space algorithm almost without change: pick a search direction, according to the same rules as in [2], and shoot a ray R' partitioning the subpolygon into two.[1] We use the exit chord of P_{curr} to determine which half contains the target. The only problem arises when R' hits the boundary in the exit chord.

In this case, we switch to the *long-jump situation*. We must first complete the ray shooting operation: we continue the ray R' into the adjacent subpolygon. If R' again hits the exit chord of this subpolygon, we continue into the third subpolygon, and so on, until R' hits a point p' on the boundary of P. The ray R' splits the wedge into two parts, and we update the wedge to the part that contains the target. The running time is $O(n/s)$ times the number of subpolygons visited.

[1] We treat the case that v is outside P_{curr} and the two rays bounding the visibility cone hit the boundary of P in P_{curr} as in the standard situation. This extension does not affect the algorithm.

In the general long-jump situation we have a current start vertex v and two shortest paths SP^+ from v to a point p^+ and SP^- from v to a point p^-, forming a *funnel* with apex v. As an invariant, we know that the target lies between p^+ and p^-, and so the shortest path must go into the funnel. Note that p^+ and p^- may lie in different subpolygons P^+ and P^-. In this case, w.l.o.g., we assume that P^+ is more advanced than P^-. Then, our first goal is to incrementally extend the shortest path SP^- to the lower endpoint of entrance gate of P^+ (Procedure *Catch-up*). Whenever $P^- = P^+$, we shoot a ray R' and extend one of the SP edges (Procedure *Extend*), and proceed depending on which side of the ray R' the exit of P^+ lies. From the funnel boundaries SP^+ and SP^-, we only store the $O(s)$ SP edges that cross some cut edge. We now give the details of the two procedures.

Procedure Catch-up. Since our goal is to proceed towards the exit of P^-, we make an angular clockwise sweep inside P^- from the endpoint p^- of the current path SP^-, starting from the direction opposite to the last edge of SP^- and only considering the $O(n/s)$ points of P^- on the lower boundary between the entrance and the exit cut. After determining the first point that is hit, we may remove some of the last vertices from SP^- (possibly also from the beginning of SP^+). Finally, we add a new edge to SP^-.

Since we do not explicitly store SP^- and SP^+, we may have to look for the predecessor or successor edge by a counter-clockwise angular sweep, as in the backward step of Section 2. These operations are only needed if the point that we look for lies in the same subpolygon, hence we will need at most $O(n/s)$ time (which we charge to the removed vertex). The shortest path SP^- will eventually reach the lower endpoint of the exit gate of P^-. Then we advance to the next subpolygon, and iterate until we reach P^+.

Procedure Extend. Assume that one of the two funnel boundaries, say SP^-, has more than one edge. We take the last-but-one edge of SP^- and extend it into P^-. If this ray R' does not hit the exit cut of P^+, we determine, in $O(n/s)$ time, on which side of R' the target lies. If it lies above R' we pop the last edge of the funnel, as in Procedure *Catch-up*, and proceed. If it lies below R' we can throw away everything above R'. The ray R' and the last SP edge start at the same vertex and end in P^+, forming a new funnel. We can thus consider P^+ with these two extra edges as our current polygon P_{curr}, and proceed.

There is also the situation that R' exits through the exit gate. In this case, we extend R' until it hits the boundary of P, in some subpolygon P'. Again, we determine on which side the target lies. If the target lies above R', R' forms the new last edge of SP^-, P^- is advanced to P'. If the target lies below R', we have a simple funnel consisting only of R' and the last edge of SP^-. We advance P^+ to P'. In either case we continue with procedure *Catch-Up*.

Theorem 5 *Let P be an n-gon, and $1 \le s \le n$. We can build a data structure of size $O(s)$ in $O(n^2)$ time and $O(1)$ variables such that shortest path queries in P can be computed in $O(n^2/s)$ time using $O(s)$ variables.*

5 Open Problems

Obvious topics for future research are improvements of the results. For example, it would be interesting to construct a memory-adjustable data structure for triangulating a plane straight-line graph. Also, Theorem 4 describes how to find a good cut edge for a simple polygon by essentially triangulating the whole polygon and giving the most balanced cut. A natural question is if we can obtain a balanced cut in subquadratic time. Moreover, the cut would not necessarily have to be a diagonal connecting two vertices.

References

[1] T. Asano, K. Buchin, M. Buchin, M. Korman, W. Mulzer, G. Rote, and A. Schulz. Memory-constrained algorithms for simple polygons. *CoRR*, abs/1112.5904, 2011.

[2] T. Asano, W. Mulzer, G. Rote, and Y. Wang. Constant-work-space algorithms for geometric problems. *J. Computat. Geometry*, 2(1):46–68, 2011.

[3] L. Barba, M. Korman, S. Langerman, and R. Silveira. Computing the visibility polygon using few variables. In *Proc. of the 22nd International Symposium on Algorithms and Computation (ISAAC'11)*, pages 80–89, 2011.

[4] B. Chazelle and J. Incerpi. Triangulation and shape-complexity. *ACM Trans. Graph.*, 3:135–152, 1984.

[5] H. ElGindy, H. Everett, and G. Toussaint. Slicing an ear using prune-and-search. *Pattern Recogn. Lett.*, 14:719–722, September 1993.

[6] J. Munro and M. Paterson. Selection and sorting with limited storage. *Theor. Comput. Sci.*, 12:315–323, 1980.

[7] R. Seidel. Convex hull computations. In *Handbook of Discrete and Computational Geometry*, chapter 22, pages 495–512. CRC Press, Inc., 2nd edition, 2004.

Kinetic Collision Detection for Low-Density Scenes in the Black-Box Model

Mark de Berg* Marcel Roeloffzen* Bettina Speckmann*

Abstract

We present an efficient method for collision detection in the *black-box KDS model* for a set S of n objects in the plane. In this model we receive the object locations at regular time steps and we know a bound $d_{\max}$ on the maximum displacement of any object within one time step. Our method maintains, in $O((\lambda+k)n)$ time per time step, a compressed quadtree on the bounding-box vertices of the objects; here λ denotes the density of S and k denotes the maximum number of objects that can intersect any disk of radius $d_{\max}$. Collisions can then be detected by testing $O((\lambda+k)^2 n)$ pairs of objects for intersection.

1 Introduction

Collision detection [12, 13] is an important problem in computer graphics, robotics, and N-body simulations. One is given a set S of n objects, some or all of which are moving, and the task is to detect the collisions which occur. In practice collision detection is often performed in two phases: a *broad phase* that serves as a filter and reports a (small) set of potentially colliding pairs of objects, and a *narrow phase* that tests each of these pairs to determine if there is indeed a collision. Here we are concerned only with broad-phase collision detection; more information on the narrow phase can be found in a survey by Kockara *et al.* [11].

Related work. The most common way to perform collision detection is to test for collisions at regular time steps; for graphics applications this is typically every frame. This approach can be wasteful, in particular if computations are performed from scratch every time: if the objects moved only a little, then much of the computation may be unnecessary. In addition, even with small time steps, collisions can be missed.

An alternative is to use the *kinetic-data-structure (KDS) framework* introduced by Basch *et al.* [3]. A KDS for collision detection maintains a collection of certificates (elementary geometric tests) such that there is no collision as long as the certificates remain true. The failure times of the certificates—these can be computed from the motion equations of the objects—are stored in an event queue. When the next event happens, it is checked whether there is a real collision and the set of certificates and the event queue are updated. (In addition, if there is a collision the motion equations of the objects involved are changed based on the collision response.) KDSs for collision detection have been proposed for 2D collision detection among polygonal objects [2, 10], for 3D collision detection among spheres [9], and for 3D collision detection among fat convex objects [1].

The KDS framework is elegant and can lead to efficient algorithms, but it has its drawbacks. One is that it requires knowledge of the exact trajectories (motion equations) to compute when certificates fail. Such knowledge is not always available. Another disadvantage is that some KDSs are complicated and may not be efficient in practice—the collision-detection KDS for fat objects [1] is an example. We therefore study collision detection in the more practical *black-box model* [5, 7]: We receive, for each object $A_i \in S$, its location $A_i(t)$, at regular time steps $t = 1, 2, \ldots$ and we know an upper bound on the maximum displacement $d_{\max}$ of any object within one time step. Our main goal is to obtain provable bounds for broad-phase collision detection in the black-box model.

Results. We present an algorithm for maintaining a compressed quadtree on the set S of objects, and we prove that our algorithm runs in $O((\lambda+k)n)$ time per time step; here λ denotes the density [6] of S, and k denotes the maximum number of objects intersecting any disk of radius $d_{\max}$. The compressed quadtree can be used to report the at most $O((\lambda+k)^2 n)$ potentially colliding pairs of objects to the narrow phase. The basis of our algorithm is a technique to efficiently maintain a compressed quadtree for a set of moving points, which is of independent interest. We describe our results for the planar case, but they generalize to 3- or higher-dimensional space in a straightforward manner.

2 Preliminaries

Assumptions on distribution and displacement. Let S be the set of constant complexity objects in the plane—e.g. a set of triangles or disks—, let $A_j(t)$ be the object $A_j \in S$ at time t, and let

*Department of Computer Science, TU Eindhoven, the Netherlands, {mdberg,mroeloff,speckman}@win.tue.nl.
M. Roeloffzen and B. Speckmann were supported by the Netherlands' Organisation for Scientific Research (NWO) under project no. 600.065.120 and 639.022.707, respectively.

$S(t) = \{A_1(t), \ldots, A_n(t)\}$. To use temporal coherence in our algorithm, we need bounds on the maximum distance the objects can move in relation to their inter-distances—otherwise the locations at time t have no relation to those at time $t+1$ and we can do nothing but compute $\mathcal{T}(t)$ from scratch. Following De Berg *et al.* [5] we make the following assumption.

Displacement Assumption: There is a maximum displacement $d_{\max}$ such that $\text{dist}(A(t), A(t+1)) \leqslant d_{\max}$ for each object $A \in S$ and any time step t.

For simplicity we assume the objects only translate, and we define $\text{dist}(A(t), A(t+1))$ as the length of the translation vector between $A(t)$ and $A(t+1)$. However, our algorithm also works in more general cases, as long as the boundaries of the objects do not move too much. As mentioned, we need to relate the maximum displacement to the inter-object distances:

Distribution Assumption: Any disk of radius $d_{\max}$ intersects at most k objects from $S(t)$, at any time step t.

Our algorithms do not know the value of k; it is used only in the analysis. We also use the concept of *density* [6]. A set S of objects has density λ if, for any disk D, the number of objects $A_j \in S$ intersecting D having $\text{diam}(A_j) \geqslant \text{diam}(D)$ is at most λ. We assume that the density of $S(t)$ is λ at every time step t.

The compressed quadtree. A *quadtree* for a set of points inside a square is a tree representing a subdivision of that square into four equal-sized subsquares (quadrants) that continues recursively until a stopping criterion is met. There can be splits where only one of the four resulting quadrants contains points. In the quadtree this corresponds to a path of nodes with only one non-empty child. A *compressed quadtree* replaces such paths by *compressed nodes*, which have two children: a child for the *hole* representing the smallest quadtree square containing all points, and a child for the *donut* representing the rest of the square. A compressed quadtree for a set of points has linear size. Our main data structure is a compressed quadtree $\mathcal{T}$ on the set P of bounding-box vertices of the objects in S, with the following stopping criterion.

Stopping Criterion: A square σ becomes a leaf when (i) σ contains at most one point from P, or (ii) σ has edge length at most $d_{\max}$.

We use $\text{region}(v)$ to denote the region associated with a node v of $\mathcal{T}$ and assume that $\text{region}(\text{root}(\mathcal{T}))$ is a fixed, large square that always contains all objects.

Observation 1 *Under our Stopping Criterion,* $\text{region}(v)$ *intersects* $O(\lambda + k)$ *objects from* S *for any leaf* v.

The leaf regions of $\mathcal{T}$ intersect only few objects: regions with edge length larger than $d_{\max}$ contain at most one bounding-box vertex and, hence, intersect $O(\lambda)$ objects [4], and regions with edge length at most $d_{\max}$ intersect at most $O(k)$ objects by definition of k.

For each leaf v in $\mathcal{T}$ we maintain a list of objects intersecting $\text{region}(v)$. Broad-phase collision detection is then performed by reporting for each leaf v all $O((\lambda+k)^2)$ pairs of objects intersecting $\text{region}(v)$. Since the tree $\mathcal{T}$ contains $O(n)$ nodes we report at most $O((\lambda+k)^2 n)$ pairs.

A compressed quadtree for a set of n points can be constructed in $O(n \log n)$ time [8]. This holds in an appropriate model of computation, where we can find the smallest canonical square—a canonical square is any square that results from recursive subdivision of the given initial square—containing two given points in $O(1)$ time. Our goal is to show that, in this model of computation, we can efficiently maintain our compressed quadtree as the objects move. We use $\mathcal{T}(t)$ to denote the compressed quadtree on the bounding-box vertices of $S(t)$. In the remainder of this abstract we sketch how to create $\mathcal{T}(t+1)$ from $\mathcal{T}(t)$ in $O((\lambda+k)n)$ time, resulting in the following theorem.

Theorem 1 *Let* S *be a set of* n *moving objects in the plane that adheres to the Displacement and Distribution Assumption and with maximum density* λ. *We can maintain a compressed quadtree for* S *in* $O((\lambda+k)n)$ *time per time step, which allows us to perform broad-phase collision detection resulting in* $O((\lambda+k)^2)$ *pairs of potentially colliding objects.*

3 Maintaining the compressed quadtree

Our compressed quadtree is built on the points in P (the bounding-box vertices). The main problem in updating $\mathcal{T}$ is that a leaf region can border many other leaf regions and many points may move into it. Constructing the subtree replacing that leaf from scratch is therefore too expensive. We solve this by first refining $\mathcal{T}(t)$ into an intermediary tree $\mathcal{T}_1(t)$. We then insert the (moved) points into $\mathcal{T}_1(t)$ to obtain $\mathcal{T}_2(t)$. We insert the (moved) objects into $\mathcal{T}_2(t)$ to obtain $\mathcal{T}_3(t)$ which we prune into $\mathcal{T}(t+1)$ (see Fig 1). Below we describe these steps in more detail. Note that $\mathcal{T}(t)$ remains unchanged, whereas $\mathcal{T}_1(t)$ is first constructed and then changed into $\mathcal{T}_2(t)$, $\mathcal{T}_3(t)$ and $\mathcal{T}(t+1)$.

Refine. The intermediary tree $\mathcal{T}_1(t)$ has the property

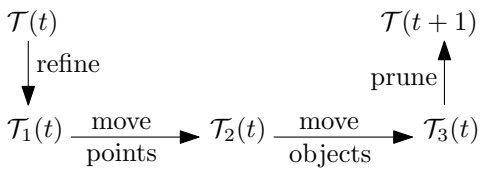


Figure 1: Constructing $\mathcal{T}(t+1)$ from $\mathcal{T}(t)$.

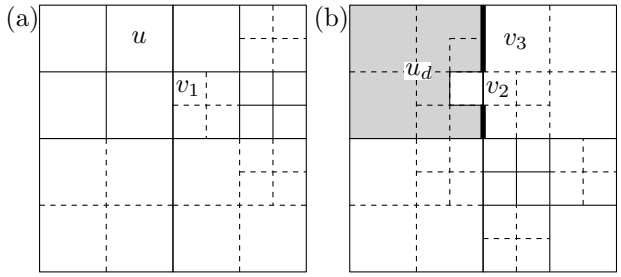


Figure 2: Subdivision defined by $\mathcal{T}(t)$ and its refinement (dashed lines), u_d and u are nodes of $\mathcal{T}(t)$, v_1, v_2, and v_3 are nodes of $\mathcal{T}_1(t)$, region(u_d) is a donut (gray), its right boundary consists of two segments (fat).

that each region in $\mathcal{T}_1(t)$ has $O(1)$ neighbor regions in $\mathcal{T}(t)$. To make this precise we define for a node v in $\mathcal{T}_1(t)$ some related internal or leaf nodes in $\mathcal{T}(t)$:

- original(v): The lowest node u in $\mathcal{T}(t)$ such that region$(v) \subseteq$ region(u) and region(u) is a square.
- $\text{nbr}_\text{n}(v)$, $\text{nbr}_\text{e}(v)$, $\text{nbr}_\text{s}(v)$, $\text{nbr}_\text{w}(v)$: We call these the horizontal and vertical neighbors of v in $\mathcal{T}(t)$. We define the west neighbor $\text{nbr}_\text{w}(v)$ as the lowest node u in $\mathcal{T}(t)$ such that the left boundary of region(v) is included in the right boundary of region(u) (in Fig. 2(b) $\text{nbr}_\text{w}(v_3) = u_d$). The other vertical and horizontal neighbors are defined similarly. (Not all neighbors need to exist.)
- $\text{nbr}_\text{nw}(v)$, $\text{nbr}_\text{ne}(v)$, $\text{nbr}_\text{se}(v)$, $\text{nbr}_\text{sw}(v)$: These are the diagonal neighbors of v in $\mathcal{T}(t)$. We define $\text{nbr}_\text{nw}(v)$ as the lowest node u in $\mathcal{T}(t)$ such that the north-west corner of region(v) is the south-east corner—or a corner of a donut cell created by the corresponding hole—of region(u) (in Fig. 2(a) $\text{nbr}_\text{nw}(v_1) = u$ and in Fig. 2(b) $\text{nbr}_\text{nw}(v_2) = u_d$).
- $\text{nbr}_\text{h1}(v), \text{nbr}_\text{h2}(v)$: These neighbors exist only for donut cells with a hole along the boundary and are the diagonal neighbors of the corner points along the boundary that are created by the hole.

For ease of notation we define the following sets:

$$\mathcal{N}_\text{hv}(v) = \{\text{nbr}_\text{n}(v), \text{nbr}_\text{e}(v), \text{nbr}_\text{s}(v), \text{nbr}_\text{w}(v)\}$$
$$\mathcal{N}_\text{d}(v) = \{\text{nbr}_\text{nw}(v), \text{nbr}_\text{ne}(v), \text{nbr}_\text{se}(v), \text{nbr}_\text{sw}(v), \text{nbr}_\text{h1}(v), \text{nbr}_\text{h2}(v)\}$$

We can now express the conditions on $\mathcal{T}_1(t)$:

(*i*) For every node u in $\mathcal{T}(t)$ such that region(u) is a square, there is a node v in $\mathcal{T}_1(t)$ with region(v) = region(u). Thus, the subdivision induced by $\mathcal{T}_1(t)$ is a refinement of the subdivision induced by $\mathcal{T}(t)$.

(*ii*) For each leaf v in $\mathcal{T}_1(t)$ every node $u \in \mathcal{N}_\text{hv}(v)$ is a leaf of $\mathcal{T}(t)$.

We construct $\mathcal{T}_1(t)$ top-down. Whenever we create a new node v of $\mathcal{T}_1(t)$, it receives a pointer to original(v) and pointers to its horizontal and vertical neighbors in $\mathcal{T}(t)$ (the nodes in $\mathcal{N}_\text{hv}(v)$). These are obtained from the parent of v and the original and neighbors of that parent. How we refine each node v in $\mathcal{T}_1(t)$ depends on original(v) and the neighbors in $\mathcal{N}_\text{hv}(v)$ and is described in more detail in Algorithm 1.

We can prove that each of the cases occurs at most $O(n)$ times and, hence, $\mathcal{T}_1(t)$ also contains $O(n)$ nodes. Besides the pointers to the nodes in $\mathcal{N}_\text{hv}(v)$, we also need pointers to the set $\mathcal{N}_\text{d}(v)$ of diagonal neighbors of each node v in $\mathcal{T}_1(t)$. The details of this are not difficult and omitted due to space limitations.

Moving the bounding-box vertices. We first create for each leaf v in $\mathcal{T}_1(t)$ a list of all points in P contained in region(v) at time $t+1$. We traverse $\mathcal{T}_1(t)$ and for each leaf v we encounter we inspect original(v) and all its neighbors in $\mathcal{N}_\text{hv}(v) \cup \mathcal{N}_\text{d}(v)$—recall that these neighbors are leaves of $\mathcal{T}(t)$. The points contained in these at most eleven leaves—ten neighbors and one original—are the only ones that can be in region(v) at time $t+1$ as points in other cells have more than distance d_max to region(v).

For each of the points in these eleven nodes we check if they are inside region(v) and if so we add them to v. This takes constant time assuming each of these nodes contains only one point. Due to the definition of our quadtree there can be nodes containing more than one point, corresponding to squares that were not refined because their edge length is d_max. Fortunately, these nodes can only be neighbor or original to at most nine nodes in $\mathcal{T}_1(t)$. Each point in these cells is inspected at most nine times and hence points from these cells only require $O(n)$ time in total.

After moving points into v there may be more than one point in v. To ensure that the tree still adheres to the Stopping Criterion we refine v. Since the points come from a constant number of nodes they occupy a constant number of cells of size d_max. Building a compressed quadtree on these cells takes constant time. The result is the second intermediary tree $\mathcal{T}_2(t)$.

Moving the objects. The objects are moved in the same way as the points. We traverse $\mathcal{T}_2(t)$ and for each node v we test each object from $S(t)$ intersecting a neighbor or original of v for intersection with region(v). Since $\mathcal{T}(t)$ adheres to the Stopping Criterion at time t, it follows from Observation 1 that we test only $O(\lambda+k)$ objects for each node in $\mathcal{T}_2(t)$. This results in the intermediary tree $\mathcal{T}_3(t)$.

Pruning the tree. We finally prune $\mathcal{T}_3(t)$ to remove any unnecessary compressed nodes or splits. Splits that put all vertices into one child are replaced by compressed nodes and nested compressed nodes—where the hole of one compressed node is another compressed node—are reduced to a single compressed node. This results in $\mathcal{T}(t+1)$, the compressed quadtree for the objects at time $t+1$.

Algorithm 1: REFINE($\mathcal{T}(t)$)

1 Create root($\mathcal{T}_1(t)$). Set $\mathcal{N}_{\text{hv}}(\text{root}(\mathcal{T}_1(t))) = \emptyset$ and original(root($\mathcal{T}_1(t)$)) = root($\mathcal{T}(t)$);
2 Add root($\mathcal{T}_1(t)$) to empty queue Q;
3 **while** Q *is not empty* **do**
4 $v \leftarrow \text{pop}(Q)$;
5 **Case 1:** original(v) **is split OR a neighbor in** $\mathcal{N}_{\text{hv}}(v)$ **is split:** (see figure) v becomes a split node and we create four children $v_{\text{nw}}, v_{\text{ne}}, v_{\text{se}}, v_{\text{sw}}$ which we add to Q;
6 **Case 2:** original(v) **is a compressed node OR a neighbor in** $\mathcal{N}_{\text{hv}}(v)$ **has a hole on shared boundary:**
7 mirror the top level square of each adjacent hole into region(v) and mirror the hole of original(v) along its top, left, bottom and right boundary;
8 $scs \leftarrow$ the smallest canonical square of the mirrored squares in region(v);
9 **Case 2a:** scs **is empty:** v remains a leaf;
10 **Case 2b:** scs = region(v)**:** v becomes a split node and we create four children $v_{\text{nw}}, v_{\text{ne}}, v_{\text{se}}, v_{\text{sw}}$ which we add to Q;
11 **Case 2c:** $scs \subset$ region(v)**:** v becomes a compressed node and we create v_{d} and v_{h} such that region(v_{d}) = region(v)$\backslash scs$ and region(v_{h}) = scs. Add v_{h} to Q;
12 **Case 3: otherwise:** v remains a leaf;

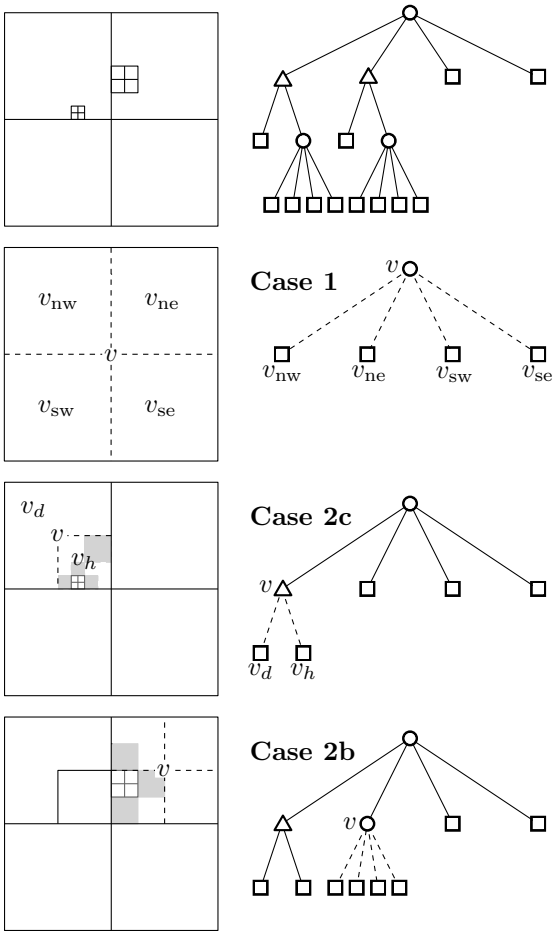


Figure 3: Algorithm 1 (left) to construct $\mathcal{T}_1(t)$ and an illustration (right) of several consecutive steps of the algorithm showing the Cases 1, 2b and 2c. The top tree is the input tree $\mathcal{T}(t)$.

References

[1] M.A. Abam, M. de Berg, S.-H. Poon, and B. Speckmann. Kinetic collision detection for convex fat objects. *Algorithmica*, 53(4):457–473, 2009.

[2] P.K. Agarwal, J. Basch, L.J. Guibas, J. Hershberger, and L. Zhang. Deformable free-space tilings for kinetic collision detection. *Int. J. Robotics Research*, 21(3):179–197, 2002.

[3] J. Basch, L.J. Guibas, and J. Hershberger. Data structures for mobile data. In *Proc. 8th ACM-SIAM Symp. Discr. Alg.* , pages 747–756, 1997.

[4] M. de Berg, H. Haverkort, S. Thite, and L. Toma. Star-quadtrees and guard-quadtrees: I/O-efficient indexes for fat triangulations and low-density planar subdivisions. *Comput. Geom. Theory Appl.* 43:493–513, 2010.

[5] M. de Berg, M. Roeloffzen and B. Speckmann. Kinetic convex hulls and Delaunay triangulations in the black-box model. In *Proc. 27th ACM Symp. Comput. Geom.*, pages 244–253, 2011.

[6] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications (3rd edition).* Springer, 2008.

[7] J. Gao, L.J. Guibas, A. Nguyen. Deformable spanners and applications. In *Proc. 20th ACM Symp. Comput. Geom.*, pages 190–199, 2004.

[8] Sariel Har-Peled. *Geometric Approximation Algorithms.* American Mathematical Society, 2011.

[9] D.-J. Kim, L.J. Guibas, and S.Y. Shin. Fast collision detection among multiple moving spheres. *IEEE Trans. Vis. Comp. Gr.*, 4:230–242 (1998).

[10] D. Kirkpatrick, J. Snoeyink, and B. Speckmann. Kinetic Collision Detection for Simple Polygons. *Int. J. Comput. Geom. Appl.*, 12(1-2):3–27 (2002).

[11] S. Kockara, T. Halic, K. Iqbal, C. Bayrak and R. Rowe. Collision detection: A survey. In *Proc. of SMC*, pages 4046–4051, 2007.

[12] M. Lin and S. Gottschalk. Collision detection between geometric models: A survey. In *Proc. of IMA Conf. Math. Surfaces*, pages 37–56, 1998.

[13] M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M. Cani, F. Faure, Magnenat N. Thalmann, W. Strasser and P. Volino. Collision detection for deformable objects. *Computer Graphics Forum*, 24:119–140, 2005.

Revisiting the Construction of SSPDs in the Presence of Memory Hierarchies*

Sylvie Temme
Faculty of Computer Science,
Technische Universität Dortmund,
44227 Dortmund, Germany
sylvie.temme@cs.tu-dortmund.de

Jan Vahrenhold
Department of Computer Science,
University of Münster, Einsteinstr. 62,
48149 Münster, Germany
jan.vahrenhold@uni-muenster.de

Abstract

We revisit the randomized internal-memory algorithm of Abam and Har-Peled [2] for constructing a semi-separated pair decomposition (SSPD) of N points in $\mathbb{R}^d$ in the context of the cache-oblivious model of computation. Their algorithm spends $\mathcal{O}\left(\varepsilon^{-d} N \log_2 N\right)$ time (assuming that the floor function can be evaluated in constant time, $\mathcal{O}\left(\varepsilon^{-d} N \log_2^2 N\right)$ time otherwise) in expectation and produces a linear-sized SSPD in which each point participates in only a logarithmic number of pairs with high probability.

Using a modified analysis of their algorithm and several cache-oblivious techniques for tree construction, labeling, and traversal, we obtain a cache-oblivious algorithm that spends an expected number of $\mathcal{O}\left(\mathrm{sort}(\varepsilon^{-d} N) \log_2 N\right)$ memory transfers.

1 Introduction

Given two point sets $A, B \subset \mathbb{R}^d$ and a real constant $s > 0$, we say that A and B are *well-separated* with respect to s if the distance between A and B is at least $s \cdot \max\{\mathrm{diameter}(A), \mathrm{diameter}(B)\}$. Callahan and Kosaraju [4] defined an *s-well-separated pair decomposition* (s-WSPD) for a point set P and $s > 0$ as a set $\{\{A_1, B_1\}, \ldots, \{A_m, B_m\}\}$ of pairs of non-empty subsets of P such that

1. $A_i \cap B_i = \emptyset$ for all $i = 1, \ldots, m$,
2. for each pair $\{p, q\}$ of distinct points of P, there is exactly one pair $\{A_i, B_i\}$ in the set, such that $p \in A_i$ and $q \in B_i$ or vice versa,
3. A_i and B_i are well-separated with respect to s for all $i = 1, \ldots, m$.

The integer m is called the *size* of the s-WSPD, its *weight* is $\sum_{i=0}^{m}(|A_i| + |B_i|)$.

In this paper, we are interested in efficiently computing an *s-semi-separated pair decomposition* (s-SSPD). In an s-SSPD the pairs only have to be semi-separated, i.e., the distance between the two point sets is at least $s \cdot \min\{\mathrm{diameter}(A), \mathrm{diameter}(B)\}$ [9].

For the analysis in this paper we use the *cache-oblivious model* [6], which is a variant of the standard two-level I/O-model. This model defines N to denote the number of objects in the problem instance, $M \ll N$ to denote the number of objects fitting into internal memory, and $2 \leq B \leq M/2$ to denote the number of objects per disk block. In the *cache-oblivious model* of Frigo *et al.* [6], we also have the parameters N, M, and B for the analysis of an algorithm, but in the design-phase of an algorithm, the parameters M and B must not be used. This allows us to model a multi-level hierarchical memory, where data is transferred in blocks between any two adjacent layers of memory.Frigo *et al.* [6] showed that sorting N items requires $\Theta\left(\mathrm{sort}(N)\right) = \Theta(\frac{N}{B} \log_{M/B} \frac{N}{B})$ memory transfers while scanning N items can obviously be done in $\Theta\left(\mathrm{scan}(N)\right) = \Theta(\frac{N}{B})$ memory transfers; both bounds match the I/O-bounds in the I/O-model.

Our starting point for the analysis in the cache-oblivious model is a modified version of the analysis of Abam and Har-Peled [2], as their analysis was missing some critical details and in places was incorrect. Following our interaction with the authors, they had fixed their paper, available as [1]. Our starting point is thus very similar to their fixed analysis.

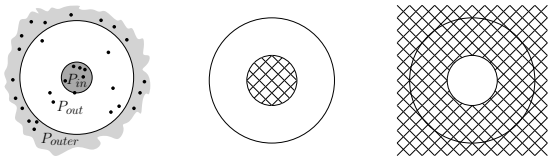
2 The Algorithm of Abam and Har-Peled

The algorithm of Abam and Har-Peled [2] is a randomized, recursive algorithm that efficiently constructs a linear-size semi-separated pair decomposition in which each point participates in a small number of pairs and which thus is of small weight.

Theorem 1 ([1], Theorem 4.10) *Given a point set P in $\mathbb{R}^d$, and a parameter $\varepsilon > 0$, one can compute, in $\mathcal{O}\left(\varepsilon^{-d} N \log_2 N\right)$ expected time, an ε^{-1}-SSPD $\mathcal{S}$ of P, such that: (A) The total expected weight of the pairs of $\mathcal{S}$ is $\mathcal{O}\left(\varepsilon^{-d} N \log_2 N\right)$. (B) Every point of P participates in $\mathcal{O}\left(\varepsilon^{-d} \log_2 N\right)$ pairs, with high probability. (C) The total number of pairs in $\mathcal{S}$ is $\mathcal{O}\left(\varepsilon^{-d} N\right)$.*

*Part of this work has been supported by Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876 "Providing Information by Resource-Constrained Analysis" (http://sfb876.tu-dortmund.de), project A2.

The algorithm proceeds by partitioning the input point set P into three sets, P_{in}, P_{out}, and P_{outer}, induced by two properly chosen concentric balls – see Figure 1(a). Intuitively, the points in P_{outer} are sufficiently far away from the points in P_{in} such that semi-separated pairs can be constructed directly from the unordered Cartesian product $P_{\text{in}} \otimes P_{\text{outer}}$; these pairs are called *long pairs*. In a separate process, the so-called *short pairs* partitioning $P_{\text{in}} \otimes P_{\text{out}}$ are constructed on the basis of a quadtree. Computing pairs involving either pairs from P_{in} or from $P_{\text{out}} \cup P_{\text{outer}}$ is handled recursively – see Figure 1(b) (by construction, each point is passed to only one recursive call).



(a) Partition of P. (b) Pairs handled recursively.

Figure 1: Partitioning scheme and recursive calls.

3 Cache-Oblivious Construction of an SSPD

In this section, we discuss how the algorithm of Abam and Har-Peled needs to be adjusted to be efficient in the cache-oblivious model of computation.

3.1 Partitioning the point set

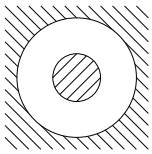
The first step on each level of the recursion is to partition the current working set set P into three sets, P_{in}, P_{out}, and P_{outer}. Abam and Har-Peled give an algorithm that is based on median-finding (or, more precisely, k-selection) and prove that this algorithm constructs a proper partition using, in expectation, no more than a constant number of iterations. Since selecting the element with rank k can be achieved spending $\mathcal{O}(\text{scan}(N))$ memory transfers, we have:

Lemma 2 *For a set P of N points in $\mathbb{R}^d$ and a constant c (depending on d), one can select a point $p \in P$ and compute a radius $r > 0$ such that the ball $B_r(p)$ contains at least $\frac{N}{c}$ points and such that at least $\frac{N}{2}$ points lie outside $B_{20r}(p)$ using $\mathcal{O}(\text{scan}(N))$ memory transfers in expectation.*

Based upon the point p and the radius r constructed by the algorithm implied by the above lemma, Abam and Har-Peled define $P_{\text{in}} := P \cap B_{\mathcal{X}}(p)$, $P_{\text{out}} := P \cap (B_{20r}(p) \setminus B_{\mathcal{X}}(p))$, and $P_{\text{outer}} := P \setminus (P_{\text{in}} \cup P_{\text{out}})$ for some randomly chosen $\mathcal{X} \in [5r, 6r]$.

Lemma 3 *Using an expected number of $\mathcal{O}(\text{scan}(N))$ memory transfers, we can partition the point set P into P_{in}, P_{out}, and P_{outer} such that the sizes of the point sets handled in recursive calls decrease geometrically.*

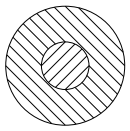
3.2 Constructing long pairs



The ring containing P_{out} separates P_{outer} from P_{in}, and thus the points in P_{outer} are sufficiently far away from any point in P_{in}. Abam and Har-Peled capitalize on this and prove that the following construction yields semi-separated pairs that partition the unordered Cartesian product $P_{\text{in}} \otimes P_{\text{outer}}$: Subdivide the d-dimensional minimal bounding box of P_{in} into cubes of diameter $\varepsilon r/20$. For each cube that contains at least one point of P_{in}, construct a pair whose first component consists of all points in this cube and whose second component is P_{outer}.
For the analysis of this step, we observe that the crucial step in either model of computation is to group the points by cubes. In the analysis of their internal-memory algorithm, Abam and Har-Peled assume that the floor-function can be evaluated in constant time and that thus all points can be hashed into their respective cubes in linear time. While we are allowed to spend more time on assigning points to cubes in the cache-oblivious model of computing (since all we are interested in are memory transfers), we need to spend $\mathcal{O}(\text{sort}(N))$ memory transfers in a postprocessing step on sorting the points by the index of their containing cube to ensure that points in the same cube are stored consecutively in memory.

Lemma 4 *Computing a semi-separated pair decomposition of $P_{\text{in}} \otimes P_{\text{outer}}$ with $\mathcal{O}(\varepsilon^{-d})$ pairs requires $\mathcal{O}(\text{sort}(N))$ memory transfers.*

3.3 Constructing short pairs



The second set of pairs constructed on each level of the recursion comprises a semi-separated pair decomposition of $P_{\text{in}} \otimes P_{\text{out}}$. In contrast to the case of long pairs, points in P_{out} are not sufficiently far away from points in P_{in}, and thus Abam and Har-Peled proceed by first constructing a $\mathcal{O}(\varepsilon^{-1})$-*well*-separated pair decomposition of $P_{\text{in}} \cup P_{\text{out}}$ using a (regular) quadtree and then clean up the output to obtain the desired pair decomposition of $P_{\text{in}} \otimes P_{\text{out}}$.
The algorithm and the original analysis of Abam and Har-Peled [2] intermixes the construction of the regular quadtree and the construction of the short pairs by constructing the quadtree top-down and reporting pairs of nodes (corresponding to pairs of sets of points) as soon as they are found to be well-separated. The algorithm maintains the invariant to only report pairs whose nodes are on the same level of the tree.

To make this step of the algorithm efficient in the cache-oblivious model of computation, we observe two important points: First, the original algorithm expands the quadtree nodes top-down in a fashion that depends on the distribution of the points and thus

is unpredictable and leads to a spatially incoherent memory access pattern. Second, traversing an N-element tree (in fact, even a linked list) has a lower bound of $\Omega(\min\{N, \mathrm{sort}(N)\})$ memory transfers in the cache-oblivious model [5], and thus the best we can hope for is an algorithm with $\mathcal{O}(\mathrm{sort}(N))$ memory transfers that constructs the quadtree.

Building the quadtree The main conceptual change to the algorithm and its original analysis we perform is that we separate the construction of the quadtree from the construction of the well-separated pairs. While this seems only a minor change to the algorithm, it enables us to avoid the unpredictable and spatially incoherent expansion of the quadtree nodes that occurs in the original algorithm. On the other hand, we need to predict the maximal depth of the tree to be constructed if we wish to reuse the analysis of Abam and Har-Peled regarding the properties of their construction and the resulting decomposition.

Lemma 5 *Let P be a set of N points in $\mathbb{R}^d$ and let $0 < \varepsilon < 1$ be a constant. Assume that the sets P_{in} and P_{out} have been constructed according to Lemma 3, and let Q denote the quadtree for $P_{\mathrm{in}} \cup P_{\mathrm{out}}$ of depth $\lfloor \log_2 \varepsilon^{-1} + 13 + \log_2(r/\min\{\|p-q\| - \mathfrak{X}| \mid q \in P_{\mathrm{in}} \cup P_{\mathrm{out}}\})\rfloor$. Then Q contains the quadtree T for $P_{\mathrm{in}} \cup P_{\mathrm{out}}$ constructed by the algorithm of Abam and Har-Peled.*

Proof. Abam and Har-Peled prove that their algorithm does not construct any node on a level deeper than $\lfloor \log_2 \varepsilon^{-1} + \beta + \log_2(r/\min\{\|p-q\| - \mathfrak{X}| \mid q \in P_{\mathrm{in}} \cup P_{\mathrm{out}}\})\rfloor$ for "some constant β" [1, Claim 4.1]. We can augment the original proof by using exact constants and obtain $\beta = \log_2(40c)$ for some $c \geq 2c_R + 2\varepsilon$ where c_R is a constant that can be shown to be lower bounded by 72ε. This yields $\beta = \log_2 5840 < 13$. □

To construct the quadtree efficiently in the cache-oblivious model of computation, we first use the above lemma to compute the depth of the tree. Since, in expectation, we have $\log(r/\min\{\|p-q\| - \mathfrak{X}| \mid q \in P_{\mathrm{in}} \cup P_{\mathrm{out}}\}) \in \mathcal{O}(1)$ [1, Claim 4.2], the expected depth of both the quadtree constructed by Abam and Har-Peled and by our construction is $\mathcal{O}(\log_2 \varepsilon^{-1})$. We then construct the tree bottom-up by first sorting the points according to the $\mathcal{Q}$-order [8, Ch. 2], creating all non-empty leaves of the tree, assigning the points to their containing leaves, and finally creating the next higher level by aggregating the at most 2^d leaves that lie inside a cube of twice the side length. No nodes corresponding to empty cubes are created; this is essential for ensuring the complexity of the algorithm. This process continues upwards while we maintain the invariant that each node v stores a flag inOutStatus indicating whether the points in the leaves of the subtree rooted at v belong to P_{in}, to P_{out}, or to both.

Lemma 6 *A quadtree containing the quadtree constructed by Abam and Har-Peled's algorithm can be constructed spending $\mathcal{O}(\mathrm{sort}(N))$ memory transfers.*

Extracting short pairs To extract short pairs, i.e., pairs for which one component belongs to P_{in} while the other belongs to P_{out}, the algorithm of Abam and Har-Peled incrementally expands pairs of nodes of the quadtree until the nodes and thus the point sets in the two subtrees are well-separated. The algorithm starts with the pair (`root`, `root`) and expands the children of a non-well-separated pair of nodes unless both nodes contain points from either P_{in} or P_{out}, exclusively.

Depending on the distribution of the points, the algorithm of Abam and Har-Peled might access children of a node more than once since they may be involved in multiple pairs. To avoid such spatially incoherent accesses each of which may cause a memory transfer, we modify the algorithm of Abam and Har-Peled to process the quadtree in breadth-first manner. While processing the current level, we maintain a queue that contains the pairs of nodes to be processed sorted first by level (recall that only pairs of nodes on the same level are considered) and then according to the $\mathcal{Q}$-order with respect to the pair's first node.

We then traverse the queue and, just as in the algorithm of Abam and Har-Peled, for each pair of nodes, generate all pairs of indices of children that warrant further inspection. The entries in the resulting queue do not contain, however, the status flags or the number and indices of the node's children (recall that no nodes corresponding to empty cubes are generated, thus a node may have less than 2^d children). This information can be obtained, however, by first sorting the newly created pairs according to the $\mathcal{Q}$-order with respect to the first component followed by a synchronized scan over the Q's next level and then by repeating the same process according to the $\mathcal{Q}$-order with respect to the second component.

To ensure that the resulting pairs indeed induce a partition of $P_{\mathrm{in}} \otimes P_{\mathrm{out}}$, a linear postprocessing scan checks each generated pair and ensures that pairs for which the inAndOutStatus of one or both components indicates that they contain points from P_{in} as well as from P_{out} are split into two pairs.

Abam and Har-Peled show [1, Lemma 4.3] that the total number of pairs generated on all quadtree levels is $\mathcal{O}(\varepsilon^{-d}N)$ in expectation. Since each pair participates in a constant number of scanning and sorting steps, we have the following:

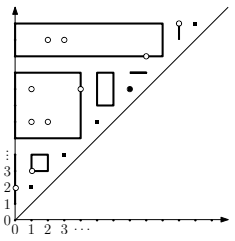
Lemma 7 *Computing a $\frac{c_R}{\varepsilon}$-well-separated pair decomposition of $P_{\mathrm{in}} \otimes P_{\mathrm{out}}$ with an expected number of $\mathcal{O}(\varepsilon^{-d}N)$ pairs requires $\mathcal{O}(\mathrm{sort}(\varepsilon^{-d}N))$ memory transfers in expectation.*

3.4 Reducing the number of pairs

The algorithm described in the previous section constructs $\mathcal{O}\left(\varepsilon^{-d}N\right)$ semi-separated pairs in each recursive step resulting in $\mathcal{O}\left(\varepsilon^{-d}N\log_2 N\right)$ pairs in total. Abam and Har-Peled observe that the short pairs are *well*-separated by construction and use this to merge certain pairs such that the resulting pair still is *semi*-separated. To this end, they first construct a linear-size $\frac{c}{\varepsilon}$-well-separated pair decomposition $\mathcal{W}$ of the point set P. To merge short pairs, they then select an arbitrary representative of each component of a short pair and find the unique pair in $\mathcal{W}$ separating these two points. All short pairs that are mapped to the same pair in $\mathcal{W}$ are then merged into what is proved to be an ε^{-1}-semi-separated pair. Assuming again that the floor-function can be evaluated in constant time and using a hashing scheme, the mapping can be accomplished in constant time per pair.

While we have shown how to efficiently compute the well-separated pair decomposition $\mathcal{W}$ in the cache-oblivious model of computation [7], there are no efficient building blocks for mapping and merging the pairs. It turns out, however, that we can replace the step by a technique originally used for pruning dense spanner graphs [7]: We first use Euler tours and time-forward processing to label the fair split tree S representing $\mathcal{W}$ such that the points stored in the leaves of S are numbered in left-to-right order and such that each internal node stores the minimal bounding interval of the indices of all points in its subtree.

Using synchronized sorting and scanning we then transfer these point labels to the points in P and we observe that each pair of points representing a short pair in the mapping phase now corresponds to a point $(i,j) \in [0, N-1]^2$. Similarly, we observe that each pair of $\mathcal{W}$ corresponds to the cross-product of the intervals stored at the nodes associated with the components of the pair. This implies that we can use (batched) range searching [3] to map (representatives of) short pairs to pairs in $\mathcal{W}$.



As a result of the above mapping process, we obtain, for each pair in $\mathcal{W}$ a (possibly empty) set of short pairs that need to be merged into one pair. This merging process, which is trivial to implement in internal memory, again uses batched range searching. Using a careful implementation, we can organize the memory layout such that all points stored in the leaves of all quadtrees generated during the recursive algorithm are stored and indexed consecutively in one array A (note that each point may occur in $\mathcal{O}\left(\log_2 N\right)$ quadtrees). We use intervals corresponding to each of the short pairs mapped to the same pair of $\mathcal{W}$ as query ranges over the indices of A and label each of the points reported, i.e., each of the points in the original short pair, with the name of the newly constructed pair. Using scanning and sorting, we can aggregate these points and store them consecutively. All steps take $\mathcal{O}\left(\mathrm{sort}(\varepsilon^{-d}N\log_2 N)\right) = \mathcal{O}\left(\mathrm{sort}(\varepsilon^{-d}N)\log_2 N\right)$ memory transfers. Due to Lemma 5, all properties regarding the size and weight of the SSPD constructed carry over from Abam and Har-Peled's analysis:

Theorem 8 *Given a point set P in $\mathbb{R}^d$, and a parameter $\varepsilon > 0$, one can compute, using $\mathcal{O}\left(\mathrm{sort}(\varepsilon^{-d}N)\log_2 N\right)$ expected memory transfers, an ε^{-1}-SSPD $\mathcal{S}$ of P, such that: (A) The total expected weight of the pairs of $\mathcal{S}$ is $\mathcal{O}\left(\varepsilon^{-d}N\log_2 N\right)$. (B) Every point of P participates in $\mathcal{O}\left(\varepsilon^{-d}\log_2 N\right)$ pairs, with high probability. (C) The total number of pairs in $\mathcal{S}$ is $\mathcal{O}\left(\varepsilon^{-d}N\right)$.*

Acknowledgements We thank Mohammad Abam and Sariel Har-Peled for their helpful interactions.

References

[1] M. A. Abam and S. Har-Peled. New constructions of SSPDs and their applications. *Computational Geometry: Theory & Applications*. To appear. An on-line version is available via `http://valis.cs.uiuc.edu/~sariel/research/papers/09/sspd/`.

[2] M. A. Abam and S. Har-Peled. New constructions of SSPDs and their applications. In *Proc. 26th Symp. Computational Geometry*, pp. 192–200. 2010.

[3] G. S. Brodal and R. Fagerberg. Cache oblivious distribution sweeping. In *Proc. 29th Intl. Colloq. Automata, Languages and Programming*, Lecture Notes in Computer Science 2380, pp. 426–438, 2002. Springer.

[4] P. B. Callahan and S. R. Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *Journal of the ACM*, 42(1):67–90, Jan. 1995.

[5] Y.-J. Chiang, M. T. Goodrich, E. F. Grove, R. Tamassia, D. E. Vengroff, and J. S. Vitter. External-memory graph algorithms. In *Proc. 6th Symp. Discrete Algorithms*, pp. 139–149, 1995.

[6] M. Frigo, C. E. Leiserson, H. Prokop, and S. Ramachandran. Cache-oblivious algorithms. In *Proc. 40th Symp. Foundations of Computer Science*, pp. 285–299. 1999.

[7] F. Gieseke, J. Gudmundsson, and J. Vahrenhold. Pruning spanners and constructing well-separated pair decompositions in the presence of memory hierarchies. *Journal of Discrete Algorithms*, 8(2):259–272, Sept. 2010.

[8] S. Har-Peled. *Geometric Approximation Algorithms*. American Mathematical Society, Providence, RI, 2011.

[9] K. R. Varadarajan. A divide-and-conquer algorithm for min-cost perfect matching in the plane. In *Proc. 39th Symp. Foundations of Computer Science*, pp. 320–331. 1998.

Erdős-Szekeres is NP-hard in 3 dimensions - and what now?

Christian Knauer* Daniel Werner†‡

Abstract

The Erdős-Szekeres theorem states that, for every k, there is a number n_k such that every set of n_k points in general position in the plane contains a subset of k points in convex position. If we ask the same question for subsets whose convex hull does not contain any other point from the set, this is not true: as shown by Horton, there are sets of arbitrary size that do not contain an empty 7-gon.

These questions have also been studied extensively from a computational point of view, and polynomial time algorithms for finding the largest (empty) convex set have been given for the planar case. In higher dimension, it is not known how to compute such a set efficiently. In this paper, we show that already in 3 dimensions no polynomial time algorithm exists for determining the largest (empty) convex set (unless P=NP), by proving that the corresponding decision problem is NP-hard. This answers a question by Dobkin, Edelsbrunner and Overmars from 1990.

As a corollary, we derive a similar result for the closely related problem of testing weak ε-nets in $\mathbb{R}^3$. Answering a question by Chazelle et al. from 1995, our reduction shows that the problem is co-NP-hard.

Finally, we make several suggestions for further research on the subject.

1 Preliminaries

The Erdős-Szekeres theorem [8] is one of the major theorems from combinatorial geometry and one of the earliest results in geometric Ramsey theory.

Theorem 1 (Erdős and Szekeres, 1935) *For every k there is a number n_k such that every planar set of n_k points in general position contains k points in convex position.*

Exact values of n_k are known only for very few cases and subject to extensive research, also for the higher dimensional cases.

A closely related question is the following: is the theorem still true if we ask for sets whose convex hull is empty, i.e., does not contain any other point from the set? That this is not the case was shown by Horton [11]: in the plane there are arbitrary large sets which do not contain empty 7-gons. Nicolás [15] and Gerken [9] independently solved the long standing open problem whether or not there is always an empty 6-gon. See Matoušek [12, Chapter 3] for further details and references.

Both these questions generalize to dimension larger than 2 in the obvious way, and clearly the numbers n_k do not increase when the dimension gets larger (proof: project to $\mathbb{R}^2$). See the surveys by Bárány and Károlyi [2] or Morris and Soltan [14] for further references and (more or less) recent progress on the subject.

The corresponding computational problems have also received a lot of attention in the past (e.g., [1], [6], [7], [13]). Polynomial time algorithms are known for both problems in the plane. The fastest algorithm is given in [7], and the question is stated whether a polynomial time algorithm for determining the largest empty convex set also exists in $\mathbb{R}^3$.

1.1 Our results

In this paper, we will consider the following decision problems:

Definition 1 (Erdős-Szekeres) *Let P be a set of points in $\mathbb{R}^d$ and $k \in \mathbb{N}$. Is there a set $Q \subseteq P$ of k points in convex position?*

and

Definition 2 (Largest-Empty-Convex-Set) *Let P be a set of points in $\mathbb{R}^d$ and $k \in \mathbb{N}$. Is there a set $Q \subseteq P$ of k points in convex position whose convex hull does not contain any other point from P?*

Using the reduction technique from Giannopoulos et al. [10], it is an easy exercise to show that both problems are NP-hard if the dimension is not fixed. For people familiar with parameterized complexity: the problem is even W[1]-hard with respect to the dimension d. This means that it is very unlikely to admit an algorithm with running time $O\left(f(d)n^c\right)$ for *any* computable function f and constant c.

Still, this does not exclude the possibility that in every fixed dimension, the problem can be solved with a running time of, say, $O(n^{d+1})$. In this paper, we

*Institut für Informatik, Universität Bayreuth, Germany christian.knauer@uni-bayreuth.de

†Institut für Informatik, Freie Universität Berlin, Germany, daniel.werner@fu-berlin.de

‡This research was funded by Deutsche Forschungsgemeinschaft within the Research Training Group (Graduiertenkolleg) "Methods for Discrete Structures".

show that this cannot be the case (under standard complexity theoretic assumptions):

Theorem 2 *The problems* Largest-Empty-Convex-Set *and* Erdős-Szekeres *are NP-hard in* $\mathbb{R}^3$.

The first part of the theorem, hardness of LECS, is shown in Sec. 2. In Sec. 3, the proof is adapted to ES. In Sec. 4, we derive a similar result for testing weak ε-nets and red-blue discrepancy. Finally, in Sec. 5, we make several suggestions for further research on the subject.

2 The reduction

We will show that the problems is NP-hard by a reduction from the following problem:

Definition 3 (ISNUD) *Given a set of pairwise non-overlapping unit disks in* $\mathbb{R}^2$*, decide whether there are* k *disks such that no two of them touch.*

Here, non-overlapping means that the interiors of the disks are pairwise disjoint. As shown by Cerioli et al. [3], the problem ISNUD is NP-hard. The reduction has the additional property that there is no K_3 in the underlying geometric graph, which we will use in the proof of Lemma 3.

We will now reduce this problem to LECS and show how to adapt it to ES in the next section.

For a given instance $\mathcal{D}$ of unit disks in the plane, we will create a set of points in $\mathbb{R}^3$. These points will *almost* lie on the elliptic paraboloid, in a sense to be made precise later.

For a point $x = (x_1, x_2) \in \mathbb{R}^2$, let

$$\text{lift}\colon (x_1, x_2) \to (x_1, x_2, x_1^2 + x_2^2)$$

denote the standard lifting transform to the paraboloid.

Let D_c denote the n centers of the disks in $\mathcal{D}$. Let L denote the set of all points $\hat{x} = \text{lift}(x)$, for $x \in D_c$.

We now want to forbid certain pairs of points to lie in empty convex positions, namely those for which the corresponding disks intersect. Thus, for a pair of intersecting disks d, d' and their centers $c_d, c_{d'}$, we add a blocking point

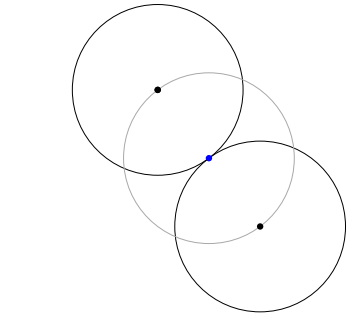
$$b_{dd'} = \frac{1}{2}\left(\text{lift}(c_{d'}) + \text{lift}(c_d)\right).$$

The set B then consist of all the points

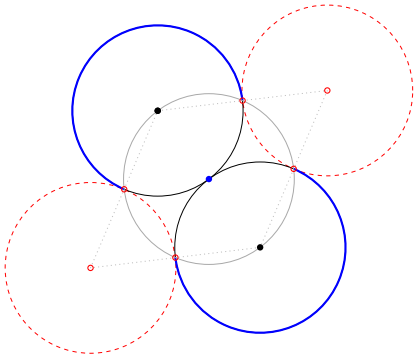
$$\{b_{dd'} \mid d \cap d' \neq \emptyset\},$$

and we set $P = L \uplus B$.

Thus, we have created $O(|\mathcal{D}|)$ points and the reduction is linear in the input size. The main property of the reduction is captured in the folowing lemma.



(a) Two intersecting disks and the projected blocking point



(b) Because the dashed circles do not appear in the construction, all projected blocking points lie on the bold arcs.

Figure 1: Finding an empty circle

Proposition 1 *Let* Q *be a set of points and* h *be a hyperplane through* p *such that* $hx \geq 0$ *for all* $x \in Q$*. Then* p *is in* $ch(Q)$ *if and only if it is in* $ch(Q \cap h)$*.*

Lemma 3 *A blocking point* $b_{dd'}$ *is contained in the convex hull of a set* $Q \subseteq L$ *if and only if* $\hat{c}_d$ *and* $\hat{c}_{d'}$ *are contained in* Q*.*

Proof. $\Leftarrow$: by definition
$\Rightarrow$: We show that there is a hyperplane that contains $b_{dd'}$, $\hat{c}_d$, and $\hat{c}_{d'}$ and has all other points strictly on the positive side. Here we will make use of the fact that our instance consists of non-overlapping unit disks — otherwise, the claim would not hold.

Let C be the circle with center $\text{proj}(b_{dd'})$ through c_d and $c_{d'}$. Because all disks are non-overlapping unit disks, this circle does not contain any other points from D_c. Further, because the underlying graph does not contain a K_3, the circle does not contain any (projection of) a blocking point. See Fig. 1. We then take as h the unique hyperplane whose intersection with the paraboloid projects to the circle C. This hyperplane contains all three points, and because C does not contain any points, all other points from L and thus B lie strictly above h. The claim then follows from Proposition 1.

□

The following states that whether or not a set is in empty convex position will depend only on which points we choose from L. The set B can always be added without destroying this property.

Proposition 2 *The sets L and B each are in empty convex position, and $ch(L) = ch(L \cup B)$.*

Proof. By construction, all points of L lie on the paraboloid. The points from B can be separated from each other by the hyperplane defined in the previous proof. As all of them are convex combinations of points in L, we have $\mathrm{ch}(B) \subseteq \mathrm{ch}(L)$. □

Corollary 4 *A set $L' \uplus Q' \subseteq P$ is in convex position if and only if no point of $Q' \subseteq Q$ is contained in the convex hull of $L' \subseteq L$.*

The main lemma then reads as follows:

Lemma 5 *There is an independent set of size m among the unit disks if and only if there are $m + |B|$ points in empty convex position.*

Proof. $\Rightarrow$: Let I, $|I| = m$, be an independent set among the set of disks. Let $\hat{I} \subset L$ denote the corresponding lifted centers. We claim that $S = \hat{I} \cup B$ is in empty convex position. Indeed, by Corollary 4, no point of $L - \hat{I}$ is in the convex hull of S. Further, by Lemma 3, if some point $b \in B$ was in $\mathrm{ch}(S)$, this would mean that there are two points in $\hat{I}$ that contained b. Thus, the corresponding disks would touch, and I would not be an independent set. This means that there are $m + |B|$ points in empty convex position.
$\Leftarrow$: Now assume that there is no independent set of size m. This means that for any choice of m disks, two of them touch. Now take any set S of $m + |B|$ points. As there are only $|L|+|B|$ points in total, this must contain at least m points from L. Thus, some two of them belong to disks that intersect. By Lemma 3, their convex hull contains a point of B. Thus, S is not in empty convex position. □

3 Adaption to Erdős-Szekeres

We now show how this exact reduction can be applied to Erdős-Szekeres. One direction of Lemma 5 is clear, since we have shown how an independent set of size m results in an empty convex set of size $m + |B|$. For the other direction, we need to show that if there is *any* (not necessarily empty) convex set of $m + |B|$ points, then there is also an independent set of size m among the disks.

Lemma 6 *There is an independent set of size m among the unit disks if and only if there are $m + |B|$ points in convex position.*

Proof. $\Rightarrow$: An empty convex set is convex.
$\Leftarrow$: Let S be a set of $m+|B|$ points in convex position with $|S \cap B| < |B|$. We show how to construct a set S' in convex position of the same size such that $|S' \cap B| = |S \cap B| + 1$.

Let $I = S \cap L$, and let D_I be the corresponding set of disks. Observe that, if $|S \cap B| < |B|$, then $|I| > m$. If all disks from D_I are independent, we are done. Otherwise, let d and d' be two disks from D_I that intersect. The point $b_{dd'}$ cannot be part of S, for otherwise S would not be in convex position. If we thus set $S' = I - \{\hat{d}\} \cup B \cup \{b_{dd'}\}$, by Corollary 4 the set is still in convex position and we have $|S'| = |S|$ and $|S' \cap B| > |S \cap B|$. Thus, after finitely many steps we end up with a set of $m+|B|$ points which contains all points from B. In particular, the set contains no point from B in the convex hull. By Lemma 5, this means that the disks corresponding to the m points from L do not intersect. Thus, we have an independent set of size m. □

This finishes the proof of Thm. 2.

4 Testing weak ε-nets and red-blue discrepancy

Here we shortly mention that the hardness proofs also show hardness for two closely related problems. Recall that a range space is a pair $(X, \mathcal{R})$, where $\mathcal{R} \subset 2^X$. If X is a set of points in $\mathbb{R}^d$ and $\mathcal{R}$ is the set of all convex sets determined by them, a weak ε-net for $(X, \mathcal{R})$ is a set of points S such that $|S \cap R| \neq \emptyset$ whenever $|R \cap X| \geq \varepsilon |X|$, for all $R \in \mathcal{R}$. See Matoušek [12, Chapter 10] for further details.

This leads to the following decision problem as follows:

Definition 4 (ε-Net-Verification) *Given a set of points $P \subset \mathbb{R}^d$, another set $S \subset \mathbb{R}^d$ and an $\varepsilon > 0$. Is S an ε-net for P with respect to all convex sets?*

Chazelle et al. [4] give a polynomial time algorithm for the problem in the plane and ask whether it is solvable in polynomial time in $\mathbb{R}^3$.

A closely related concept is that of red-blue discrepancy: Given a set R of red and a set B of blue points, the *discrepancy* of a set C is defined as

$$D(C) = ||R \cap C| - |B \cap C|| \,.$$

The discrepancy of the set $P = R \cup B$ is then defined as $D(P) = \max_C D(C)$. The corresponding decision problem Red-Blue-Discrepancy asks whether the discrepancy of a given set is at least some value $k \in \mathbb{N}$.

Now observe that the set of blocking points B determines an (m/n)-net[1] for the set of lifted points L if and only if there is *no* empty convex set of size

[1] Recall that n was the number of unit disks.

$m + |B|$ among the points of $L \uplus B$. A similar argument holds for RED-BLUE-DISCREPANCY. Our reduction thus also shows the following:

Theorem 7 *The problem ε-NET-VERIFICATION is co-NP-hard in $\mathbb{R}^3$ and RED-BLUE-DISCREPANCY is NP-hard in $\mathbb{R}^3$.*

5 Conclusion and open problems

The major open question is to find an approximation algorithm for the problems ERDŐS-SZEKERES and LARGEST-EMPTY-CONVEX-SET. The obvious approach (projecting to $\mathbb{R}^2$ and solving the problem there) does not work very well: as shown by Chazelle et al. [5], there are polytopes whose projection in any direction has $\Theta(\log n)$ vertices on the convex hull. This leads to a polynomial time $(\log n)/n$-approximation, which only very few people will be happy about. Thus, the question for a more intelligent (probably constant-factor) approximation algorithm remains and seems to be very challenging.

In addition to this, the most interesting question is the following: Is LARGEST-EMPTY-CONVEX-SET in $\mathbb{R}^3$ fixed parameter tractable with respect to the size of the solution? That is, can we decide whether there are k points in empty convex position in time $O\left(f(k) \cdot n^c\right)$ for some computable function f and constant c? More generally, given a point set P in $\mathbb{R}^d$, can we decide whether there is an empty convex set of size k in time $O\left(f(k)n^{O(d)}\right)$?

Observe that due to the Erdős-Szekeres theorem itself, the problem ERDŐS-SZEKERES is trivially fixed-parameter tractable: Given a point set P and a $k \in \mathbb{N}$, if $n := |P| \leq 2^k$, we use a brute force algorithm, i.e., simply try all subsets of size k. This takes time $\binom{n}{k} \approx n^k \leq (2^k)^k$. If $n > 2^k$, we simply answer yes. In any case, the running time is bounded by 2^{k^2}, and thus we have an algorithm with running time $O\left(f(k)n\right)$. Here, the question for a polynomial size problem kernel is of interest.

Acknowledgements We thank Günter Rote and Nabil Mustafa for pointing us out to [5] and [4], respectively.

References

[1] D. Avis and D. Rappaport. Computing the largest empty convex subset of a set of points. In *Proceedings of the first annual Symposium on Computational geometry*, SCG '85, New York, NY, USA, 1985. ACM.

[2] I. Bárány and G. Károlyi. Problems and Results around the Erdős-Szekeres Convex Polygon Theorem. In *Discrete and Computational Geometry*, volume 2098 of *Lecture Notes in Computer Science*. Springer, 2001.

[3] M. Cerioli, L. Faria, T. Ferreira, and F. Protti. On minimum clique partition and maximum independent set on unit disk graphs and penny graphs: complexity and approximation. *Electronic Notes in Discrete Mathematics*, 18, 2004.

[4] B. Chazelle, H. Edelsbrunner, D. Eppstein, M. Grigni, L. Guibas, M. Sharir, and E. Welzl. Algorithms for weak epsilon-nets, 1995.

[5] B. Chazelle, H. Edelsbrunner, and L. J. Guibas. The complexity of cutting complexes. *Discrete & Computational Geometry*, 4, 1989.

[6] V. Chvátal and G. Klincsek. Finding largest convex subsets. *Congresus Numeratium 29*, 1980.

[7] D. Dobkin, H. Edelsbrunner, and M. Overmars. Searching for empty convex polygons. *Algorithmica 5*, 1990.

[8] P. Erdős and G. Szekeres. A combinatorial problem in geometry. *Compositio Math. 2*, 1935.

[9] T. Gerken. Empty convex hexagons in planar point sets. *Discrete Comput. Geom.*, 39, 2008.

[10] P. Giannopoulos, C. Knauer, M. Wahlström, and D. Werner. Hardness of discrepancy computation and epsilon-net verification in high dimension. *Journal of Complexity*, 2012. In press; available online.

[11] J. D. Horton. Sets with no empty convex 7-gons. *C. Math. Bull. 26*, 1983.

[12] J. Matoušek. *Lectures on Discrete Geometry*. Springer, 2002.

[13] J. S. B. Mitchell, G. Rote, G. Sundaram, and G. Woeginger. Counting convex polygons in planar point sets. *Information Processing Letters 56*, 1995.

[14] W. Morris and V. Soltan. The Erdős-Szekeres problem on points in convex position – a survey. *Bull. Amer. Math. Soc. 37*, 2000.

[15] C. M. Nicolás. The empty hexagon theorem. *Discrete Comput. Geom.*, 38, September 2007.

Unsolvability of the Weighted Region Shortest Path Problem*

Jean-Lou De Carufel† Carsten U. Grimm‡† Anil Maheshwari† Megan Owen§ Michiel Smid†

Abstract

Let $\mathcal{S}$ be a subdivision of the plane into polygonal regions, where each region has an associated positive weight. The *weighted region shortest path problem* is to determine a shortest path in $\mathcal{S}$ between two points $s, t \in \mathcal{S}$, where the distances are measured according to the weighted Euclidean metric - the length of a path is defined to be the weighted sum of (Euclidean) lengths of the subpaths within each region. We show that this problem cannot be solved in the *Algebraic Computation Model over the Rational Numbers* (ACM$\mathbb{Q}$). The ACM$\mathbb{Q}$ can compute exactly any number that can be obtained from the rationals $\mathbb{Q}$ by applying a finite number of operations from $+$, $-$, $\times$, $\div$, $\sqrt[k]{}$, for any integer $k \geq 2$. Our proof uses Gröbner bases and Galois theory, and is based on Bajaj's technique.

1 Introduction

The weighted region shortest path problem is one of the classical path problems in Computational Geometry and has been studied over the last two decades. It was originally introduced in [11] as a generalization of the two-dimensional shortest path problem with obstacles. There are several well known approximation algorithms for this problem; see, e.g., [1, 5, 11]. In this paper, we show that determining the exact shortest path distance in this setting is an unsolvable problem in an algebraic model of computation, confirming the suspicion expressed in [11, Section 4].

The algebraic complexity of geometric optimization problems was first studied by Bajaj, who showed that Euclidean shortest paths among polyhedral obstacles in three dimensions [3] and solutions to the Weber problem and its variations [4] cannot be expressed as finite algebraic expressions. More recently, the algebraic complexity of semi-definite programming [12] and shortest paths through certain cube complexes [2] were investigated. We employ Bajaj's technique [4] to show that the weighted region shortest path problem is unsolvable. In a nutshell, the technique is as follows.

As a consequence of the fundamental theorem of Galois [9], we know that there are no general formulas to solve a polynomial equation of degree $d \geq 5$ *using radicals*. However, there are some polynomial equations of degree $d \geq 5$ that can be solved by radicals. It is the *Galois group* $\mathrm{Gal}(p)$ of an irreducible polynomial p over $\mathbb{Q}$ that determines the solvability of p by radicals. The criterion is the following one: A polynomial equation $p(x) = 0$ is solvable by radicals if and only if $\mathrm{Gal}(p)$ is *solvable* (refer to [9]). We will present an instance of the weighted region shortest path problem such that solving this instance exactly within ACM$\mathbb{Q}$ is equivalent to the statement that the polynomial equation $p_{12}(x) = 0$ in (11) is solvable. However, we will show that the Galois group of p_{12} is S_{12} (i.e., the symmetric group over 12 elements) up to isomorphism. This is proved using the following theorem.[1]

Theorem 1 (Bajaj [4]) *Let p be a polynomial of even degree $d \geq 6$. Suppose that there are three prime numbers q_1, q_2 and q_3 that do not divide the discriminant $\Delta(p)$ of p and such that*

$$p(x) \equiv p_d(x) \pmod{q_1}, \quad (1)$$
$$p(x) \equiv p_1(x)p_{d-1}(x) \pmod{q_2}, \quad (2)$$
$$p(x) \equiv p_1'(x)p_2(x)p_{d-3}(x) \pmod{q_3}, \quad (3)$$

where $p_d(x)$ is an irreducible polynomial of degree d modulo q_1; $p_{d-1}(x)$ (respectively $p_1(x)$) is an irreducible polynomial of degree $d-1$ (respectively of degree 1) modulo q_2; $p_{d-3}(x)$ (respectively $p_1'(x)$ and $p_2(x)$) is an irreducible polynomial of degree $d-3$ (respectively of degree 1 and of degree 2) modulo q_3. Then $\mathrm{Gal}(p) \cong S_d$.

If $d \geq 5$ is odd, the same result holds if we replace (3) by

$$p(x) \equiv p_2(x)p_{d-2}(x) \pmod{q_4}, \quad (4)$$

where q_4 is a prime number such that $q_4 \nmid \Delta(p)$ and $p_{d-2}(x)$ (respectively $p_2(x)$) is an irreducible polynomial of degree $d-2$ (respectively of degree 2) modulo q_4.

*This work has been partially funded by NSERC and FQRNT

†School of Computer Science, Carleton University, Ottawa, Canada

‡Faculty of Mathematics, Otto-von-Guericke University, Magdeburg, Germany

§School of Computer Science, University of Waterloo, Waterloo, Canada

[1] Alternatively, it can be verified using symbolic computation software. For example, GAP uses the algorithm from [8] to test the solvability of polynomials up to degree 15 via the command `isSolvable`, and MAGMA implements an extension of the algorithm in [10], that works for polynomials of arbitrary degree, limited only by time and space constraints.

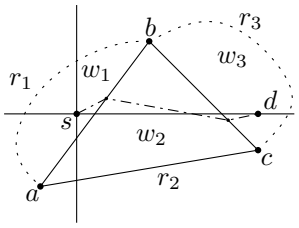


Figure 1: The weighted region shortest path problem with two bends.

Observe that (1) implies that $p(x)$ is irreducible over $\mathbb{Q}$, which implies that $\mathrm{Gal}(p)$ is a *transitive* group. (2) and (3) guarantee the existence of a $(d-1)$-cycle and of an element with cycle decomposition $(2, d-3)$ in $\mathrm{Gal}(p)$. These two elements, together with the transitivity of $\mathrm{Gal}(p)$, imply that $\mathrm{Gal}(p) \cong S_d$.

Lemma 2 (Chapter 4 [9]) *A symmetric group S_n over n elements is solvable if and only if $n \leq 4$.*

If a problem is considered solvable with respect to the ACM$\mathbb{Q}$, we can express its solution as a finite sequence of the allowed operations on the rational input data. For practical applications however, we may need to rely on approximations of such an explicit representation, due to the occurrence of roots. The latter can hardly be avoided for the weighted region shortest path problem, as the length of a path is the weighted sum of Euclidean distances. A problem may be unsolvable in ACM$\mathbb{Q}$ even though its solution can be approximated with sufficient precision in practice. Nonetheless, we use the ACM$\mathbb{Q}$ as a viewpoint to gain insights about algebraic complexity and the applicability of symbolic computation. One advantage of the latter is the reusability of a result without cascaded approximation error. Unsolvability on the other hand concludes any search for a closed formula for solutions and provides further justification for the employment of approximation approaches.

2 Unsolvability

Consider the situation depicted in Fig. 1, where $s = (0,0)$ is the source and $d = (5,0)$ is the target. The three regions r_1, r_2 and r_3 have weights $w_1 = 1$, $w_2 = 2$ and $w_3 = 3$, respectively. Region r_2 is the triangle $\triangle abc$ where $a = (-1,-2)$, $b = (2,2)$ and $c_2 = (5,-1)$. The shortest path (identified by a dashed line) goes through the two edges ba and bc. The equations of the line segments ba and bc are

$$ba : (x,y) = (2,2) + \lambda_1(-3,-4),$$
$$bc : (x,y) = (2,2) + \lambda_2(3,-3),$$

where $\lambda_1, \lambda_2 \in [0,1]$. The function to optimize is therefore

$$\phi(\lambda_1,\lambda_2) = \sqrt{25\lambda_1^2 - 28\lambda_1 + 8} + 2\sqrt{18\lambda_2^2 - 6\lambda_2\lambda_1 + 25\lambda_1^2} + 3\sqrt{18\lambda_2^2 - 30\lambda_2 + 13}.$$

The candidates (λ_1,λ_2) to minimize $\phi(\lambda_1,\lambda_2)$ are $(0,0)$, $(1,1)$ and the solutions of the following system of equations

$$\frac{\partial}{\partial\lambda_1}\phi(\lambda_1,\lambda_2) = 0, \qquad \frac{\partial}{\partial\lambda_2}\phi(\lambda_1,\lambda_2) = 0, \tag{5}$$

where $\lambda_1, \lambda_2 \in [0,1]$. Computing (5), we get

$$\frac{25\lambda_1 - 14}{\sqrt{25\lambda_1^2 - 28\lambda_1 + 8}} + \frac{50\lambda_1 - 6\lambda_2}{\sqrt{18\lambda_2^2 - 6\lambda_2\lambda_1 + 25\lambda_1^2}} = 0, \tag{6}$$

$$\frac{36\lambda_2 - 6\lambda_1}{\sqrt{18\lambda_2^2 - 6\lambda_2\lambda_1 + 25\lambda_1^2}} + \frac{54\lambda_2 - 45}{\sqrt{18\lambda_2^2 - 30\lambda_2 + 13}} = 0, \tag{7}$$

and this system of equations leads to the following one:

$$\begin{aligned} &15100\lambda_1^2 - 52500\lambda_1^3 + 46875\lambda_1^4 \\ &-3624\lambda_1\lambda_2 + 12600\lambda_1^2\lambda_2 - 11250\lambda_1^3\lambda_2 \\ &-3240\lambda_2^2 + 11592\lambda_1\lambda_2^2 - 10350\lambda_1^2\lambda_2^2 = 0, \end{aligned} \tag{8}$$

$$\begin{aligned} &5573\lambda_1^2 - 726\lambda_1\lambda_2 - 13380\lambda_1^2\lambda_2 \\ &+2178\lambda_2^2 + 1800\lambda_1\lambda_2^2 + 8028\lambda_1^2\lambda_2^2 \\ &-5400\lambda_2^3 - 1080\lambda_1\lambda_2^3 + 3240\lambda_2^4 = 0. \end{aligned} \tag{9}$$

The standard technique to isolate λ_1 and λ_2 in this system is to use Buchberger's algorithm [6] from Gröbner bases theory. Gröbner bases theory is a generalization of the ideas from linear algebra for solving a system of linear equations, namely bases and Gaussian elimination, to systems of multivariate polynomials. Just as the solution set to a system of linear equations forms a vector space, the solution set to a system of polynomials forms what is called a *variety*. The goal is to find a set of polynomials whose solution set is the variety of the original polynomials, but which have good computational properties. This is analogous to finding a basis for the nullspace of a system of linear equations. A *Gröbner basis* is such a set, and can be found by running Buchberger's algorithm, which is a generalization of Gaussian elimination. The algorithm has different outcomes depending on the choice of monomial ordering used, i.e., ordering first by total degree of monomial versus degree of first variable. If a lexicographic ordering is used, then the Gröbner basis has the property that it contains a polynomial in only the last variable, a polynomial in only the last two variables, etc., and thus can be used to solve the original system of polynomial equations. This algorithm is implemented in many symbolic computation software such as Mathematica and Singular. Using the command

```
eqn1 = 15100*lambda1^2
-52500*lambda1^3
```

```
+46875*lambda1^4
-3624*lambda1*lambda2
+12600*lambda1^2*lambda2
-11250*lambda1^3*lambda2
-3240*lambda2^2
+11592*lambda1*lambda2^2
-10350*lambda1^2*lambda2^2;

eqn2 = 5573*lambda1^2
-726*lambda1*lambda2
-13380*lambda1^2*lambda2
+2178*lambda2^2
+1800*lambda1*lambda2^2
+8028*lambda1^2*lambda2^2
-5400*lambda2^3
-1080*lambda1*lambda2^3
+3240*lambda2^4;

GB = GroebnerBasis[{eqn1,eqn2},
                   {lambda1,lambda2}];

p15 = GB[[1]]
p14 = GB[[2]]
pp14 = GB[[3]]
```

in Mathematica, we obtain the following equivalent system of three equations[2]:

$$\begin{aligned} p_{15}(\lambda_2) &= 0, \\ p_{14}(\lambda_1, \lambda_2) &= 0, \\ p'_{14}(\lambda_1, \lambda_2) &= 0, \end{aligned} \tag{10}$$

where p_{15} is a polynomial of degree 15 in λ_2, and p_{14} and p'_{14} are polynomials of total degree 14 in λ_1 and λ_2. In (10), $p_{15}(\lambda_2)$ factors into $\lambda_2^3 \cdot p_{12}(\lambda_2)$, where p_{12} is the degree-12 polynomial

$$\begin{aligned} p_{12}(x) = &-745794461011616 \\ &+ 7579514247189376x \\ &- 35835761266110832x^2 \\ &+ 106594990715823828x^3 \\ &- 227453553923353605x^4 \\ &+ 372947142859928208x^5 \\ &- 484843926044252448x^6 \\ &+ 504065307805559136x^7 \\ &- 416814671916200304x^8 \\ &+ 267526211688938880x^9 \\ &- 125338196832117120x^{10} \\ &+ 37669520655360000x^{11} \\ &- 5350784184000000x^{12}. \end{aligned} \tag{11}$$

[2]The corresponding calculations will appear in the full version and will be provided upon request.

Theorem 3 *The weighted region shortest path problem cannot be solved exactly within the* ACM$\mathbb{Q}$.

Proof. Following the notation of Theorem 1, and the above example, we have $p(x) = p_{12}(x)$, $d = 12$ and $\Delta(p) = 2^{117} \cdot 3^{144} \cdot 5^{49} \cdot 7^{16} \cdot 47 \cdot 53^2 \cdot 241 \cdot 5573^2 \cdot 9067 \cdot 13417^2 \cdot 35993^2 \cdot 5242801 \cdot q_{96}$, where q_{96} is a prime number with 96 digits. With $q_1 = 61$, $q_2 = 89$ and $q_3 = 139$, one finds

$$\begin{aligned} p(x) &\equiv 33 + 19x + 40x^2 + 15x^3 + 23x^4 + 33x^5 + 56x^6 \\ &\quad +39x^7 + 57x^8 + 26x^9 + 49x^{10} + 56x^{11} \\ &\quad +17x^{12} \pmod{61}, \\ p(x) &\equiv 38(x+32)(1 + 84x + 57x^2 + 16x^3 + 71x^4 \\ &\quad +86x^5 + 12x^6 + 54x^7 + 12x^8 + 22x^9 \\ &\quad +82x^{10} + x^{11}) \pmod{89}, \\ p(x) &\equiv 95(x+35)(58 + 41x + x^2)(61 + 62x + 42x^2 \\ &\quad +3x^3 + 106x^4 + 57x^5 + 28x^6 + 29x^7 \\ &\quad +106x^8 + x^9) \pmod{139}, \end{aligned}$$

so that $\mathrm{Gal}(p) \cong S_{12}$. Theorem 2 tells us that S_{12} is non-solvable.

With numerical methods, one easily finds that the minimum of $\phi(\lambda_1, \lambda_2)$ for $\lambda_1, \lambda_2 \in [0, 1]$ is at $(\lambda_1, \lambda_2) \approx (0.39398, 0.72450)$, which is a solution of the system of equations (8) and (9). Hence, $\lambda_2 \approx 0.72450...$ is a solution of $p_{12}(\lambda_2) = 0$. But we showed above that $p_{12}(\lambda_2) = 0$ cannot be solved exactly within the ACM$\mathbb{Q}$. Therefore, in general, the weighted region shortest path problem cannot be solved exactly within the ACM$\mathbb{Q}$. □

Observation 1 *Theorem 1 cannot be used if the Galois group* $\mathrm{Gal}(p)$ *of a polynomial* p *of degree* d *is such that* $\mathrm{Gal}(p) \ncong S_d$. *Therefore, it is not an appropriate tool to compute the Galois group of a polynomial* p *in general. It is rather meant for the following: Let* $\mathcal{P}$ *be a problem that can be translated into a (system of) polynomial equation(s). Theorem 1 is a good tool to prove that* $\mathcal{P}$ *cannot be solved exactly within the* ACM$\mathbb{Q}$, *when it is the case. Usually,* $\mathcal{P}$ *admits infinitely many different instances leading to infinitely many different polynomial equations. Our experience shows that most of the time, Theorem 1 applies on the first instance of* $\mathcal{P}$ *we can think of. Otherwise, one can use a symbolic computation software as a black box and compute* $\mathrm{Gal}(p)$. *To use Theorem 1, we need to find three prime numbers that satisfy the constraining properties. Bajaj [4] explains why trying* $d + 1$ *prime numbers that do not divide* $\Delta(p)$ *will most likely be sufficient. As for the factorisation of a polynomial modulo a prime number, refer to [7] for standard algorithms that perform this task.*

3 Generalization of the counter-example to other instances of the problem

We have shown that one instance of the weighted region shortest path problem is unsolvable, which shows this problem is unsolvable in general. However, it would be useful to know how often an instance of the weighted region shortest path problem is unsolvable. If we know the sequence of regions that the shortest path goes through, then we know that the path itself is made up of a sequence of straight lines passing through the interiors of the prescribed regions and that it bends only on the boundaries of the regions. Furthermore, the shortest path is locally optimal in between any two bendpoints. That is, if we treat the bendpoints u_i and u_{i+3} as fixed, then the intermediate bendpoints u_{i+1} and u_{i+2} must be optimal with respect to u_i and u_{i+3}. This implies that any instance of the weighted region shortest path problem in which the shortest path goes through at least three regions, will contain a generalization of the given counter-example. In particular, the equations to optimize will have the same form, but different coefficients. Now a polynomial of degree 5 or higher is unsolvable if its coefficients are algebraically independent, i. e. not related by an algebraic expression. This will be true in the general instance of the problem, except in very specific cases, and thus a generic instance of the weighted region shortest path problem in which the path passes through at least three regions is usually unsolvable.

4 Conclusions & Future Work

We demonstrated that even a toy example of the weighted region shortest path problem cannot be solved in $\mathsf{ACM}\mathbb{Q}$. Thus, we cannot expect this to be the case for inputs of realistic size or for instances of more general versions, such as the flow path [13] or the anisotropic shortest path problem [14]. The method we employed, Bajaj's technique, will be a useful tool-kit to prove similar unsolvability results in the future and guide more realistic analysis of problems in computational geometry with algebraic components.

References

[1] L. Aleksandrov, A. Maheshwari, and J.-R. Sack. Determining approximate shortest paths on weighted polyhedral surfaces. *J. ACM*, 52:25–53, January 2005.

[2] F. Ardila, M. Owen, and S. Sullivant. Geodesics in CAT(0) cubical complexes. *Advances in Applied Mathematics*, 48:142–163, 2012.

[3] C. Bajaj. The algebraic complexity of shortest paths in polyhedral spaces. Technical Report 442, Purdue University, 1985.

[4] C. L. Bajaj. The algebraic degree of geometric optimization problems. *Discrete & Computational Geometry*, 3:177–191, 1988.

[5] P. Bose, A. Maheshwari, C. Shu, and S. Wuhrer. A survey of geodesic paths on 3d surfaces. *Comput. Geom.*, 44(9):486–498, 2011.

[6] B. Buchberger. A theoretical basis for the reduction of polynomials to canonical forms. *SIGSAM Bull.*, 10:19–29, 1976.

[7] H. Cohen. *A Course in Computational Algebraic Number Theory*. Springer, 1993.

[8] A. Distler. Ein Algorithmus zum Lösen einer Polynomgleichung durch Radikale. Master's thesis, TU Braunschweig, 2005. Diplomarbeit.

[9] D. S. Dummit and R. M. Foote. *Abstract Algebra*. John Wiley & Sons, 3rd edition, 2003.

[10] K. Geissler and J. Klüners. Galois group computation for rational polynomials. *Journal of Symbolic Computation*, 30(6):653–674, 2000.

[11] J. S. B. Mitchell and C. H. Papadimitriou. The weighted region problem: finding shortest paths through a weighted planar subdivision. *J. ACM*, 38(1):18–73, 1991.

[12] J. Nie, K. Ranestad, and B. Sturmfels. The algebraic degree of semidefinite programming. *Mathematical Programming*, 122:379–405, 2010.

[13] J. H. Reif and Z. Sun. Movement planning in the presence of flows. *Algorithmica*, 39(2):127–153, 2004.

[14] N. C. Rowe and R. S. Ross. Optimal grid-free path planning across arbitrarily contoured terrain with anisotropic friction and gravity effects. *Robotics and Automation, IEEE Transactions on*, 6(5):540–553, 1990.

The expected number of points in circles

Ruy Fabila-Monroy* Clemens Huemer† Eulàlia Tramuns‡

Abstract

Let S be a plane point set of n points with no three of them collinear and no four cocircular. Consider the circle passing through three points of S chosen uniformly at random and let X be the random variable that counts the number of points of S inside this circle. Urrutia (2004) showed that the expectation $E_S(X)$ of X only depends on the rectilinear crossing number $cr(S)$ of S. We prove that this also holds for the variance $V_S(X)$. More precisely, $E_S(X) = \frac{cr(S)}{\binom{n}{3}} + \frac{n-3}{4}$ and

$$V_S(X) = (n-3)^2 \left(\frac{cr(S)}{8\binom{n}{4}} - \frac{cr(S)^2}{16\binom{n}{4}^2} - \frac{1}{80} + \frac{1}{5(n-3)} \right).$$

Then, consider the smallest circle containing two points of S chosen uniformly at random, and let Y be the random variable that counts the number of points of S inside this circle. We show an upper bound on the expectation $E_S(Y) \leq \frac{n-2}{3}$, and there are point sets S attaining this bound.

1 Introduction

A point set S in the plane is in general position if no three points of the set lie on a common line, and no four points of the set lie on a common circle. Throughout, all considered point sets will be in general position in the plane and $|S| = n$. We denote circles defined by three points of a set S as circumcircles. It is well known that circumcircles that do not contain points of S in the interior define the Delaunay triangulation of S. The number of empty circumcircles is known to be $2n - 2 - h$ where h is the number of extreme points of S. Also relations for the numbers of circumcircles containing more points are known [6]. Denote with c_k the number of circumcircles containing exactly k points of S in its interior. Then, $c_k + c_{n-k-3} = 2(k+1)(n-2-k)$, see [4, 9]. Note that this expression only depends on n but not on the positions of the points of S. However, the value c_k does depend on the positions of the points of S. We study how many points of S lie in the interior of a random circumcircle. More formally, given a set S of n points, let X be the random variable that counts the number of points of S inside the circle defined by three points chosen uniformly at random from S. Let $E_S(X)$ denote the expectation of X and denote by $V_S(X)$ the variance.

Urrutia [11] proved that $E_S(X) \geq (n-3)/3.33..$ and if S is in convex position then $E_S(X) = \frac{n-3}{2}$. We show in Section 2 that $E_S(X)$ and $V_S(X)$ depend only on the rectilinear crossing number $cr(S)$ of S, a well studied parameter that is equal to the number of convex quadrilaterals with vertices in S. In particular, we rediscover the equality $E_S(X) = \frac{cr(S)}{\binom{n}{3}} + \frac{n-3}{4}$ from [11]. Using the current best lower bound on $cr(S)$ [2], this implies $0.344993n - O(1) \leq E_S(X) \leq \frac{n-3}{2}$ for every set S. The current best examples of point sets S with small crossing number [1] have $E_S(X) = 0.345122n - O(1)$.

We further express the variance $V_S(X)$ of X in terms of $cr(S)$ and obtain $0.0259729n^2 - O(n) \leq V_S(X) \leq \frac{(n-3)^2}{20} + \frac{n-3}{5}$ for every set S. The upper bound on $V_S(X)$ is obtained for point sets in convex position and its lower bound for point sets that minimize the rectilinear crossing number. The point sets from [1] have $V_S(X) = 0.0260128n^2 - O(n)$.

We then consider this problem for circles defined by only two points, in which case the center of the circle is the midpoint of the segment connecting them. We call these circles *Gabriel circles*. In this setting, circles containing no points of S in the interior constitute the so-called Gabriel graph. It is well known that the Gabriel graph is a subgraph of the Delaunay triangulation. The number of Gabriel circles not containing points of S in the interior is between $n-1$ and $3n-8$. Also higher order Gabriel graphs have been studied, where two vertices are connected by an edge in this graph if the Gabriel circle defined by the two points hast at most k points of S in its interior. The number of edges in this graph is known to be between $(k+1)n/2$ and $3(k+1)(n-2-k)$. We refer to the recent thesis [10] for more results and references.

Our problem can also be formulated in terms of angles. A point p lies inside the Gabriel circle defined

*Departamento de Matemáticas, Cinvestav-IPN, ruyfabila@math.cinvestav.edu.mx Partially supported by Conacyt of Mexico, Grant 153984.

†Departament de Matemàtica Aplicada IV, Universitat Politècnica de Catalunya, clemens@ma4.upc.edu Partially supported by projects MEC MTM2009-07242 and Gen. Cat. DGR 2009SGR1040 and ESF EUROCORES programme EuroGIGA, CRP ComPoSe: grant EUI-EURC-2011-4306, for Spain.

‡Departament de Matemàtica Aplicada IV, Universitat Politècnica de Catalunya, etramuns@ma4.upc.edu Partially supported by MTM2009-13060-C02-02 from Spanish MEC.

by two points a and b if and only if the angle apb at p is obtuse, that is, greater than $\pi/2$. Thus we are asking for the number of obtuse triangles in S. See [7] for some prominent problems and results on obtuse and acute angles in point sets.

Let Y denote the random variable that counts the number of points inside a Gabriel circle defined by two points chosen uniformly at random from S. Denote by $E_S(Y)$ the expectation of Y. Simple examples show that $E_S(Y)$ is not determined by the rectilinear crossing number $cr(S)$. We show in Section 3 that $E_S(Y)$ is at most $\frac{n-2}{3}$ and there are sets S where $E_S(Y)$ is exactly $\frac{n-2}{3}$. Concerning lower bounds, Conway et al. [5] proved that each set S of n points contains at least $\left\lceil \frac{n^2(n-3)}{18} \right\rceil$ obtuse and right triangles. This implies that $E_S(Y) \geq \frac{n}{9} - O(1)$ for every set S. Conway et al. also provide point sets S for which $E_S(Y)$ is no more than $\frac{4}{27}n + O(1)$.

2 The expected number of points in a circumcircle

Lemma 1 *Let S be a set of n points in general position in the plane. Then,*

$$\sum_{k=0}^{n-3} k \cdot c_k = \binom{n}{4} + cr(S),$$

where c_k is the number of circumcircles containing exactly k points in the interior.

Proof. In the following, the rectilinear crossing number $cr(S)$ is also denoted by $\square$, representing the number of four-tuples of S which are in convex position. Let $\triangle$ denote the number of four-tuples of S which are not in convex position. We have $\square + \triangle = \binom{n}{4}$. Four points define four circumcircles. If the four points are in convex position then exactly two of the four circles contain the remaining point in the interior. If the four points are not in convex position, then exactly one of the four circumcircles contains the remaining point in the interior. Let $\odot = \sum_{p \in S}$ (number of circumcircles containing p in its interior). We have $\odot = \odot_{\square} + \odot_{\triangle}$, where in $\odot_{\square}$ and $\odot_{\triangle}$ we distinguish whether the considered four points are in convex position or not. Then $\odot_{\square} = 2\square$ and $\odot_{\triangle} = \triangle$. We get

$$\sum_{k=0}^{n-3} k \cdot c_k = \odot = \odot_{\square} + \odot_{\triangle} = 2\square + \triangle = \binom{n}{4} + \square.$$

□

Theorem 2 *Let S be a set of n points in general position in the plane. Then, $E_S(X) = \frac{cr(S)}{\binom{n}{3}} + \frac{n-3}{4}$.*

Proof. The probability that $X = k$, for $0 \leq k \leq n-3$, is the probability that the circle defined by three randomly chosen points from S contains exactly k points in its interior. Since there are c_k circles containing k points in its interior, this probability is $\frac{c_k}{\binom{n}{3}}$. Hence,

$$E_S(X) = \sum_{k=0}^{n-3} k \cdot \frac{c_k}{\binom{n}{3}} = \frac{1}{\binom{n}{3}} \sum_{k=0}^{n-3} k \cdot c_k =$$

$$\frac{1}{\binom{n}{3}} \left(\binom{n}{4} + cr(S) \right) = \frac{cr(S)}{\binom{n}{3}} + \frac{n-3}{4}.$$

□

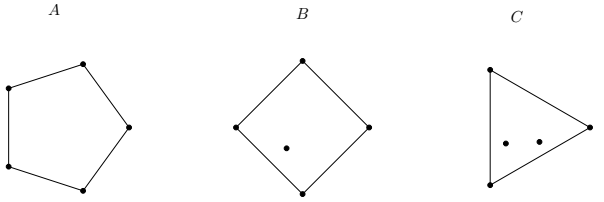


Figure 1: Three point sets representing the three different order types for 5 points.

Lemma 3 *Let S be a set of n points in general position in the plane. Then,*

$$\sum_{k=0}^{n-3} k^2 \cdot c_k = \binom{n}{5} + \binom{n}{4} + (n-3)cr(S).$$

Proof. We consider sets of five points. Figure 1 shows the three possible configurations corresponding to different order types [3, 8]: Configuration A is a convex five-gon, B is a convex quadrilateral with one interior point, and C is a triangle with two interior points. We first express the rectilinear crossing number of S in terms of $|A|, |B|$ and $|C|$, which denote the number of times the corresponding configuration appears in S. Configuration A defines five crossings, configuration B three crossings, and configuration C defines one crossing. When counting $cr(S)$ in this way by considering five-tuples, each crossing is counted $n-4$ times. We thus have

$$(n-4)cr(S) = 5|A| + 3|B| + |C|.$$

Since $|A| + |B| + |C| = \binom{n}{5}$, we get $(n-4)cr(S) = \binom{n}{5} + 4|A| + 2|B|$. We then calculate c_k, the number of circumcircles containing exactly k points, $0 \leq k \leq 2$, for each of the three configurations. Note that for sets of five points, c_0, c_1 and c_2 are determined by the number of extreme points; c_0 is the number of triangles of the Delaunay triangulation, $c_0 + c_2 = 6$ [4, 9], and $c_0 + c_1 + c_2 = \binom{5}{3}$. For configuration A we get $c_0(A) = 3, c_1(A) = 4$ and $c_2(A) = 3$. Configuration B

has $c_0(B) = 4, c_1(B) = 4$ and $c_2(B) = 2$. And configuration C has $c_0(C) = 5, c_1(C) = 4$ and $c_2(C) = 1$. We count $\sum \binom{k}{2} \cdot c_k$ by considering for each circumcircle all pairs of points in the interior of the circle. This sum also equals

$$\sum_{k=0}^{n-3} \binom{k}{2} \cdot c_k = c_2(A) + c_2(B) + c_2(C) =$$

$$3|A| + 2|B| + |C| = \binom{n}{5} + 2|A| + |B|.$$

Multiplying by two, we obtain

$$\sum_{k=0}^{n-3} k \cdot (k-1) \cdot c_k = \binom{n}{5} + \binom{n}{5} + 4|A| + 2|B|$$

and replacing the last terms using $(n-4)cr(S) = \binom{n}{5} + 4|A| + 2|B|$ we get

$$\sum_{k=0}^{n-3} k \cdot (k-1) \cdot c_k = \binom{n}{5} + (n-4)cr(S).$$

Using Lemma 1 we obtain

$$\sum_{k=0}^{n-3} k^2 \cdot c_k - \binom{n}{4} - cr(S) = \binom{n}{5} + (n-4)cr(S).$$

The result follows. □

Theorem 4 *Let S be a set of n points in general position in the plane. Then,*

$$V_S(X) = (n-3)^2 \left(\frac{cr(S)}{8\binom{n}{4}} - \frac{cr(S)^2}{16\binom{n}{4}^2} - \frac{1}{80} + \frac{1}{5(n-3)} \right).$$

Proof. The probability that $X^2 = k^2$, for $0 \le k \le n-3$, is the probability that $X = k$, which is $\frac{c_k}{\binom{n}{3}}$. Hence, by Lemma 3, $E_S(X^2) = \sum_{k=0}^{n-3} k^2 \cdot \frac{c_k}{\binom{n}{3}} =$

$$\frac{1}{\binom{n}{3}} \left(\binom{n}{5} + \binom{n}{4} + (n-3) \cdot cr(S) \right).$$

From Theorem 2 we get

$$E_S(X)^2 = \left(\frac{cr(S)}{\binom{n}{3}} + \frac{n-3}{4} \right)^2$$

We then use $V_S(X) = E_S(X^2) - E_S(X)^2$ and simplify to obtain the claimed expression.

□

3 The expected number of points in a Gabriel circle

Theorem 5 *For every set S of n points in general position in the plane, $E_S(Y) \le \frac{n-2}{3}$ and there are point sets attaining this bound.*

Proof. The probability that $Y = k$, for $0 \le k \le n-2$, is the probability that exactly k points lie in the Gabriel circle of two points chosen uniformly at random from S. Let e_k denote the number of segments of S for which the Gabriel circle defined by the endpoints of the segment contains exactly k points in the interior. Then, $E_S(Y) = \sum_{k=0}^{n-2} k \cdot \frac{e_k}{\binom{n}{2}}$. A point $p \in S$ lies in the interior of the Gabriel circle of two points a and b of S if and only if the angle apb at p is greater than $\frac{\pi}{2}$. We assign an obtuse angle apb at p to the edge ab. Since the interior angles of the triangle apb sum up to π, at most one of these three angles is greater than $\frac{\pi}{2}$. In total S spans $\binom{n}{3}$ triangles and thus at most this many obtuse angles. Hence, $\sum_{k=0}^{n-2} k \cdot e_k \le \binom{n}{3}$ and $E_S(Y) = \frac{1}{\binom{n}{2}} \sum_{k=0}^{n-2} k \cdot e_k \le \frac{n-2}{3}$.

For an example where this bound is obtained, consider the set S of points $i = (i, 0)$, $1 \le i \le n$, and perturb them slightly such that no three points are collinear and no four points are cocircular. Then each triangle abc with $a < b < c$ has an obtuse angle at b. In total we count $\binom{n}{3}$ obtuse triangles and the above reasoning gives $E_S(Y) = \frac{n-2}{3}$.

□

4 Concluding remarks and further research

Almost tight bounds on the minimum rectilinear crossing number allowed us to also obtain almost tight bounds on the mimimum value of $E_S(X)$ and of $V_S(X)$ among all sets S of n points. But also the other direction looks appealing: Can we derive a good bound on the rectilinear crossing number from the first moments of X? Then, we think it is interesting to express further problems in terms of crossing numbers. Concerning Gabriel circles, an open problem is to bound the variance $V_S(Y)$. Another challenging question is to reduce the gap on the lower bound on $E_S(Y)$.

Acknowledgements

This research was carried out while Ruy Fabila was visiting Departament de Matemàtica Aplicada IV of Universitat Politècnica de Catalunya, Castelldefels, July 2011.

References

[1] B.M. Ábrego, M. Cetina, S. Fernández-Merchant, J. Leaños, G. Salazar. *3-symmetric and 3-decomposable*

geometric drawings of K_n. Discrete Applied Mathematics 158, 1240–1258, 2010.

[2] B.M. Ábrego, M. Cetina, S. Fernández-Merchant, J. Leaños, G. Salazar. *On ($\leq k$)-edges, crossings, and halving lines of geometric drawings of K_n.* arXiv:1102.5065, 2011.

[3] O. Aichholzer, H. Krasser. *Abstract order type extension and new results on the rectilinear crossing number.* Computational Geometry: Theory and Applications 36, 2–15, 2006.

[4] F. Ardila. *The number of halving circles.* American Mathematical Monthly 111, 586–591, 2004.

[5] J.H. Conway, H.T. Croft, P. Erdős, M.J.T. Guy. *On the distribution of values of angles determined by coplanar points.* Journal of the London Mathematical Society (2) 19, 137–143, 1979.

[6] H. Edelsbrunner. Algorithms in Combinatorial Geometry. Springer, 1987.

[7] P. Erdős, Z. Füredi. *The greatest angle among n points in the d-dimensional Euclidean space.* Annals of Discrete Mathematics 17, 275–283, 1983.

[8] J.E. Goodman, R. Pollack. *Multidimensional Sorting.* SIAM Journal on Computing 12, 484–507, 1983.

[9] R. Lindenbergh. *A Voronoi poset.* Journal for Geometry and Graphics 7, 41-52, 2003.

[10] M. Saumell. *Some problems on proximity graphs.* PhD Thesis, Universitat Politècnica de Catalunya, 2011. http://kam.mff.cuni.cz/ saumell/PhDThesis.pdf, accessed December 2011.

[11] J. Urrutia. *A containment result on points and circles.* Preprint, February 2004. http://www.matem.unam.mx/urrutia/online_papers/ PointCirc2.pdf, accessed December 2011.

Many collinear k-tuples with no $k+1$ collinear points

József Solymosi* Miloš Stojaković†

Abstract

For every $k > 3$, we give a construction of planar point sets with many collinear k-tuples and no collinear $(k+1)$-tuples.

1 Introduction

In the early 60's Paul Erdős asked the following question about point-line incidences in the plane: *Is it possible that a planar point set contains many collinear four-tuples, but it contains no five points on a line?* There are various constructions for n-element point sets with $n^2/6 - O(n)$ collinear triples with no four on a line (see [3] or [10]). However, no similar construction is known for larger numbers.

Let us formulate Erdős' problem more precisely. For a finite set P of points in the plane and $k \geq 2$, let $t_k(P)$ be the number of lines meeting P in exactly k points, and let $T_k(P) := \sum_{k' \geq k} t_{k'}(P)$ be the number of lines meeting P in at least k points. For $r > k$ and n, we define

$$t_k^{(r)}(n) := \max_{\substack{|P|=n \\ T_r(P)=0}} t_k(P).$$

In plain words, $t_k^{(r)}(n)$ is the number of lines containing exactly k points from P, maximized over all n point sets P that do not contain r collinear points. In this paper we are concerned about bounding $t_k^{(k+1)}(n)$ from below for $k > 3$. Erdős conjectured that $t_k^{(k+1)}(n) = o(n^2)$ for $k > 3$ and offered \$100 for a proof or disproof [8] (the conjecture is listed as Conjecture 12 in the problem collection of Brass, Moser, and Pach [2]).

1.1 Earlier Results

This problem was one of Erdős' favorite geometric problems, he frequently talked about it and listed it among the open problems in geometry, see [8, 7, 5, 6, 9]. It is not just a simple puzzle which might be hard to solve, it is related to some deep and difficult problems in other fields. It seems that the key to attack this question would be to understand the group structure behind point sets with many collinear triples. We will not investigate this direction in the present paper, our goal is to give a construction showing that Erdős conjecture, if true, is sharp – for $k > 3$, one can not replace the exponent 2 by $2 - c$, for any $c > 0$.

The first result was due to Kárteszi [13] who proved that $t_k^{(k+1)}(n) \geq c_k n \log n$ for all $k > 3$. In 1976 Grünbaum [11] showed that $t_k^{(k+1)}(n) \geq c_k n^{1+1/(k-2)}$. For some 30 years this was the best bound when Ismailescu [12], Brass [1], and Elkies [4] consecutively improved Grünbaum's bound for $k \geq 5$. However, similarly to Grünbaum's bound, the exponent was going to 1 as k went to infinity.

In what follows we are going to give a construction to show that for any $k > 3$ and $\delta > 0$ there is a threshold $n_0 = n_0(k, \delta)$ such that if $n \geq n_0$ then $t_k^{(k+1)}(n) \geq n^{2-\delta}$. On top of that, we note that each of the collinear k-tuples that we count in our construction has an additional property – the distance between every two consecutive points is the same.

1.2 Notation

For $r > 0$, a positive integer d and $x \in \mathbb{R}^d$, by $B_d(x, r)$ we denote the closed ball in $\mathbb{R}^d$ of radius r centered at x, and by $S_d(x, r)$ we denote the sphere in $\mathbb{R}^d$ of radius r centered at x. When $x = 0$, we will occasionally write just $B_d(r)$ and $S_d(r)$.

For a set $S \subseteq \mathbb{R}^d$, let $N(S)$ denote the number of points from the integer lattice $\mathbb{Z}^d$ that belong to S, i.e., $N(S) := \mathbb{Z}^d \cap S$.

2 A lower bound for $t_k^{(k+1)}(n)$

We will prove bounds for even and odd value of k separately, as the odd case needs a bit more attention.

2.1 k is even

Theorem 1 *For $k \geq 4$ even and $\varepsilon > 0$, there is a positive integer n_0 such that for $n > n_0$ we have $t_k^{(k+1)}(n) > n^{2-\varepsilon}$.*

Proof. We will give a construction of a point set P containing no $k+1$ collinear points, with a high value of $t_k(P)$.

*Department of Mathematics, University of British Columbia, Vancouver, Canada, solymosi@math.ubc.ca

†Department of Mathematics and Informatics, University of Novi Sad, Serbia. milos.stojakovic@dmi.uns.ac.rs. Partly supported by Ministry of Education and Science, Republic of Serbia, and Provincial Secretariat for Science, Province of Vojvodina.

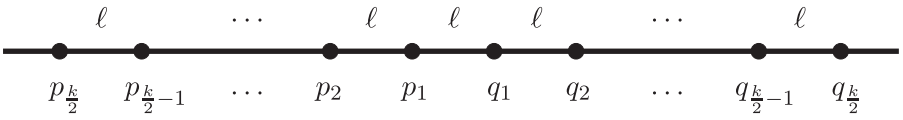


Figure 1: Line s with k points, for k even.

Let d be a positive integer, and let $r_0 > 0$. It is known, see, e.g., [15], that for large enough r_0 we have

$$\begin{aligned} N(B_d(r_0)) &= (1+o(1))V(B_d(r_0)) \\ &= (1+o(1))C_d r_0^d \\ &\geq c_1 r_0^d, \end{aligned}$$

where $C_d = \frac{\pi^{d/2}}{\Gamma((n+2)/2)}$, and $c_1 = c_1(d)$ is a constant depending only on d.

For each integer point from $B_d(r_0)$, the square of its distance to the origin is at most r_0^2. As the square of that distance is an integer, we can apply pigeonhole principle to conclude that there exists r, with $0 < r \leq r_0$, such that the sphere $S_d(r)$ contains at least $1/r_0^2$ fraction of points from $B_d(r_0)$, i.e.,

$$N(S_d(r)) \geq \frac{1}{r_0^2} N(B_d(r_0)) \geq \frac{1}{r_0^2} c_1 r_0^d = c_1 r_0^{d-2}.$$

We now look at unordered pairs of different points from $\mathbb{Z}^d \cap S_d(r)$. The total number of such pairs is at least

$$\binom{N(S_d(r))}{2} \geq \binom{c_1 r_0^{d-2}}{2} \geq c_2 r_0^{2d-4},$$

for some constant $c_2 = c_2(d)$. On the other hand, for every $p, q \in \mathbb{Z}^d \cap S_d(r)$ we know that the Euclidean distance $d(p,q)$ between p and q is at most $2r$, and that the square of that distance is an integer. Hence, there are at most $4r^2$ different possible values for $d(p,q)$. Applying pigeonhole principle again, we get that there are at least

$$\frac{c_2 r_0^{2d-4}}{4r^2} \geq \frac{c_2}{4} r_0^{2d-6}$$

pairs of points from $\mathbb{Z}^d \cap S_d(r)$ that all have the same distance. We denote that distance by ℓ.

Let $p_1, q_1 \in \mathbb{Z}^d \cap S_d(r)$ with $d(p_1, q_1) = \ell$, and let s be the line going through p_1 and q_1. We define $k-2$ points $p_2, \ldots, p_{k/2}$, $q_2, \ldots, q_{k/2}$ on the line s such that $d(p_i, p_{i+1}) = \ell$ and $d(q_i, q_{i+1}) = \ell$, for all $1 \leq i < k/2$, and such that all k points $p_1, \ldots, p_{k/2}$, $q_1, \ldots, q_{k/2}$ are different, see Figure 1.

Knowing that p_1 and q_1 are points from $\mathbb{Z}^d$, the way we defined points $p_2, \ldots, p_{k/2}, q_2, \ldots, q_{k/2}$ implies that they have to be in $\mathbb{Z}^d$ as well. If we set $r_i := \sqrt{r^2 + i(i-1)\ell^2}$, for all $i = 1, \ldots, k/2$, then the points p_i and q_i belong to the sphere $S_d(r_i)$, and hence, $p_i, q_i \in \mathbb{Z}^d \cap S_d(r_i)$, for all $i = 1, \ldots, k/2$, see Figure 2.

We define the point set P to be the set of all integer points on spheres $S_d(r_i)$, for all $i = 1, \ldots, k/2$, i.e.,

$$P := \mathbb{Z}^d \cap \left(\cup_{i=1}^{k/2} S_d(r_i) \right).$$

Let $n := |P|$. Obviously, $P \subseteq B_d(r_{k/2})$, so we have

$$n \leq N\left(B_d(r_{k/2})\right) = (1+o(1))V(B_d(r_{k/2})) = c_1 r_{k/2}^d.$$

Plugging in the value of $r_{k/2}$ and having in mind that $\ell < 2r$, we obtain

$$\begin{aligned} n &\leq c_1 \left(\sqrt{r^2 + k/2(k/2-1)4r^2} \right)^d \\ &\leq c_1 \sqrt{k^2+1}^d r^d \\ &\leq c_3 r^d \\ &\leq c_3 r_0^d, \end{aligned}$$

where $c_3 = c_3(d,k)$ is a constant depending only on d and k.

As the point set P is contained in the union of $k/2$ spheres, there are obviously no $k+1$ collinear points in P. On the other hand, every pair of points $p_1, q_1 \in \mathbb{Z}^d \cap S_d(r)$ with $d(p_1, q_1) = \ell$ defines one line that contains k points from P. Hence, the number of lines containing exactly k points from P is

$$t_k(P) \geq \frac{c_2}{4} r_0^{2d-6} \geq \frac{c_2}{4} \frac{1}{c_3^{\frac{2d-6}{d}}} n^{\frac{2d-6}{d}} \geq c_4 n^{\frac{2d-6}{d}},$$

where $c_4 = c_4(d,k)$ is a constant depending only on d and k.

To obtain a point set in two dimensions, we can project our d dimensional point set to an arbitrary (two dimensional) plane in $\mathbb{R}^d$. The vector v along which we project should be chosen so that every two points from our point set are mapped to different points, and every three points that are not collinear are mapped to points that are still not collinear. Obviously, such vector can be found.

For ε given, we can pick d such that $\frac{2d-6}{d} > 2 - \varepsilon$. As we increase r_0, we obtain constructions with n growing to infinity. When n is large enough, the statement of the theorem will hold. □

2.2 k is odd

Theorem 2 *For $k \geq 4$ odd and $\varepsilon > 0$, there is a positive integer n_0 such that for $n > n_0$ we have $t_k^{(k+1)}(n) > n^{2-\varepsilon}$.*

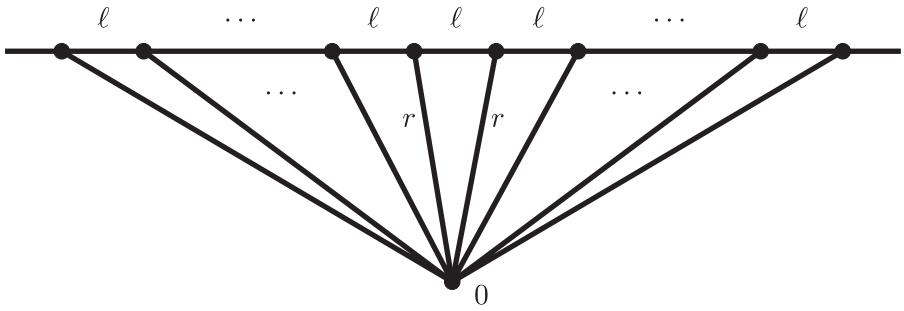


Figure 2: The position of the k points related to the origin, for k even.

The proof for k odd follows the lines of the proof for k even, but requires an additional twist, as we cannot obtain all the points in pairs as before. What we do, roughly speaking, is find the k-tuple of collinear points on $\frac{k+1}{2}$ selected spheres, where the inner most sphere will contain just one point of the k-tuple and each of the other spheres will contain two. Then we make an additional restriction on the point set on the inner most sphere to prevent $k+1$ collinear points from appearing in our construction.

As due to space restrictions we cannot present this proof in full detail, we omit it altogether. It can be found in [14].

Acknowledgments

The results presented in this paper are obtained during the authors' participation in 8th Gremo Workshop on Open Problems – GWOP 2010. We thank the organizers for inviting us to the workshop and providing us with a gratifying working environment. Also, we are grateful for the inspiring conversations with the members of the group of Professor Emo Welzl.

References

[1] Peter Brass. On point sets without k collinear points. In *Discrete geometry*, volume 253 of *Monogr. Textbooks Pure Appl. Math.*, pages 185–192. Dekker, New York, 2003.

[2] Peter Brass, William Moser, and János Pach. *Research problems in discrete geometry*. Springer, New York, 2005.

[3] Stefan A. Burr, Branko Grünbaum, and N. J. A. Sloane. The orchard problem. *Geometriae Dedicata*, 2:397–424, 1974.

[4] Noam D. Elkies. On some points-and-lines problems and configurations. *Period. Math. Hungar.*, 53(1-2):133–148, 2006.

[5] P. Erdős. Some combinational problems in geometry. In *Geometry and differential geometry (Proc. Conf., Univ. Haifa, Haifa, 1979)*, volume 792 of *Lecture Notes in Math.*, pages 46–53. Springer, Berlin, 1980.

[6] Pál Erdős. Néhány elemi geometriai problémáról (on some problems in elementary geometry, in hungarian). *Középiskolai Matematikai Lapok*, 61:49–54, 1980.

[7] Paul Erdős. On some problems of elementary and combinatorial geometry. *Ann. Mat. Pura Appl. (4)*, 103:99–108, 1975.

[8] Paul Erdős. Research problems. *Periodica Mathematica Hungarica*, 15:101–103, 1984.

[9] Paul Erdős and George Purdy. *Some extremal problems in geometry. IV*. Utilitas Math., Winnipeg, Man., 1976.

[10] Z. Füredi and I. Palásti. Arrangements of lines with a large number of triangles. *Proc. Amer. Math. Soc.*, 92(4):561–566, 1984.

[11] Branko Grünbaum. New views on some old questions of combinatorial geometry. In *Colloquio Internazionale sulle Teorie Combinatorie (Rome, 1973), Tomo I*, pages 451–468. Atti dei Convegni Lincei, No. 17. Accad. Naz. Lincei, Rome, 1976.

[12] Dan Ismailescu. Restricted point configurations with many collinear k-tuplets. *Discrete Comput. Geom.*, 28(4):571–575, 2002. Discrete and computational geometry and graph drawing (Columbia, SC, 2001).

[13] Ferenc Kárteszi. Sylvester egy tételéről és Erdős egy sejtéséről. *Középiskolai Matematikai Lapok*, 26:3–10, 1963.

[14] József Solymosi and Miloš Stojaković. Many collinear k-tuples with no $k+1$ collinear points (full paper). *arXiv:1107.0327v1 [math.CO]*, 2011.

[15] Arnold Walfisz. *Gitterpunkte in mehrdimensionalen Kugeln*. Monografie Matematyczne. Vol. 33. Państwowe Wydawnictwo Naukowe, Warsaw, 1957.

Topology-Preserving Watermarking of Vector Data

Stefan Huber* Martin Held* Roland Kwitt† Peter Meerwald*

Abstract

The embedding of a digital watermark in vector data results in a perturbation of the vertices which needs to be constrained in order to maintain geometric properties of the data. In this paper we investigate the problem of computing so-called perturbation regions in which the vertices of a planar straight-line graph may be dislocated while still preserving the topology of the input. We propose two different algorithms to solve this problem and discuss how they can form the geometric part of a watermarking framework.

1 Introduction

A standard way for watermarking digital data is to embed imperceptible, yet detectable, information into a digital signal. Since the watermark is only known to the embedder, copyright holders can use this technology to mark their content in order to prove ownership by being able to detect the embedded signal.

Most watermarking research has been directed towards techniques applicable to raster data and audio content. However, complex vector data in computer-aided design as well as maps and infrastructure data in geographic information systems, for instance, constitute equally valuable digital assets. When watermarking vector data statistical features are imposed by dislocating vertices. Hence, novel geometric requirements are introduced, while still demanding imperceptibility of the watermark and robustness against different kinds of attacks. For example, one needs to guarantee that watermarking does not introduce overlaps among rivers and streets in a map.

Surprisingly, copyright protection of vector data with distortion constraints has not received much attention. Doncel et al. [2] consider multiple polygonal chains sharing common vertices, such as the border of neighboring countries, and discuss how to preserve connectivity after watermarking. In [6, 7], methods are proposed that avoid perturbing certain vertices to preserve the visual appearance of the original data.

In [4] we presented a watermarking framework for planar straight-line graphs which consists of a geometric part, a watermarking part and a final correction step, cf. Fig. 1. In this paper we focus on the preservation of the input topology, i.e., on ensuring that

[T1] the numbers of vertices and edges
[T2] all containment relations
[T3] all incidence orders at vertices

remain unchanged, and that

[T4] no intersections are introduced.

We discuss two algorithms for computing a so-called *maximum perturbation region* (MPR) for every vertex such that T1–T4 can be guaranteed: If all vertices stay within their MPRs after watermarking then the input topology is guaranteed to be preserved. Once the MPRs are known, we embed the watermark into the input data. Since this process need not respect T2–T4, in a final correction step we use the MPRs to correct the output of the watermarking step in order to guarantee T1–T4. While the basic idea for preserving the input topology is simple, the computational challenge is to efficiently compute MPRs that are large enough such that the correction step does not destroy the watermark embedding. Our approach is general enough to work with any watermarking method that does not insert or remove vertices or edges of the input.

2 Maximum perturbation regions

Consider a planar straight-line graph $G = (V, E)$ with vertex set $V := \{v_1, \ldots, v_n\}$ and edge set E. The process of embedding a watermark in G can be interpreted as a transformation of the vertex set V to a *perturbed* vertex set $V' = \{v'_1, \ldots, v'_n\}$. We denote the graph given by the perturbed vertex set by $G' = (V', E')$ and by E' the corresponding edge set of G'. We seek *maximum perturbation regions* (MPRs) as regions $R_1, \ldots, R_n$, with $v_i \in R_i \subset \mathbb{R}^2$, such that the following property holds: If $v'_i \in R_i$ for all $1 \leq i \leq n$ then T1–T4 hold for G'. Thus, $R_1, \ldots, R_n$ are regions in which it is safe to perform perturbations without violating the input topology.

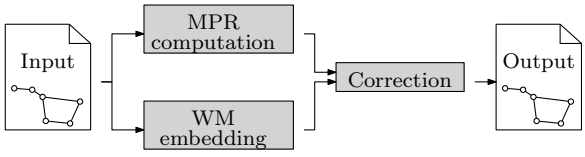


Figure 1: Our watermarking framework.

*Universität Salzburg, FB Computerwissenschaften, A-5020 Salzburg, {shuber,held,pmeerw}@cosy.sbg.ac.at. Work was supported by Austrian FWF Grant L367-N15.

†Kitware Inc., NC, USA, {roland.kwitt@kitware.com}

2.1 Computing MPRs using Voronoi diagrams

For the Voronoi-based computation of MPRs we consider the vertices of V and the straight-line segments of E as the set S of input sites. The Voronoi diagram $\mathcal{VD}(S)$ of S decomposes the plane into Voronoi cells $\mathcal{VC}(s, S)$, with $s \in S$. Our MPR algorithm relies on the following observation: If the perturbed counterpart $e' \in E'$ of an edge $e \in E$ does not intersect the Voronoi cells of edges non-adjacent to e, then G' is planar as the Voronoi cells of different sites do not overlap.

We denote by $|e|$ the length of the line segment e and define for any $r > 0$ and any line segment e the set $B(e, r) \subset \mathbb{R}^2$ as the rectangle with width $2r$ and length $|e|$ that has e as its center-line. Next, we denote by $D_v(r)$ the open disk with center v and radius r. Our MPR-algorithm consists of two phases:

Phase 1: For each vertex $v_i \in V$, we consider the incident edges $e^i_1, \ldots, e^i_{d_i}$ and we denote by $\hat{e}^i_j$ the half of e^i_j that is incident to v_i. Then we compute for each vertex $v_i \in V$ the largest $t_i \leq \min_{1 \leq j \leq d_i} |\hat{e}^i_j|$ such that

$$D_{v_i}(t_i) \cup \bigcup_{j=1}^{d_i} B(\hat{e}^i_j, t_i) \subseteq \mathcal{VC}(v_i, S) \cup \bigcup_{j=1}^{d_i} \mathcal{VC}(e^i_j, S). \quad (1)$$

Let $T(v_i) := D_{v_i}(t_i) \cup \bigcup_{j=1}^{d_i} B(\hat{e}^i_j, t_i)$, cf. Fig. 2. In order to compute $t_1, \ldots, t_n$ we first cut the Voronoi cell $\mathcal{VC}(e, S)$ of every edge $e \in E$, with $e = v_i v_j$, into two halves along the bisector of v_i and v_j and insert the two points of intersection as Voronoi nodes. Every parabolic Voronoi edge is also split by a node at its apex. Then we traverse all halved Voronoi cells and compute the (non-zero) distances of the nodes of the halved Voronoi cells to their input sites.

Lemma 1 *The interiors of $T(v_i)$ and $T(v_j)$ do not overlap for different v_i and v_j.*

Phase 2: For every vertex v_i we consider its adjacent vertices $v^i_1, \ldots, v^i_{d_i}$ and compute the value

$$r_i := \min\{t_{v_i}, t_{v^i_1}, \ldots, t_{v^i_{d_i}}\}. \quad (2)$$

Then we define the maximum perturbation region R_i of the vertex v_i as

$$R_i := D_{v_i}(r_i). \quad (3)$$

In Fig. 3, we illustrate all MPRs as dashed circles. For an edge $e \in E$ we call the area that is bounded by the MPRs of the incident vertices v_i, v_j of e and their bitangents the *hose* H_e around e. In fact, H_e represents the valid area in which the perturbed edge $v'_i v'_j$ may lie. All hoses are shown as shaded areas in Fig. 3. In order to avoid hoses that touch we can simply multiply all radii r_i by $1-\epsilon$, for a small positive constant $\epsilon < 1$.

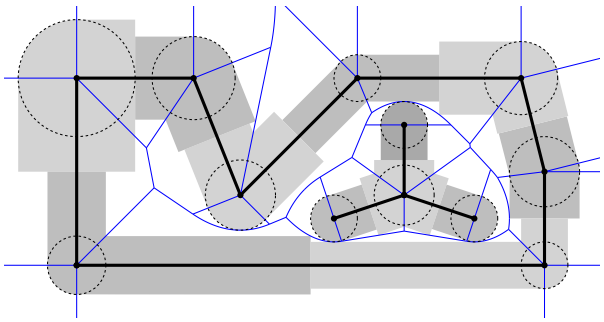


Figure 2: Phase 1: The regions $T(v_i)$ for vertices v_i are shaded in gray levels. The circles $D_{v_i}(t_i)$ are shown dashed.

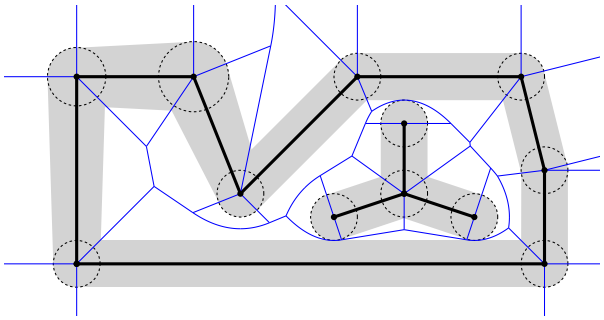


Figure 3: Phase 2: The MPRs are shown dashed. The union $\bigcup_{e \in E} H(e)$ of all hoses is shaded.

Lemma 2 *A hose $H(e)$ of an edge $e = v_i v_j$ is contained in $T(v_i) \cup T(v_j)$.*

Corollary 3 *$H(e_i)$ and $H(e_j)$ do not overlap if e_i and e_j have no common vertex.*

Theorem 4 *If the perturbation v'_i of the vertex $v_i \in V$ is constrained to R_i as defined in (3), for $1 \leq i \leq n$, then T1–T4 are guaranteed for G'.*

Proof. Since no vertices or edges are added or removed, T1 is met. The hoses of one connected component of G are separated from all other hoses by the Voronoi diagram, thus enforcing T2. Similarly, due to the Voronoi diagram, the cyclic order of the hoses of edges incident upon a vertex is identical to the order of those edges and, therefore, T3 is guaranteed. Finally, Cor. 3 ensures T4. □

The Voronoi-based approach assigns the MPRs in a *fair* manner in the following sense: If two hoses touch each other then they touch at the apex of a parabolic Voronoi edge. Hence, both hoses are equally wide.

This approach does not necessarily determine the largest possible MPRs, though: Phase 2 is easy to implement but rather conservative. (For example, the MPR of the vertex in the upper left-hand corner of Fig. 3 could be chosen larger.) In general, we could increase individual MPRs in an additional third phase

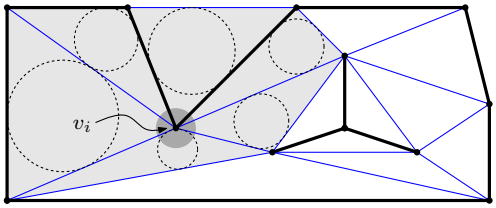


Figure 4: The triangulation-based approach: R_i is defined by triangles incident to v_i.

as long as Lemma 2 remains valid. We did not study this strategy in more detail, though.

We can compute all $t_1, \ldots, t_n$ in $O(n)$ time as the Voronoi diagram $\mathcal{VD}(S)$ is of linear size. The computation of the Voronoi diagram itself takes $O(n \log n)$, which leads to a total runtime of $O(n \log n)$ for the Voronoi-based MPR-algorithm. We implemented this approach in C++, based on the Voronoi package VRONI [3] to compute the Voronoi diagram. Computing all MPRs on a dataset with 60 000 vertices takes about a second on a mid-range personal computer.

The Voronoi-based approach also permits generalisations to input elements more general than straight-line segments. For example, one could consider circular arcs, which can also be processed by VRONI.

2.2 Computing MPRs using triangulations

Our second approach to compute MPRs is based on triangulations. The algorithm starts with a constrained triangulation T of the convex hull of G, with E forming the line segments of the constraints. Hence, G can be interpreted as a subgraph of T, see Fig. 4.

Let us denote by T' the graph that results from T by replacing V by V'. The key observation in the triangulation-based approach is the following: If dislocating the vertices violates T2–T4 then at least one triangle in the triangulation has changed its orientation. Hence, the objective is to restrict the perturbations of the vertices to MPRs such that no orientation of a triangle changes.

Consider a triangle Δ of T and denote by $I(\Delta)$ the radius of its incircle. If the vertices of Δ are dislocated by a distance of less than $I(\Delta)$, then the orientation of Δ remains the same. Hence, we compute the MPR R_i of the vertex $v_i \in V$ as

$$R_i := D_{v_i}(r_i), \tag{4}$$

where $r_i := \min_{1 \le j \le D_i} I(\Delta^i_j)$, with $\Delta^i_1, \ldots, \Delta^i_{D_i}$ denoting all triangles incident to v_i. Since for each triangle Δ in T it holds that the MPRs of its vertices are disks with radii at most $I(\Delta)$, we get that all triangles in T preserve their orientations.

Theorem 5 *If the perturbation v'_i of the vertex $v_i \in V$ is constrained to R_i as defined in (4), for $1 \le i \le n$, then T1–T4 are guaranteed for G'.*

Proof. Consider the Voronoi diagram of T: all Voronoi nodes coincide with the centers of the incircles of the triangles of T, and every Voronoi cell of an interior edge $e = v_iv_j$ is given by two triangles formed by v_i, v_j and one of the incircle centers of the two triangles of T that have e as an edge. Hence, the triangulation-based approach can be interpreted as a slightly modified variant of the Voronoi-based approach applied to the edges of T, allowing us to confer the correctness result of Thm. 4. □

We note that this algorithm works with any constrained triangulation T of G. However, in order to permit perturbations to robustly embed the watermark, we seek MPRs that are as large as possible. Hence, the question arises which triangulations of G have large incircles and subsequently large MPRs. Obviously, among all triangles with fixed circumcircle, the equilateral triangle has the largest incircle. This observation motivated us to employ the (constrained) Delaunay triangulation for our actual implementation of the above algorithm.

Guaranteed-quality triangulations that use Steiner vertices in order to guarantee a lower bound for the smallest angle in a triangulation are known, see, e.g., [8]. However, maximizing the smallest incircle and maximizing the smallest inner angle of a triangle are not the same objectives. For example, a skinny triangle may be fine for us if it is just large enough and, in further consequence, contains still a large incircle.

To the best of our knowledge, maximizing the smallest incircle did not attract attention so far. Note that successively adding Steiner vertices may increase the smallest angle, but, at some point, certainly does not enlarge the smallest incircle any further.

We implemented the following heuristic approach in order to enlarge the incircles: After computing the triangulation of G, we determine a list of triangles Δ, where $I(\Delta)$ is by a factor of $f > 1$ larger when compared to its three neighboring triangles and insert Steiner points at the centers of their incircles. In practice, we observed that using $f = 1.5$ increases the average incircle radius by a few percent. However, applying this refinement step multiple times did not lead to a further improvement in our experiments.

The constrained Delaunay triangulation can be computed in $O(n \log n)$ time. Computing $r_1, \ldots, r_n$ can be done in $O(n)$ time as a triangulation contains $O(n)$ triangles. Hence, we end up with a total time complexity of $O(n \log n)$. For our C++ implementation of the triangulation-based algorithm we used the GNU Triangulated Surface (GTS) library [5], which implements a semi-dynamic algorithm. Computing all MPRs on a dataset with 60 000 vertices and running one refinement step with $f = 1.5$ takes about a second on a mid-range computer.

The triangulation-based method can also be ex-

tended to $\mathbb{R}^3$, if we want to embed a watermark on a polyhedron P with n vertices. Chazelle and Palios [1] showed that there exists a tetrahedralization of P that employs up to $O(n + r^2)$ Steiner vertices, where r denotes the number of reflex edges of P, and that it can be computed in $O(nr + r^2 \log r)$ time. The radius of the MPR of a vertex v of P is then defined as the minimum of the radii among the inscribed spheres of the tetrahedra incident to v.

3 Application to watermarking

We conclude with an outline of a watermarking scheme that was proposed in the context of vector data, cf. [2, 4]. The common basis of many watermarking approaches is to transform the input in a way such that conventional strategies can be applied. For this purpose, we regard all vertices V of G as points in the complex plane $\mathbb{C}$. To foster embedding a long, and consequently more robust, watermark sequence we only consider chains of G of a certain length as our input data and then perform a Discete Fourier Transform (DFT). This gives us a frequency-domain representation of the chains in the form of a set of Fourier coefficients. A scaled bipolar {+1,-1} random sequence (generated using the secret seed K belonging to the copyright holder) is then added to the DFT magnitudes and the inverse DFT is computed to finally reclaim the points in $\mathbb{C}$ and, consequently, the perturbed vertex set V'.

If all vertices perturbed by the watermarking stay within their MPRs then nothing needs to be done. Otherwise, if a vertex is outside of its pre-computed MPR, we can choose between two correction methods which both aim at preserving most of the watermark signal: (i) project each vertex onto the boundary of its corresponding MPR disk, and (ii) only perform the correction for the vertices of edges that actually intersect. Method (i) runs in $O(n)$ time, while the conditional correction (ii) can be done in $O(kn)$ time, where $k \in O(n)$ denotes the number of edges of G which have at least one vertex not in its MPR. (Very recent results suggest that $O(n \log n + m)$ time is achievable for (ii), where m denotes the number of intersections among $E \cup E'$.)

This trade-off between time and strength of the watermark needs to be evaluated in the context of the desired application. Imposing the topology constraints T2–T4 on the watermarked data dampens the watermark. To get an impression of detection performance for both strategies, we embedded a watermark in a vector graphic of roughly 24000 vertices and only considered chains of length > 200. With a modest watermark embedding strength, this led to ≈ 1600 vertices subject to MPR correction. Using correction strategy (i), the probability of missing the watermark is $\approx 10^{-20}$, while strategy (ii) yields $\approx 10^{-60}$.

4 Discussion

Our experiments with both MPR computations allow the following conclusions: First, using a Voronoi tessellation yields larger MPR regions and consequently gives more freedom for the watermarking process. Further, it can be extended to more general input, such as circular arcs for instance. Using triangulations, on the other hand, leads to potentially smaller MPRs and less freedom for watermarking. However, in consideration of a potential 3-D extension, one might favor this approach due to the simpler implementation.

Our work also reveals two interesting problems for future research: How can we compute a triangulation such that the radii of the incircles are maximized? And, more generally, how can me make the watermarking respect other important geometric properties of vector data, such as right angles or parallelism?

References

[1] B. Chazelle and L. Palios. Triangulating a Nonconvex Polytope. *Discrete Comput. Geom.*, 5:505–526, 1990.

[2] V. Doncel, N. Nikolaidis, and I. Pitas. An Optimal Detector Structure for the Fourier Descriptor Domain Watermarking of 2D Vector Graphics. *IEEE Trans. Visualizat. Comput. Graph.*, 13(5):851–863, Sept. 2007.

[3] M. Held and S. Huber. Topology-Oriented Incremental Computation of Voronoi Diagrams of Circular Arcs and Straight-Line Segments. *Comput. Aided Design*, 41(5):327–338, May 2009.

[4] S. Huber, R. Kwitt, P. Meerwald, M. Held, and A. Uhl. Watermarking of 2D Vector Graphics with Distortion Constraint. In *IEEE Int. Conf. on Multimedia & Expo (ICME 2010)*, pages 480–485, Singapore, July 2010.

[5] S. Popinet. GTS – The GNU Triangulated Surface Library. Online available from `http://gts.sourceforge.net/`.

[6] Y.-C. Pu, W.-C. Du, and I.-C. Jou. Perceptually Transparent Polyline Watermarking Based on Normal Multi-Resolution Representation. *IEICE Trans. Information and Systems*, E89-D(12):2939–2949, Digital Press 2006.

[7] C. Y. Shao, H. L. Wang, X. M. Niu, and X. T. Wang. Shape-Preserving Algorithm for Watermarking 2-D Vector Map Data. In *Proc. 7th IEEE Workshop on Multimedia Signal Proc.*, pages 1–4, Shanghai, China, Oct. 2005.

[8] J. Shewchuk. Delaunay Refinement Algorithms for Triangular Mesh Generation. *Comput. Geom. Theory and Appl.*, 22(1-3):21–74, May 2002.

Locally Correct Fréchet Matchings

Kevin Buchin* Maike Buchin* Wouter Meulemans* Bettina Speckmann*

Abstract

The Fréchet distance is a metric to compare two curves, which is based on monotonous matchings between these curves. We call a matching that results in the Fréchet distance a Fréchet matching. There are often many different Fréchet matchings and not all of these capture the similarity between the curves well. We propose to restrict the set of Fréchet matchings to "natural" matchings and to this end introduce *locally correct* Fréchet matchings. We prove that at least one such a matching exists for two polygonal curves and give an algorithm to compute it.

1 Introduction

Many problems ask for the comparison of two curves. Consequently, several distance measures have been proposed for the similarity of two curves P and Q, for example, the Hausdorff and the Fréchet distance. Such a distance measure simply returns a number indicating the (dis)similarity. However, the Hausdorff and the Fréchet distance are both based on matchings of the points on the curves. The distance returned is the maximum distance between any two matched points. The Fréchet distance uses monotonous matchings (and limits hereof): if point p on P and q on Q are matched, then any point on P after p must be matched to q or a point on Q after q. The Fréchet distance is the maximal distance between two matched points minimized over all monotonous matchings of the curves. We call a matching resulting in the Fréchet distance a Fréchet matching.

There are often many different Fréchet matchings for two curves. However, as the Fréchet distance is determined only by the maximal distance, not all of these matchings capture the similarity between the curves well (see Fig. 1). There are applications that directly use a matching, for example, to compute the average distance [5] or to morph between the curves [3]. In such situations it is particularly important to be able to compute a "good" matching.

*Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands. k.a.buchin@tue.nl, m.e.buchin@tue.nl, w.meulemans@tue.nl, and speckman@win.tue.nl. M. Buchin is supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 612.001.106. W. Meulemans and B. Speckmann are supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 639.022.707.

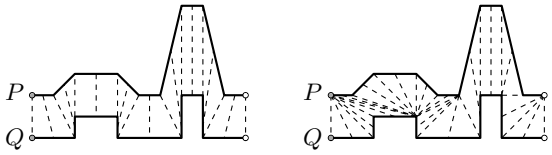


Figure 1: Two Fréchet matchings for curves P and Q.

Results. We propose to restrict the set of Fréchet matchings to "natural" matchings and to this end introduce *locally correct* Fréchet matchings: matchings that are Fréchet matchings for any two matched subcurves. We prove the following theorem.

Theorem 1 *For any two polygonal curves P and Q, there exists a locally correct Fréchet matching.*

This theorem follows directly from Lemma 3 in Section 3. The proof of this lemma results in a recursive algorithm to compute a locally correct matching.

Related work. The first algorithm to compute the Fréchet distance was given by Alt and Godau [1]. Since then, the Fréchet distance has received significant attention. Here we focus on approaches that restrict the allowed matchings. Efrat *et al.* [3] introduced Fréchet-like metrics, the geodesic width and link width, to restrict to matchings suitable for curve morphing. Their method is suitable only for non-intersecting polylines. Moreover, geodesic width and link width do not resolve the problem illustrated in Fig. 1: both matchings also have minimal geodesic width and minimal link width. Maheshwari *et al.* [4] studied a restriction by "speed limits", which may exclude all Fréchet matchings and may cause undesirable effects near "outliers" (see Fig. 2). Buchin *et al.* [2] describe a framework for restricting Fréchet matchings, which they illustrate by restricting slope and path length. The former corresponds to speed limits. We briefly discuss the latter in Section 3.

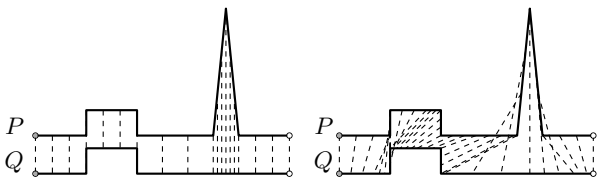


Figure 2: Two Fréchet matchings. The right one results from speed limits and is not locally correct.

2 Preliminaries

Curves. Let C be a polygonal curve with n edges, defined by vertices $c_0, \ldots, c_n$. We treat a curve as a continuous map $C : [0, n] \to \mathbb{R}^2$. In this map, $C(i)$ equals c_i for integer i. Furthermore, $C(i + \lambda)$ equals $(1 - \lambda) \cdot c_i + \lambda \cdot c_{i+1}$, for integer i and $0 < \lambda < 1$. As a reparametrization $\sigma : [0, 1] \to [0, n]$ of a curve C, we allow any continuous, non-decreasing function such that $\sigma(0) = 0$ and $\sigma(1) = n$. We denote by $C_\sigma(t)$ the actual location according to reparametrization σ: $C_\sigma(t) = C(\sigma(t))$. By $C_\sigma[a, b]$ we denote the subcurve of C in between $C_\sigma(a)$ and $C_\sigma(b)$.

Fréchet matchings. We are given two polygonal curves P and Q with m and n edges. A (monotonous) *matching* μ between P and Q is a pair of reparametrizations (σ, θ), such that $P_\sigma(t)$ matches to $Q_\theta(t)$. The Euclidean distance between two matched points is denoted by $d_\mu(t) = |P_\sigma(t) - Q_\theta(t)|$. The maximum distance over a range of values is denoted by $d_\mu[a, b] = \max_{a \le t \le b} d_\mu(t)$. The *Fréchet distance* between two curves is defined as $\delta_{\mathrm{F}}(P, Q) = \inf_\mu d_\mu[0, 1]$. A *Fréchet matching* is a matching μ that realizes the Fréchet distance, that is, $d_\mu[0, 1] = \delta_{\mathrm{F}}(P, Q)$ holds.

Free space diagrams. Alt and Godau [1] describe an algorithm to compute the Fréchet distance based on the decision variant (that is, solving $\delta_{\mathrm{F}}(P, Q) \le \varepsilon$ for some given ε). Their algorithm uses a *free space diagram*, a two-dimensional diagram on the range $[0, m] \times [0, n]$. Every point (x, y) in this diagram is either "free" (white) or not (indicating whether $|P(x) - Q(y)| \le \varepsilon$). The diagram has m columns and n rows; every cell (c, r) ($1 \le c \le m$ and $1 \le r \le n$) corresponds to the edges $p_{c-1}p_c$ and $q_{r-1}q_r$. To compute the Fréchet distance, one finds the smallest ε such that there exists an x- and y-monotone path from point $(0, 0)$ to (m, n) in free space. For this, only certain *critical values* have to be checked. These values correspond to emergence of potentially new allowed paths, so-called *critical events*. The three event types are illustrated in Fig. 3. We scale every row and column in the diagram to correspond to the (relative) length of the actual edge of the curve instead of using unit squares for cells.

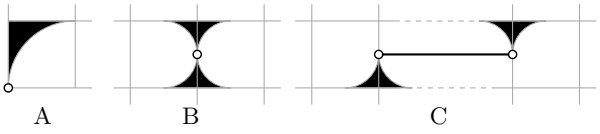


Figure 3: Three event types. (A) Endpoints come within range of each other. (B) Passage opens on cell boundary. (C) Passage opens in row (or column).

3 Locally correct matchings

We introduce *locally correct* matchings, which are a Fréchet matching for any two matched subcurves.

Definition 1 (Local correctness) *Given two polygonal curves P and Q, a matching $\mu = (\sigma, \theta)$ is* locally correct *if for all a, b with $0 \le a \le b \le 1$*

$$d_\mu[a, b] = \delta_{\mathrm{F}}(P_\sigma[a, b], Q_\theta[a, b]).$$

Note that not every Fréchet matching is locally correct. The question arises whether a locally correct matching always exists and if so, how to compute it.

Existence. We prove that there always exists a locally correct matching for any two curves by induction on the number of edges in the curves. First, we present two simple observations for the two base cases.

Observation 1 (P is a point) *For two polygonal curves P and Q with $m = 0$, a locally correct matching is (σ, θ), where $\sigma(t) = 0$ and $\theta(t) = t \cdot n$.*

Observation 2 (Line segments) *For two polygonal curves P and Q with $m = n = 1$, a locally correct matching is (σ, θ), where $\sigma(t) = \theta(t) = t$.*

For induction, we split the two curves based on events (see Fig. 4). Since each split must reduce the problem size, we ignore any events on the left or bottom boundary of cell $(1, 1)$ or on the right or top boundary of cell (m, n). This excludes both events of type A. A free space diagram is *connected* at value ε, if a monotonous path exists from the boundary of cell $(1, 1)$ to the boundary of cell (m, n). A *realizing event* is a critical event at the minimal value ε such that the corresponding free space diagram is connected.

Let $\mathcal{E}$ denote the set of concurrent realizing events for two curves. A *realizing set* E_{r} is a subset of $\mathcal{E}$ such that the free space admits a monotonous path from cell $(1, 1)$ to cell (m, n) without using an event

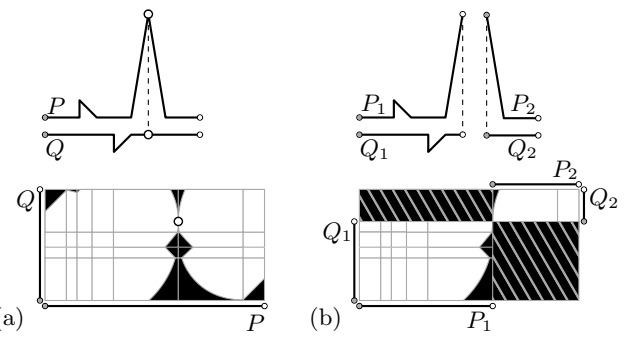


Figure 4: (a) Curves with the free space diagram for $\varepsilon = \delta_{\mathrm{F}}(P, Q)$ and the realizing event. (b) The event splits each curve into two subcurves. The hatched areas indicate parts that disappear after the split.

in $\mathcal{E}\backslash E_r$. Note that a realizing set cannot be empty. When $\mathcal{E}$ contains more than one realizing event, some may be "insignificant": they are never required to actually make a path in the free space diagram. A realizing set is *minimal* if it does not contain a strict subset that is a realizing set. Such a minimal realizing set contains only "significant" events. We omit the (straightforward) proof of the following lemma.

Lemma 2 *For two polygonal curves P and Q with $m > 1$ and $n \geq 1$, there exists a minimal realizing set.*

In the remainder, we use realizing set to indicate a minimal realizing set, unless indicated otherwise. To prove the existence of a locally correct Fréchet matching, we prove the following, stronger lemma. Informally, it states that curves have a locally correct matching that is "closer" (except in cell $(1,1)$ or (m,n)) than the distance of their realizing set. In addition, this matching is linear inside every cell.

Lemma 3 *If the free space diagram of two polygonal curves P and Q is connected at value ε, then there exists a locally correct Fréchet matching $\mu = (\sigma, \theta)$ such that $d_\mu(t) \leq \varepsilon$ for all t with $\sigma(t) \geq 1$ or $\theta(t) \geq 1$, and $\sigma(t) \leq m-1$ or $\theta(t) \leq n-1$. Furthermore, μ is linear in every cell.*

Proof. We prove this by induction on $m+n$. The base cases ($m = 0$, $n = 0$, and $m = n = 1$) follow from Observation 1 and Observation 2.

For the induction, we assume that $m \geq 1$, $n \geq 1$, and $m + n > 2$. By Lemma 2, there exists a realizing set E_r for P and Q, say at value ε_r. The set contains realizing events $e_1, \ldots, e_k$ ($k \geq 1$), numbered in lexicographic order. By definition, $\varepsilon_r \leq \varepsilon$ must hold. Suppose that E_r splits curve P into $P_1, \ldots, P_{k+1}$ and curve Q into $Q_1, \ldots, Q_{k+1}$. By definition of a realizing event, none of the events in E_r occur on the right or top boundary of cell (m,n). Therefore, the following holds for any i ($1 \leq i \leq k+1$): $m_i \leq m$, $n_i \leq n$, and $m_i < m$ or $n_i < n$. Since there is a path in the free space diagram at ε_r through all events in E_r, the induction hypothesis implies that, for any i ($1 \leq i \leq k+1$), there exists a locally correct matching $\mu_i = (\sigma_i, \theta_i)$ for P_i and Q_i such that μ_i is linear in every cell and $d_{\mu_i}(t) \leq \varepsilon_r$ for all t with $\sigma_i(t) \geq 1$ or $\theta_i(t) \geq 1$, and $\sigma_i(t) \leq m_i - 1$ or $\theta_i(t) \leq n_i - 1$. Combining these matchings with the events in E_r yields a matching $\mu = (\sigma, \theta)$ for (P, Q). As we argue below, this matching is locally correct and satisfies the additional properties. The matching of an event corresponds to a single point (type B) or a horizontal or vertical line (type C). By induction, μ_i is linear in every cell. Since all events occur on cell boundaries, the cells of the matchings and events are disjoint. Therefore, the matching μ is also linear inside every cell.

For $i < k+1$, d_{μ_i} is at most ε_r at the point where μ_i enters cell (m_i, n_i) in the free space diagram of P_i and Q_i. We also know that d_{μ_i} equals ε_r at the top right corner of cell (m_i, n_i). Since μ_i is linear inside the cell, $d_{\mu_i}(t) \leq \varepsilon_r$ also holds for t with $\sigma_i(t) > m_i - 1$ and $\theta_i(t) > n_i - 1$. Analogously, for $i > 0$, $d_{\mu_i}(t)$ is at most ε_r for t with $\sigma_i(t) < 1$ and $\theta_i(t) < 1$. Hence, $d_\mu(t) \leq \varepsilon_r \leq \varepsilon$ holds for t with $\sigma(t) \geq 1$ or $\theta(t) \geq 1$, and $\sigma(t) \leq m-1$ or $\theta(t) \leq n-1$.

To show that μ is locally correct, suppose for contradiction that we have values for a, b such that $\delta_F(P_\sigma[a,b], Q_\theta[a,b]) < d_\mu[a,b]$. If a, b are in between two consecutive events, then we know that the submatching corresponds to one of the matchings μ_i. However, we know that these are locally correct and thus $\delta_F(P_\sigma[a,b], Q_\theta[a,b]) = d_\mu[a,b]$ holds.

Hence, assume that there is at least one event of E_r in between a and b. There are two possibilities: either $d_\mu[a,b] = \varepsilon_r$ or $d_\mu[a,b] > \varepsilon_r$. $d_\mu[a,b] < \varepsilon_r$ cannot hold, since $d_\mu[a,b]$ includes a realizing event. First, assume $d_\mu[a,b] = \varepsilon_r$ holds. If $\delta_F(P_\sigma[a,b], Q_\theta[a,b]) < \varepsilon_r$ holds, then there is a matching that does not use the events in between a and b and has a lower maximum. Therefore, the free space connects point $(\sigma(a), \theta(a))$ with point $(\sigma(b), \theta(b))$ at a lower value than ε_r. This implies that all events between a and b can be omitted, contradicting that E_r is a minimal realizing set.

Now, assume $d_\mu[a,b] > \varepsilon_r$. Let t' denote the highest t for which $\sigma(t) \leq 1$ and $\theta(t) \leq 1$ holds, that is, the point at which the matching exits cell $(1,1)$. Similarly, let t'' denote the lowest t for which $\sigma(t) \geq m-1$ and $\theta(t) \geq n-1$ holds. We know that $d_\mu(t) \leq \varepsilon_r$ holds for any $t' \leq t \leq t''$. Hence, $d_\mu(t) > \varepsilon_r$ can only hold for $t < t'$ or $t > t''$. Suppose that $d_\mu(a) > \varepsilon_r$ holds. Thus, we have that $a < t'$ and μ is linear between a and t'. Therefore, $d_\mu(a) > d_\mu(t)$ holds for any t with $a < t < t'$. Analogously, if $d_\mu(b) > \varepsilon_r$ holds, then $d_\mu(b) > d_\mu(t)$ holds for any t with $t'' < t < b$. Hence, we conclude that $d_\mu[a,b] = \max\{d_\mu(a), d_\mu(b)\}$. Since this gives a lower bound on the Fréchet distance, we conclude that the matching μ is locally correct. □

Further restrictions. We considered restricting the matchings to the "shortest" locally correct matching, where "shortest" refers to the length of the path in the free space diagram. However, Fig. 5 shows that such a restriction does not necessarily improve the

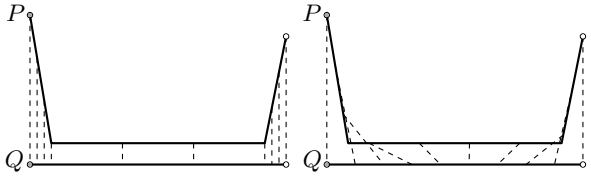


Figure 5: Two locally correct Fréchet matchings for curves P and Q. The right matching is the shortest.

quality of the matching. Another potential constraint we are currently investigating is "local optimality". Intuitively, a matching is locally optimal if no small change decreases the matched distance locally.

4 Algorithm

The existence proof results in a recursive algorithm with execution time $O((m+n)\cdot m\cdot n\cdot \log(m\cdot n))$. Fig. 1 (left), Fig. 2 (left), Fig. 5 (left), and Fig. 6 illustrate matchings computed with our algorithm.

Algorithm 1 `FindLocallyCorrectMatching`(P,Q)

Require: P and Q are curves with m and n edges
Ensure: A locally correct matching for P and Q

1: **if** $m=0$ **or** $n=0$ **then**
2: **return** (σ,θ) where $\sigma(t)=t\cdot m$, $\theta(t)=t\cdot n$
3: **else if** $m=n=1$ **then**
4: **return** (σ,θ) where $\sigma(t)=\theta(t)=t$
5: **else**
6: Find event e_r of a minimal realizing set
7: Split P into P_1 and P_2 according to e_r
8: Split Q into Q_1 and Q_2 according to e_r
9: $\mu_1 \to$ `FindMatching`(P_1,Q_1)
10: $\mu_2 \to$ `FindMatching`(P_2,Q_2)
11: **return** concatenation of μ_1, e_r, and μ_2

Using the notation of Alt and Godau [1], $L^F_{i,j}$ denotes the interval of free space on the left boundary of cell (i,j); $L^R_{i,j}$ denotes the subset of $L^F_{i,j}$ that is reachable from point $(0,0)$ of the free space diagram with a monotonous path in the free space. Analogously, $B^F_{i,j}$ and $B^R_{i,j}$ are defined for the bottom boundary.

With a slight modification to the decision algorithm, we can compute the minimal value of ε such that a path is available from cell $(1,1)$ to cell (m,n). This requires only two changes: $B^R_{1,2}$ should be initialized with $B^F_{1,2}$ and $L^R_{2,1}$ with $L^F_{2,1}$; the answer should be "yes" if and only if $B^R_{m,n}$ or $L^R_{m,n}$ is non-empty.

Realizing set. By computing the Fréchet distance using the modified Alt and Godau algorithm, we obtain an ordered, potentially non-minimal realizing set $\mathcal{E}=\{e_1,\ldots,e_l\}$. Let E_k denote the first k events of $\mathcal{E}$. The algorithm must find an event that is contained in a realizing set. We use a binary search on $\mathcal{E}$ to find the r such that E_r contains a realizing set, but E_{r-1} does not. This implies that event e_r must be contained in a realizing set. We use e_r to split the curves. Note that r is unique due to monotonicity.

For correctness, the order of events in $\mathcal{E}$ must be consistent in different iterations, for example, by using a lexicographic order. Set E_r contains only realizing sets that use e_r. Therefore, E_{r-1} contains a realizing set to connect cell $(1,1)$ to e_r and e_r to cell (m,n). Thus any event found in subsequent iterations is part of E_{r-1} and is part of a realizing set with e_r.

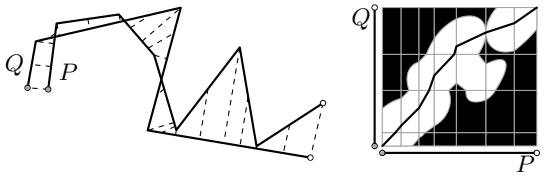


Figure 6: Locally correct matching produced by the algorithm. Free space diagram drawn at $\varepsilon=\delta_\mathrm{F}(P,Q)$.

To determine whether some E_k contains a realizing set, we check whether cells $(1,1)$ and (m,n) are connected without "using" the events of $\mathcal{E}\backslash E_k$. To do this efficiently, we further modify the Alt and Godau algorithm. We require only a method to prevent events in $\mathcal{E}\backslash E_k$ from being used. After $L^R_{i,j}$ is computed, we check whether the event e (if any) that ends at the left boundary of cell (i,j) is part of $\mathcal{E}\backslash E_k$ and necessary to obtain $L^R_{i,j}$. If this is the case, we replace $L^R_{i,j}$ with an empty interval. Event e is necessary if and only if $L^R_{i,j}$ is a singleton. To obtain an algorithm that is numerically more stable, we introduce entry points. The *entry point* of the left boundary of cell (i,j) is the maximal $i'<i$ such that $B^R_{i',j}$ is non-empty. These values are easily computed during the decision algorithm. Assume e starts on the left boundary of cell (i_s,j). Event e is necessary if and only if $i'<i_\mathrm{s}$. Therefore, we use the entry point instead of checking whether $L^R_{i,j}$ is a singleton. This process is analogous for horizontal boundaries of cells.

We silently assumed that each event in $\mathcal{E}$ ends at a different cell boundary. If multiple events end at the same cell boundary, then these events occur in the same row (or column) and it suffices to consider only the event that starts at the rightmost column (or highest row). This justifies the assumption and ensures that $\mathcal{E}$ contains $O(m\cdot n)$ events. Hence, computing e_r (line 6) takes $O(m\cdot n\cdot\log(m\cdot n))$ time.

References

[1] H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *Int. J. of Comp. Geometry and Appl.*, 5(1):75–91, 1995.

[2] K. Buchin, M. Buchin, and J. Gudmundsson. Constrained free space diagrams: a tool for trajectory analysis. *Int. J. of GIS*, 24(7):1101–1125, 2010.

[3] A. Efrat, L. Guibas, S. Har-Peled, J. Mitchell, and T. Murali. New Similarity Measures between Polylines with Applications to Morphing and Polygon Sweeping. *Discrete & Comp. Geometry*, 28(4):535–569, 2002.

[4] A. Maheshwari, J. Sack, K. Shahbaz, and H. Zarrabi-Zadeh. Fréchet distance with speed limits. *Comp. Geometry: Theory and Appl.*, 44(2):110–120, 2011.

[5] C. Wenk, R. Salas, and D. Pfoser. Addressing the need for map-matching speed: Localizing global curve-matching algorithms. In *Proc. 18th Int. Conf. on Sci. and Stat. Database Management*, pages 379–388, 2006.

Selection of Extreme Points and Halving Edges of a Set by its Chirotope

T. Miltzow* A. Pilz†

Abstract

Many properties of finite point sets only depend on the relative position of the points, e.g., on the order type of the set. However, many fundamental algorithms in computational geometry rely on coordinate representations. This includes the straightforward algorithms for finding a halving line for a given planar point set, as well as finding a point on the convex hull, both in linear time. In his monograph *Axioms and Hulls*, Knuth asks whether these problems can be solved in linear time in a more abstract setting, given only the orientation of each point triple, i.e., the set's chirotope, as a source of information. We answer this question in the affirmative for sets realizable in the Euclidean plane. More precisely, we can find a halving line through any given point, as well as the vertices of the convex hull edges that are intersected by the supporting line of any two given points of the set in linear time.

1 Introduction

In computational geometry, many fundamental properties of finite point sets do not depend on the actual coordinates of each point in real space, but rather on the relative position of the points among each other. In their landmark paper, Goodman and Pollack [5] define the order type of a point set. In the plane, two point sets have the same *order type* if there is a bijection π between the sets s.t. for every triple p, q, r of the first set, the corresponding points $\pi(p), \pi(q)$, and $\pi(r)$ have the same orientation (i.e., are both oriented clockwise or counterclockwise).[1] This orientation can be tested by the inequality

$$det \begin{pmatrix} p_x & p_y & 1 \\ q_x & q_y & 1 \\ r_x & r_y & 1 \end{pmatrix} > 0,$$

which indicates whether r is to the left of the directed line through p and q, i.e., whether the triple is oriented counterclockwise. The sign of the determinant therefore gives a predicate $\nabla(p, q, r)$ that is true iff the triple is oriented counterclockwise. Many combinatorial properties of a set of points only depend on its order type, like its convex hull, the set of its crossing-free graphs, etc. Note that we implicitly assume throughout this paper that all sets are in general position, i.e., do not contain collinear triples.

There are properties of point sets that are more "metric", like the set's Delaunay triangulation. It is straightforward to construct two sets of the same order type whose Delaunay triangulations are different. Guibas and Stolfi [6] separate topological from geometric aspects, using a predicate `InCircle`(p, q, r, s) that is true iff the triple (p, q, r) is oriented counterclockwise and the point s lies inside the circle defined by the first three points. Their Delaunay triangulation algorithm depends almost entirely on this predicate, making it a robust approach, that is intended to be easy to implement and to prove.

Motivated by this approach, Knuth [9] develops axiomatic systems following these two tests. He defines five axioms over a ternary predicate P and calls sets of triples obeying them *CC Systems*. These are fulfilled by all point sets in the Euclidean plane, with $P = \nabla$ defined as point triple orientation. For these sets, the CC Systems are equivalent to the point set order types. However, there exist CC Systems that can not be constructed as point sets in $\mathbb{R}^2$. These are called *non-realizable*. CC Systems are equivalent to *abstract order types*. These are used in so-called *abstract order type extension* [1].

The concept of the convex hull of a point set generalizes to all CC Systems. The axiomatic approach can also be extended to cover Delaunay triangulations. In the axiomatic settings, Knuth provides $O(n \log n)$ time algorithms for both problems, where the time bound for the latter holds in the average case. He points out that the algorithm of Guibas and Stolfi uses the coordinate representation to find a line that partitions the point set into two equally sized subsets (cf. [6, pp. 110–111]). Open Problem 1 in [9, pp. 97–98] therefore asks for an algorithm to find such a partition of a CC System in linear time. The problem is straightforward when given an extreme point of the set (i.e., an element of the convex hull boundary). Proving the existence of a linear-time algorithm for finding a single extreme point is also explicitly part of the open problem.

*Institute of Computer Science, Freie Universität Berlin, Germany. miltzow@mi.fu-berlin.de

†Recipient of a DOC-fellowship of the Austrian Academy of Sciences at the Institute for Software Technology, Graz University of Technology, Austria. apilz@ist.tugraz.at. Part of this work was done while A.P. was visiting the Work Group Theoretical Computer Science at the Institute of Computer Science, Freie Universität Berlin, Germany.

[1] It is common to also consider sets to be of the same order type if the orientation of all triples is inverted in the second set.

This is an extended abstract of a presentation given at EuroCG 2012. 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 in a conference with formal proceedings and/or in a journal.

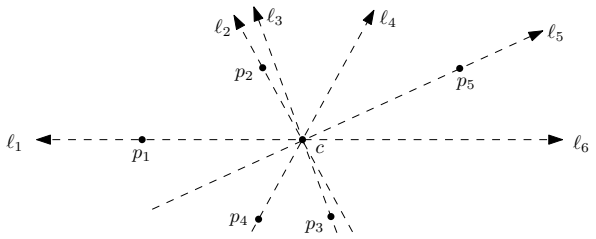


Figure 1: Rotating the line ℓ around c starting at p_1 induces an order on the set.

In this work, we answer both parts of the open problem in the affirmative. We first give a simple algorithm that, given a point c of a set S of size n, finds a halving edge through c; more specifically, a second point $c' \in S$ s.t. not more than $\lceil \frac{n-2}{2} \rceil$ points are on each side of the supporting line of c and c' in $O(n)$ time. We then describe an algorithm that, given two points p and q, returns the edge of the convex hull that is crossed by the ray from p through q. We show that the algorithm runs in $O(n)$ time for realizable sets. Our algorithms are correct for general CC Systems. In this extended abstract, we prove the time bound only for realizable sets. The linear time bound for all abstract order types will be considered in [2].

Our main motivation is to show that the asymptotic running time of these problems does not depend on the representation by coordinates. While an arbitrary halving edge can easily be found by picking a point with median, say, x-coordinate, the problem is more sophisticated when the halving line should pass through a predefined point. E.g., the linear time Ham-Sandwich-Cut algorithm of Lo, Matoušek and Steiger [10] can be adapted to find a halving line through a point. The straightforward way of finding an extreme point of a set given by coordinates is selecting the one with, say, lowest x-coordinate. Finding a convex hull edge that is traversed by a given line in linear time is a subroutine of the Ultimate Convex Hull Algorithm [8]. There, the median of the slopes of an arbitrary matching of the points is used for the search and prune approach. Recent algorithms that operate only on triple orientations involve computing the convex hull of disks [7].

2 Computing a halving edge in linear time

In this section, we describe an algorithm to compute a halving edge through a given point c of a finite point set S of size n in the plane. The idea is to split the point set evenly by two double wedges centered at c. We identify the double wedge that must contain a halving edge, and continue with the points therein.

We start by picking a point $p_1 \in S \setminus \{c\}$ arbitrarily. Imagine a directed line ℓ rotated by 180° clockwise around c, starting at p_1. Let $p_2, \ldots, p_{n-1}$ be the remaining points in the order ℓ passes through them. We denote this order by $\prec$. This order is actually the one described in [9, p. 16]. By $\ell_1, \ldots, \ell_n$ we denote the "snapshots" of ℓ at the points $p_1, \ldots, p_{n-1}, p_1$ respectively. Thus, ℓ_1 and ℓ_n have opposite directions. See Figure 1 for an illustration.

Since for each pair of $S \setminus \{c\}$, we can compute the relative order in which ℓ passes through them in constant time, the median $m = p_{\lfloor \frac{n}{2} \rfloor}$ with respect to $\prec$ can be computed in $O(n)$ time [3]. (Note that the number of points to the right of ℓ may *not* change monotonically while rotating.) The supporting lines of cp_1 and cm define two double wedges, containing $S_1 = \{p \in S : p \prec m\}$ and $S_2 = \{p \in S : m \prec p\}$, respectively, each having not more than $\lceil \frac{n-2}{2} \rceil$ points (see Figure 2). We use the following standard argument. If ℓ_1 is not a halving line, we have w.l.o.g. more than $\lceil \frac{n-2}{2} \rceil$ points to the right of ℓ_1 and thus less than $\lfloor \frac{n-2}{2} \rfloor$ points to the right of ℓ_n. Due to our general position assumption, only one point changes sides at the same time. Thus, ℓ must be a halving line at some point during the rotation. If less than $\frac{n-2}{2}$ points are right of $\ell_{\lfloor \frac{n}{2} \rfloor}$, then one of the points in S_1 forms a halving edge with c (recall that there are w.l.o.g. more points right of ℓ_1). Otherwise one of the points in S_2 does.

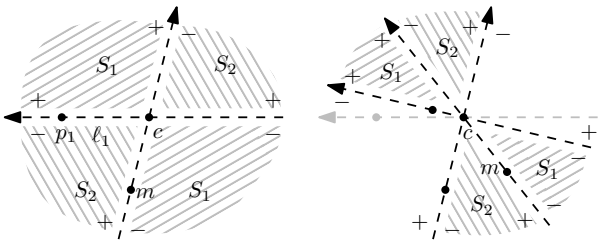


Figure 2: The sign + marks the half plane which contains more than $\frac{n-2}{2}$ points. S_1 and S_2 are indicated by the gray tiling patterns. The figure shows the first (left) and a later iteration (right).

This way we can decide which part contains a halving edge, exclude the other subset and iterate. Observe that all the points we exclude belong to two (open) wedges starting at c, one entirely to the left and one entirely to the right of the non-removed lines of $\ell_2, \cdots, \ell_{n-1}$. The closure of each wedge contains a ray of ℓ_1. We remember how many points each of these wedges contained. Now if we want to compute the number of points to the right of some line ℓ_i in subsequent iterations, then we only need to consider the remaining points and add the number of points belonging to the excluded wedge to the right of ℓ_i. The algorithm has linear running time, as we exclude approximately half of the points in each linear-time iteration.

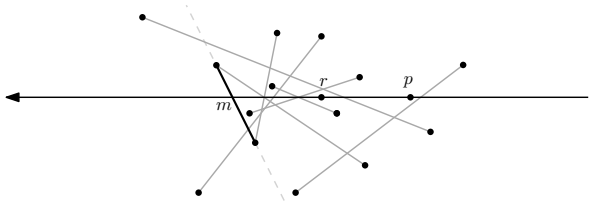


Figure 3: An arbitrary matching of edges that partition S, and m, the result of BasicMin.

Theorem 1 *Given a point set $S \subset \mathbb{R}^2$ of size n in general position and a point $c \in S$, one can find a halving edge of S incident to c in linear time.*

The algorithm can directly be extended to higher dimensions. Order types in $\mathbb{R}^d$ are defined analogously as in the plane, see [5]. The orientation of a $(d+1)$-simplex can be computed by the corresponding determinant. Let $(p_1, \ldots, p_{d+1})$ be an ordered tuple of points in $\mathbb{R}^d$. For the oriented hyperplane H through the points $(p_1, \ldots, p_d)$, $\nabla(p_1, \ldots, p_{d+1})$ indicates on which side of H the point p_{d+1} lies.

Corollary 2 *Given a point set $S \subset \mathbb{R}^d$ in general position, d constant, and distinct points $c_1, \ldots, c_{d-1} \in S$, one can find a halving hyperplane through $c_1, \ldots, c_{d-1}$ in linear time.*

3 Computing a Convex Hull Edge in Linear Time

As a first step we describe an algorithm called BasicMin, which plays a crucial role as a subroutine. Suppose we are given two points $p, r \in S$. We assume that pr is a halving edge of S and that $n = |S|$ is even. W.l.o.g. let r be the coordinate origin and let p be on the positive part of the x-axis. Let M be an arbitrary perfect matching between the points above and below the x-axis, i.e., for any edge $s = ab \in M$ we have $\nabla(p, r, a)$ and $\neg\nabla(p, r, b)$. See Figure 3 for an illustration.

Let $\ltimes$ be the binary operator that accepts two edges $s, s' \in M$ as input and returns the edge on the convex hull boundary of $s \cup s'$ that crosses the x-axis at the smallest x-coordinate, i.e., the pair of endpoints whose upward-directed supporting line has all other points of s and s' to the right. The relevant property of the operator is that the crossing of $\ltimes(s, s')$ with the x-axis is not right of the crossings of s and s' with the x-axis.

BasicMin takes a point set S and two points p and r as input, partitions S arbitrarily into the matching $M = \{s_1, \ldots, s_{\left(\frac{n-2}{2}\right)}\}$, and computes a special edge $m = m_{\left(\frac{n-2}{2}\right)}$ iteratively via

$$\begin{aligned} m_1 &= s_1 \\ m_{(i+1)} &= \ltimes(m_i, s_{(i+1)}) \end{aligned}$$

Note that m does not need to be on the convex hull of the whole set. Also, m may depend on the (undefined) order in which the elements of M are processed by BasicMin. However, it has the following useful property.

Lemma 3 *Let ℓ be a line directed upwards, crossing the x-axis left of the crossing of m. Then at least $\frac{n-2}{2}$ points of S are to the right of ℓ.*

Proof. Note that the crossing of ℓ with the x-axis is further to the left than any other such crossing of M. Assume, for the purpose of contradiction, that there is an edge in M for which both points lie to the left of ℓ. Then also the crossing with the x-axis is to the left of the crossing of m, a contradiction. □

Note that, even though the argumentation of Lemma 3 involves relative positions of crossings, the constant-time operation $\ltimes$ only needs to use ∇. Now we are ready to give the main algorithm.

Theorem 4 *Given two points p, q of a point set $S \subset \mathbb{R}^2$ of size n in general position, one can find the edge e of the set's convex hull that passes through the ray pq in $O(n)$ time.*

Proof. W.l.o.g. let pq be horizontal with q left of p. Note first that the case where q is a vertex of the convex hull can be identified in linear time. We therefore concentrate on the setting where one endpoint of e is below pq and the other one is above. Let U be the set strictly above pq, whereas L is the set strictly below pq. W.l.o.g. let $|U| \geq \frac{n-2}{2}$.

Consider the endpoint $u_1 \in U$ of e. If we remove u_1, the ray pq crosses the convex hull at a new edge e' with an endpoint $u_2 \in U$. Note that the other endpoint of e' might now be q. In any case, iteratively removing points from U of the intersected convex hull edge induces an order on U (see Figure 4). Note that this corresponds to the order in which the points of U are traversed by a tangent t of $CH(L \cup \{p, q\})$ that is rotated clockwise around that hull, starting at e and ending at pq. The main observation is that, given a point u_i and the tangent t passing through u_i, we can discard all points of U to the right of u_i with respect to the point l where t touches $CH(L \cup \{p, q\})$, since none of these points can be u_1. The support l of t can be found in linear time, since the radial order of $L \cup \{p, q\}$ around any u_i is linear.

Note that these observations already imply the following randomized approach. Select any element u_i of U at random. The other support l of the tangent (recall that this might as well be q) can be found in linear time. We discard the points of U right of lu_i and iterate. However, consecutive "bad" choices of u_i result in overall quadratic worst-case behavior. We

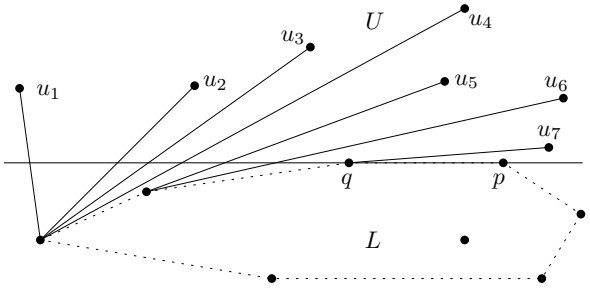


Figure 4: The order on U defined by removing vertices of the intersected convex hull edge.

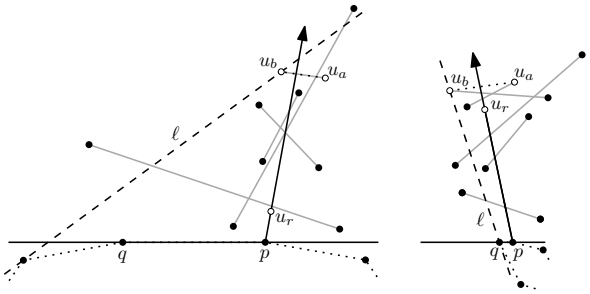


Figure 5: The edge $u_a u_b$ allows to prune half of the upper points.

therefore have to make a "good" choice of u_i in order to discard a linear number of points per iteration.

The points of U are ordered linearly around p. Let u_r be the median of this order, which we select in linear time. Let M be an arbitrary perfect matching between the points of U to the left and to the right of pu_r (maybe omitting one point). Now we apply BasicMin on M resulting in an edge $m = u_a u_b$. By construction, all edges of M as well as $u_a u_b$ cross the ray pu_r. Now, find the upward-directed tangents t_a and t_b of $CH(L \cup \{p, q\})$ through u_a and u_b, respectively. Let $\ell = t_a$ if u_b is to the right of t_a, otherwise let $\ell = t_b$. Note that p is right of ℓ since q is included in the set that we place the tangent on. There are two cases to consider. If ℓ crosses the ray pu_r, the crossing of $u_a u_b$ is on the ray between p and its crossing with ℓ. Due to Lemma 3, at least half of the points of M are right of ℓ. Otherwise, if ℓ does not cross the ray pu_r, then all points of U to the right of the ray are also right of ℓ. In both cases, we can discard at least half of the points, which is at least a quarter of the overall set S. We can therefore in linear time reduce this problem to constant size such that it then can be solved by a brute-force approach. □

Note that the parts of the so-called Ultimate Convex Hull Algorithm by Kirkpatrick and Seidel [8] that depend on coordinates are essentially the one that find the convex hull edge on the ray that separates a subproblem into two parts. Also, Chan's output sensitive algorithm [4] can be implemented in our setting using Theorem 4. Both allow to improve the time bound given in [9] for realizable point sets regarding output-sensitivity to $O(n \log h)$ for h extreme points.

4 Conclusion

We showed two algorithms that only use the information whether a point triple is oriented clockwise or counterclockwise. Both a halving edge at a given point and a convex hull edge crossing a specified ray can be found in linear time, even without being given the coordinate representation. Our proofs used geometric arguments. While it is rather straightforward to generalize the proof of Theorem 1 to abstract order types, generalization is more sophisticated for the second result, especially concerning Lemma 3. There, our proof relies on the relative position of crossings among supporting lines of the set. The general setting will be addressed in [2].

Acknowledgements

We thank Oswin Aichholzer, Thomas Hackl, and Günter Rote, as well as an anonymous referee, for helpful suggestions on improving the presentation of the results.

References

[1] O. Aichholzer and H. Krasser. Abstract order type extension and new results on the rectilinear crossing number. *Comput. Geom.*, 36(1):2–15, 2007.

[2] O. Aichholzer, T. Miltzow, and A. Pilz. Extreme point search in abstract order types. In preparation.

[3] M. Blum, R. W. Floyd, V. Pratt, R. L. Rivest, and R. E. Tarjan. Time bounds for selection. *J. Comput. System Sci.*, 7(4):448–461, 1973.

[4] T. M. Chan. Optimal output-sensitive convex hull algorithms in two and three dimensions. *Discrete Comput. Geom.*, 16(4):361–368, 1996.

[5] J. E. Goodman and R. Pollack. Multidimensional sorting. *SIAM J. Comput.*, 12(3):484–507, 1983.

[6] L. J. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and computation of Voronoi diagrams. *ACM Trans. Graph.*, 4(2):74–123, 1985.

[7] L. Habert and M. Pocchiola. Computing the convex hull of discs using only their chirotope. In *Proc. 20th European Workshop Comput. Geom.*, pages 111–114, Seville, Spain, March 2004.

[8] D. G. Kirkpatrick and R. Seidel. The ultimate planar convex hull algorithm? *SIAM J. Comput.*, 15(1):287–299, 1986.

[9] D. E. Knuth. *Axioms and Hulls*, volume 606 of *Lecture Notes Comput. Sci.* Springer, 1992.

[10] C.-Y. Lo, J. Matoušek, and W. Steiger. Algorithms for ham-sandwich cuts. *Discrete Comput. Geom.*, 11:433–452, 1994.

Coloring and Guarding Arrangements*

Prosenjit Bose† Jean Cardinal‡ Sébastien Collette‡ § Ferran Hurtado¶ Matias Korman‡
Stefan Langerman‡ ‖ Perouz Taslakian‡

Abstract

Given an arrangement of lines in the plane, what is the minimum number c of colors required to color the lines so that no cell of the arrangement is monochromatic? In this paper we give bounds on the number c, as well as some of its variations. We cast these problems as characterizing the chromatic and the independence numbers of a new family of geometric hypergraphs.

1 Introduction

While dual transformations may allow converting a combinatorial geometry problem about a *configuration of points* into a problem about an *arrangement of lines*, or reversely, the truth is that most mathematical questions appear to be much cleaner and natural in only one of the settings. In many cases, the dual version is considered solely when, besides making sense, it is additionally useful. Both kinds of geometric objects have inspired many problems and attracted much attention. Concerning arrangements of lines, possibly the most prevalent problems consist of studying the number of cells of each size, say triangles, that appear in every arrangement, but many other issues have been considered (see [3,6,7]). There are also problems that combine both kinds of objects, like counting incidences between points and lines, or studying the arrangements of lines spanned by point sets, which includes the celebrated *Sylvester-Gallai problem* on ordinary lines [2].

Many other natural questions can be asked when considering arrangements of colored lines. For example, is it true that every bicolored arrangement of lines has a monochromatic cell? We prove in this paper that the generic answer is no, but that it is yes when the colors are slightly unbalanced. This leads immediately to another simple question: How many colors are always sufficient, and occasionally necessary, to color any set of n lines in such a way that the induced arrangement contains no monochromatic cell? This last question brings manifestly the flavor of *Art Gallery Problems*. While coloring and guarding arrangements of lines may appear at first glance as unrelated problems, there is a clean unifying framework provided by considering appropriate geometric hypergraphs. For example, minimally coloring an arrangement while avoiding monochromatic cells can be reformulated as follows: Let $\mathcal{H}_{line-cell}$ be the geometric hypergraph where vertices are lines and edges represent cells of the arrangement; what is its chromatic number? (Here a proper coloring is one where no hyperedge is monochromatic.)

In this work we consider several questions as the ones described above, which arise as fundamental in terms of coloring and guarding arrangements of lines, and translate consistently into problems on geometric hypergraphs, like maximum independent set, minimum vertex cover, or some coloring parameter.

The terminology for hypergraphs on arrangements is introduced in Section 2, where we also provide a table summarizing our results. Coloring problems are then discussed in Section 3 and guarding problems in Section 4. Due to lack of space, proofs of the results in this paper have been omitted. Details will be given in an upcoming extended version.

2 Definitions and Summary of Results

Let $\mathcal{A}$ be an arrangement of a set of lines L in $\mathbb{R}^2$. This arrangement decomposes the plane into different cells, where a *cell* is a maximal connected component of $\mathbb{R}^2 \setminus L$.

We define $\mathcal{H}_{line-cell} = (L, C)$ as the geometric hypergraph corresponding to the arrangement, where C is the set containing all cells defined by L. Similarly, $\mathcal{H}_{vertex-cell} = (V, C)$ is the hypergraph defined by the vertices of the arrangements and its cells, where $V = \binom{L}{2}$ is the set of intersection of lines in $\mathcal{A}$. Finally, $\mathcal{H}_{cell-zone} = (C, Z)$ is the hypergraph defined by the cells of the arrangement and its zones. The zone of a line ℓ in $\mathcal{A}$ is the set of cells bounded by ℓ. The set Z is defined as the set of subsets of C induced

*Research supported by the the ESF EUROCORES programme EuroGIGA, CRP ComPoSe: Fonds National de la Recherche Scientique (F.R.S.-FNRS) - EUROGIGA NR 13604, for Belgium, and MICINN Project EUI-EURC- 2011-4306, for Spain.

†Carleton University, Ottawa, Canada. jit@scs.carleton.ca

‡Université Libre de Bruxelles (ULB), Brussels, Belgium. {jcardin, secollet, mkormanc,slanger, perouz.taslakian}@ulb.ac.be

§Chargé de Recherches du F.R.S.-FNRS.

¶Universitat Politècnica de Catalunya (UPC), Barcelona, Spain. ferran.hurtado@upc.edu. Partially supported by projects MTM2009-07242 and Gen. Cat. DGR 2009SGR1040.

‖Maître de Recherches du F.R.S.-FNRS.

by the zones of $\mathcal{A}$. Note that this hypergraph is the dual hypergraph of $\mathcal{H}_{line-cell}$.

An *independent set* of a hypergraph $\mathcal{H} = (V, E)$ is a set $S \subseteq V$ such that $\forall e \in E : e \not\subseteq S$. This definition is the natural extension from the graph variant, and requires that no hyperedge is completely contained in S. Analogously, a *vertex cover* of $\mathcal{H}$ is a set $S \subseteq V$ such that $\forall e \in E : e \cap S \neq \emptyset$. The *chromatic number* $\chi(\mathcal{H})$ of $\mathcal{H}$ is the minimum number of colors that can be assigned to the vertices $v \in V$ so that $\forall e \in E : \exists v_1, v_2 \in e : col(v_1) \neq col(v_2)$; that is, no hyperedge is monochromatic.

In the forthcoming sections we give upper and lower bounds on the worst-case values for these quantities on the three hypergraphs defined from a line arrangement. Our results are summarized in Table 1. Note that the maximum independent set and minimum vertex cover are complementary problems. As a result, any lower bound on one gives an upper bound on the other and vice versa. This property, along with the facts that $|L| = n$, $|V| = \binom{n}{2}$, and $|C| = \frac{n(n+1)}{2} + 1$, are used to complement many entries of the table.

The definitions of an independent set and a proper coloring of the $\mathcal{H}_{line-cell}$ hypergraph of an arrangement are illustrated in Figures 1(a) and 1(b), respectively. Similarly, the definition of a vertex cover of the $\mathcal{H}_{vertex-cell}$ and $\mathcal{H}_{cell-zone}$ hypergraphs are illustrated in Figures 1(c), and 1(d), respectively.

3 Coloring Lines, and Related Results

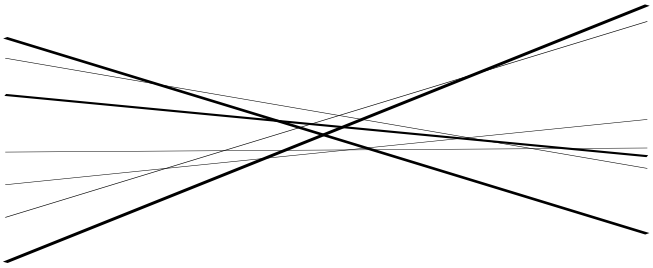
We first consider the chromatic number of the line-cell hypergraph of an arrangement, that is, the number of colors required for coloring the lines so that no cell has a monochromatic boundary. At the end of the section we include some similar results.

3.1 Two-colorability

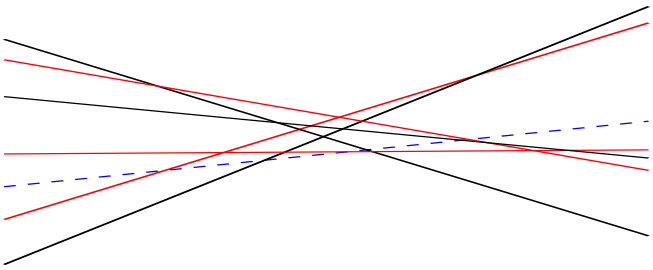
We say that a set of lines L is *k-colorable* if we can color L with k-colors such that no cell is monochromatic (in other words, the corresponding $\mathcal{H}_{line-cell}$ hyper graph has chromatic number k). Any coloring $c : L \to \{0, \ldots, k\}$ that satisfies such a property is said to be *proper*. We first tackle the (simple) question of whether the two-colorable $\mathcal{H}_{line-cell}$ hypergraphs have bounded size:

Theorem 1 *There are arbitrarily large two-colorable sets of lines.*

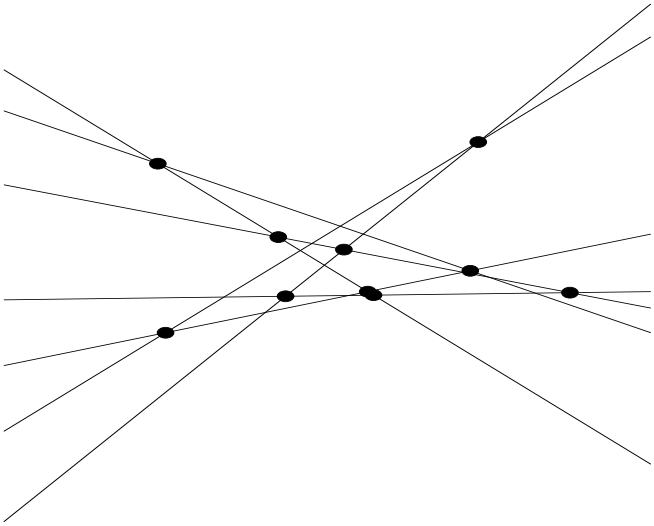
An infinite family of such examples are provided by a set of $2q + 1$ lines in convex position (for any $q \in \mathbb{N}$). It is easy to check that, if we color the lines alternatively red and blue by order of slope, no cell will be monochromatic.



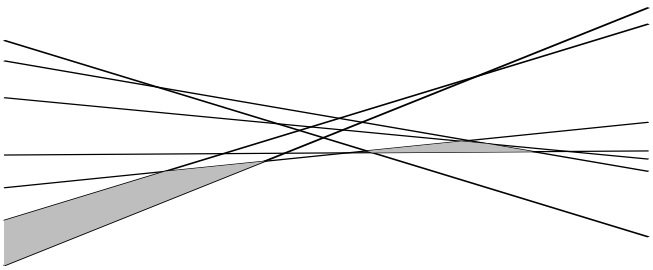
(a) The thick lines form an independent set in the $\mathcal{H}_{line-cell}$ hypergraph: no cell is bounded by those lines only.



(b) A proper 3-coloring of the $\mathcal{H}_{line-cell}$ hypergraph: no cell is monochromatic.



(c) The marked intersections form a vertex cover of the $\mathcal{H}_{vertex-cell}$ hypergraph: every cell has at least one such intersection on its boundary.



(d) The two marked cells form a vertex cover of the $\mathcal{H}_{cell-zone}$ hypergraph: every line has a segment that lies on the boundary of one of those cells.

Figure 1: Illustrations of the definitions.

Hypergraph	Max. Ind. Set	Vertex Cover	Chromatic number
$\mathcal{H}_{line-cell}$	$\geq \frac{\sqrt{n}}{2}$ (Th. 3) $\leq \frac{2n}{3}$ (Th. 4)	$\geq \frac{n}{3}$ (Cor. 14) $\leq n - \frac{\sqrt{n}}{2}$ (Cor. 14)	$\Omega(\log n / \log\log n)$ (Th. 6) $\leq 2\sqrt{n} + O(1)$ (Th. 5)
$\mathcal{H}_{vertex-cell}$	$\geq \frac{n^2}{3} - \frac{5n}{2}$ (Cor. 10) $\leq \frac{n^2}{3} - \frac{n}{2}$ (Cor. 10)	$\geq \frac{n^2}{6}$ (Th. 9) $\leq \frac{n^2}{6} + 2n$ (Th. 9)	2 (Th. 7)
$\mathcal{H}_{cell-zone}$	$\geq \frac{n^2}{2} + \frac{5n}{48} - o(1)$ (Cor. 13) $\leq \frac{n^2}{2} + \frac{5n}{4} + 1$ (Cor. 13)	$\geq \frac{n}{4}$ (Th. 12) $\leq \frac{19n}{48} + o(n)$ (Th. 12)	2 (Th. 8)

Table 1: Worst-case bounds for the different problems studied in this paper.

The coloring used in Theorem 1 uses essentially the same number of lines of each color. This actually holds in general, up to a lower order term.

Theorem 2 *Each color class of a proper 2-coloring $c : L \to \{0, 1\}$ of a set L of n lines has size at most $\frac{n}{2} + \frac{\sqrt{n-1}-1}{2}$.*

3.2 Independent lines in $\mathcal{H}_{line-cell}$

Recall that an independent set of lines in an arrangement is defined as a subset of lines S so that no cell of the arrangement is only adjacent to lines in S.

Theorem 3 *For any set L of n lines, the corresponding $\mathcal{H}_{line-cell}$ hypergraph has an independent set of size $\sqrt{n}/2$.*

Theorem 4 *Given a set L of n lines, an independent set of the corresponding $\mathcal{H}_{line-cell}$ hypergraph has size at most $2n/3$.*

3.3 Chromatic number of $\mathcal{H}_{line-cell}$

In this section, we study the problem of coloring the $\mathcal{H}_{line-cell}$ hypergraph. That is, we want to color the set L so that no cell is monochromatic. We start by giving an upper bound on the required number of colors. This result is proved by iteratively picking a maximal independent set of size $\sqrt{n}/2$.

Theorem 5 *Any arrangement of n lines can be colored with at most $2\sqrt{n} + O(1)$ colors so that no edge of the associated $\mathcal{H}_{line-cell}$ hypergraph is monochromatic.*

We were also able to prove a slightly sublogarithmic lower bound for the chromatic number of $\mathcal{H}_{line-cell}$:

Theorem 6 *There exists an arrangement of n lines whose corresponding hypergraph $\mathcal{H}_{line-cell}$ has chromatic number $\Omega(\log n / \log\log n)$.*

Proof. (Sketch) In order to show the claim, we construct a set of (roughly) k^k lines in which any k-coloring will contain a monochromatic cell (for any $k > 0$). The basic idea behind our construction is the following: consider any coloring with k colors of a set of $k + 1$ lines. By the pigeonhole principle there will be two lines with the same assigned color. Moreover, since the two lines must cross, these two lines must be consecutive in the vertical ordering of the lines at some given x coordinate. Our approach is to cross these two lines with a second pair of lines that have the same color assigned, hence obtaining a monochromatic quadrilateral. The main difficulty of the proof is that the line set must satisfy this property for any k-coloring of L. In particular, we do not know neither which color will be repeated, nor at which x-coordinate. □

3.4 Other coloring results

For the sake of completeness, we end this section by stating two easy results on coloring vertices or cells instead of lines.

Theorem 7 *The chromatic number of $\mathcal{H}_{vertex-cell}$ is 2.*

(Remark that cells of size two only have one vertex, hence cannot be polychromatic. Therefore, we only consider cells of size at least 3.)

The following claim is equivalent to the fact that the dual graph of the arrangement is bipartite

Theorem 8 (Folk.) *The chromatic number of $\mathcal{H}_{cell-zone}$ is 2.*

4 Guarding Arrangements

We now consider the vertex cover problem of the above hypergraphs. That is, we would like to select the minimum number of vertices so that any hyperedge is adjacent to the selected subset. Geometrically speaking, we would like to select the minimum number of vertices (or cells or lines), so that each cell (or line or cell, respectively) contains at least one of the selected items.

4.1 Guarding cells with vertices

We first consider the following problem: given an arrangement of lines $\mathcal{A}$, how many vertices do we need

to pick in order to guard the whole arrangement when lines act as obstacles blocking visibility? This can be rephrased as finding the smallest subset of vertices V so that each cell contains a vertex in V, and thus we are looking for bounds on the size of a vertex cover for $\mathcal{H}_{vertex-cell}$.

Theorem 9 *For any set L of n lines, a vertex cover of the corresponding $\mathcal{H}_{vertex-cell}$ hypergraph has size at most $n^2/6+2n$. Furthermore, $n^2/6$ vertices might be necessary.*

Recall that the hypergraph $\mathcal{H}_{vertex-cell}$ has $\binom{n}{2} = \frac{n^2}{2} - \frac{n}{2}$ vertices. Combining this fact with the previous bounds on the size of a vertex cover allow us to get similar bounds for the independent set problem.

Corollary 10 *For any set L of n lines, a maximum independent set of the corresponding $\mathcal{H}_{vertex-cell}$ hypergraph has size at least $n^2/3 - O(n)$. Furthermore, there exists sets of lines whose largest independent set has size at most $\frac{n^2}{3} - \frac{n}{2}$.*

4.2 Guarding lines with cells

We now consider the problem of touching all lines of L with a smallest subset of cells, i.e., we look for bounds on the size of a vertex cover for $\mathcal{H}_{cell-zone}$.

Theorem 11 *Given a set L of n lines, a minimal vertex cover of the corresponding $\mathcal{H}_{cell-zone}$ hypergraph has size at most $\lceil \frac{n}{2} \rceil$.*

We next provide a lower bound, and improve as well on the upper bound, for large values of n.

Theorem 12 *Given any set L of n lines, a minimal vertex cover of the corresponding $\mathcal{H}_{cell-zone}$ hypergraph has size at most $\frac{19n}{48} + o(n)$. Moreover, there exists a set L of n lines, such that every vertex cover of the corresponding $\mathcal{H}_{cell-zone}$ hypergraph has size at least $\frac{n}{4}$.*

Corollary 13 *For any set L of n lines, a maximum independent set of the corresponding $\mathcal{H}_{cell-zone}$ hypergraph has size at least $\frac{n^2}{2} + \frac{5n}{48} - o(1)$ and at most $\frac{n^2}{2} + \frac{n}{4} + 1$.*

4.3 Guarding cells with lines

For the sake of completeness, we also give bounds on the number of lines needed to guard (touch) all cells.

Corollary 14 *For any set L of n lines, its minimal vertex cover of the corresponding $\mathcal{H}_{line-cell}$ hypergraph has size at least $n/3$ and at most $n - \frac{\sqrt{n}}{2}$.*

5 Concluding Remarks

Clearly, the main open problems arising from our work consist of closing gaps (when they exist) between lower and upper bounds; this is especially interesting in our opinion for the problem of coloring lines without producing any monochromatic cell.

However, it is worth noticing that there are several computational issues that are interesting as well. For example, it is unclear to us which is the complexity of deciding whether a given arrangement of lines admits a two-coloring in which no cell is monochromatic.

References

[1] P. Bose, H. Everett, and S. Wismath. Properties of Arrangements. *International Journal of Computational Geometry*, 13(6), pp. 447-462, 2003.

[2] P. Brass, W. Moser and J. Pach. *Research Problems in Discrete Geometry*. Vieweg Verlag, 2004.

[3] S. Felsner. *Geometric Graphs and Arrangements*. Springer, Berlin, 2005.

[4] S. Felsner, F. Hurtado, M. Noy and I. Streinu. Hamiltonicity and colorings of arrangement graphs. *Discrete Applied Mathematics*, 154(17):2470-2483, 2006.

[5] Z. Füredi and I. Palásti. Arrangements of lines with a large number of triangles. *American Mathematical Society*, 92(4), 1984.

[6] B. Grünbaum. *Arrangements and Spreads*. Regional Conf. Ser. Math., *American Mathematical Society*, 1972 (reprinted 1980).

[7] B. Grünbaum. How many triangles? *Geombinatorics*, 8:154–159, 1998.

[8] B. Grünbaum. Arrangements of colored lines. Abstract 720-50-5, *Notices Amer. Math. Soc.*, 22(1975), A-200.

[9] B. Grünbaum. Monochromatic intersection points in families of colored lines. *Geombinatorics*, 9:3–9, 1999.

[10] B. Grünbaum. Two-coloring the faces of arrangements. *Periodica Mathematica Hungarica*, 11(3):181–185, 1980.

[11] A. Kaneko and M. Kano. Discrete geometry on red and blue points in the plane - a survey -. In *Discrete and Computational Geometry, The Goodman-Pollack Festschrift*, Springer, Berlin, B. Aronov, S. Basu, J. Pach, M. Sharir, eds., pp. 551-570, 2003.

[12] J. Leaños, M. Lomelí, C. Merino, G. Salazar, and J. Urrutia. Simple euclidean arrangements with no (≥ 5)-gons. *Discrete and Computational Geometry*, 38(3):595–603, 2007.

[13] E. Lucas. *Récréations mathématiques IV*. Paris, 1894.

[14] T. S. Motzkin. Nonmixed connecting lines. Abstract 67T 605, *Notices Amer. Math. Soc.*, 14(1967), p. 837.

[15] J. O'Rourke. *Art Gallery, Theorems and Algorithms*. Oxford University Press, 1987.

[16] J. O'Rourke. Visibility. Chapter in *Handbook of Discrete and Computational Geometry*, CRC Press LLC, Boca Raton, FL, 2nd edition, J. E. Goodman and J. O'Rourke eds., pp. 643-665, 2004 2nd ed.ition).

[17] J. Urrutia. Art Gallery and Illumination Problems. Chapter in *Handbook on Computational Geometry*, Elsevier Science Publishers, J.R. Sack and J. Urrutia, eds., pp. 973-1026, 2000.

Energy-Aware Art Gallery Illumination

Alexander Kröller* Christiane Schmidt*

Abstract

We consider a variant of the Art Gallery Problem, where a polygonal region is to be covered with light sources that emit light fading with distance. We describe a practical approximation algorithm based on the simplex method and primal-dual LP separation. For the case where the light positions are given, we describe a fully polynomial-time approximation scheme.

1 Introduction

The classical Art Gallery Problem asks for the minimum number of guards placed in a polygon that allow for complete coverage of the polygon. This is one of the best-known problems in computational geometry, see the excellent book by O'Rourke [5] for an introduction into the subject.

In this paper, we consider a variant of the problem, motivated from discussions with practitioners in 3D laser scanning. When planning where to place scanners, they solve an Art Gallery type problem. As scanning quality diminishes over distance, they face additional constraints. In our problem variant, we model this as fading light: The guards are seen as light sources that can be operated at different brightness levels. The emitted light fades over distance.

A restricted case of this problem has been addressed by Eisenbrand *et al.* [2]; they assume fixed positions for the light sources and only consider illuminating a 1D line. In this model, they describe exact and approximative solution based on mathematical programming.

2 Problem Definition and Notations

We consider a given polygon P, possibly with holes. We denote by $D := \mathrm{diam}P$ the diameter of P. For a point $p \in P$ the *visibility polygon* $\mathcal{V}(p)$ is the (star-shaped) set of all points of P visible from p.

The original Art Gallery Problem (AGP) is defined as follows: We say a *guard set* $G \subseteq P$ *covers* P iff $\cup_{g\in G}\mathcal{V}(g) = P$. We ask for a covering G of minimum cardinality.

In this paper, we consider a variant of this problem: We assume the guards $g \in G$ to be, say, light sources whose intensity, and therefore energy consumption, can be controlled. We denote the intensity of $g \in G$ by $x_g \geqslant 0$. Furthermore, light suffers from fading over distance. A point $w \in \mathcal{V}(g)$ receives only $\varrho(g, w)x_g$ from g, where

$$\varrho(g,w) := \begin{cases} \|g-w\|^{-\alpha} & : \quad \|g-w\| \geqslant 1 \\ 1 & : \quad \text{otherwise} \end{cases} . \qquad (1)$$

Here $\|\cdot\|$ denotes the Euclidean norm in $\mathbb{R}^2$. The parameter $\alpha \geqslant 0$ defines the fading rate of the emitted light. We limit ϱ to be at most 1 to avoid strange effects close to g—otherwise the light intensity would be unbounded, no matter how intense the light itself is.

We say a guard set $G \subseteq P$ with intensities $(x_g)_{g\in G}$ *illuminates* P iff every point in P is sufficiently lit, which we normalize to mean receiving a total light intensity of at least one, i.e.,

$$\sum_{g\in P:\ w\in\mathcal{V}(g)} \varrho(g,w)x_g \geqslant 1 \quad \forall w \in P \qquad (2)$$

Our objective is to find an illuminating guard set G of minimum total energy $\sum_{g\in G} x_g$.

It should be noted that the case $\alpha = 0$ corresponds to the original Art Gallery problem with fractional guards. A falloff of $\alpha = 2$ models the physical behavior of light, which has quadratic fading.

Our actual main interest is the case $\alpha = 1$, which models the falloff of a laser scanner rotating around an axis at constant angular speed. The intensity x_g then describes the scanning time, and the light intensity received at a point $w \in G$ equals the scan point density of a surface located at w.

3 The Original Art Gallery Problem

In previous work [1, 4], we have presented an LP-based procedure for the original art gallery problem. AGP can be formulated as an integer linear program with infinitely (actually uncountably) many binary variables and inequalities:

$$\min \quad \sum_{g\in P} x_g \qquad (3)$$

$$\text{s.t.} \quad \sum_{g\in\mathcal{V}(w)} x_g \geqslant 1 \quad \forall w \in P \qquad (4)$$

$$x_g \in \{0,1\} \qquad \forall g \in P \qquad (5)$$

*Braunschweig University of Technology, Germany, {a.kroeller, c.schmidt}@tu-bs.de

This is an extended abstract of a presentation given at EuroCG 2012. 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 in a conference with formal proceedings and/or in a journal.

Figure 1: An optimal solution with fading is not an optimal solution to the original Art Gallery Problem.

The infinite size of the above formulation makes it impossible to solve it directly. Instead, we consider the LP-relaxation obtained relaxing the integrality constraint (5). Furthermore we restrict the guard positions to be from a finite set $G \subset P$, and only require a finite set $W \subset P$ of "witnesses" to be covered. We denote this problem by $\mathrm{AGR}(G, W)$.

$$\min \quad \sum_{g \in G} x_g \tag{6}$$

$$\text{s.t.} \quad \sum_{g \in G \cap \mathcal{V}(w)} x_g \geqslant 1 \quad \forall w \in W \tag{7}$$

$$x_g \geqslant 0 \qquad \forall g \in G \tag{8}$$

We have shown that $\mathrm{AGR}(P, P)$, the LP-relaxation of AGP itself, can be solved efficiently for many problem instances, provided one can solve the associated separation problems. These are:

1. Given a solution $(x_g)_{g \in P}$, decide if it is feasible for $\mathrm{AGR}(G, P)$, i.e., covers the polygon completely, or prove infeasibility by presenting an insufficently covered point.
2. Given a solution for the dual LP of (6)–(8), decide whether it is feasible for the dual of $\mathrm{AGR}(P, W)$, or prove infeasibility by presenting a violated dual constraint. This is equivalent to presenting an additional guard point that will improve the solution.

Our procedure starts with restricted sets G and W and solves $\mathrm{AGR}(G, W)$ using the simplex method, obtaining optimal (for $\mathrm{AGR}(G, W)$) primal and dual solutions. It then solves the two separation problems, and thereby either finds additional points for G and/or W, or proves that the current solution is optimal for $\mathrm{AGR}(P, P)$. In the latter case, the algorithm terminates.

4 Art Gallery with Fading Illumination

Introducing fading constraints (2) to the Art Gallery Problem results in a new problem. We call it *Art Gallery with Fading* (AGPF).

It is tempting to solve AGPF by reducing it to AGP; unfortunately, this suffers from a bound on approximation:

Lemma 1 *AGPF cannot be approximated better than 2^α by first solving the Art Gallery Problem and computing intensities afterwards.*

Proof. Consider the polygon shown in Figure 1 with a long spike of length s. The optimal covering guard set consists of one guard in the lower left corner ($\star$). Illuminating the polygon with it requires an intensity of s^α. An optimal illumination uses two lights ($\star$ and $\bullet$) with intensities 1 and about $(\frac{1}{2}s)^\alpha$. The approximation factor of the first solution is therefore at least

$$\frac{s^\alpha}{1 + (\frac{1}{2}s)^\alpha}\,, \tag{9}$$

which is about 2^α for large s. □

Our goal is to find $(1+\varepsilon)$-approximative solutions for arbitrary $\varepsilon > 0$, by extending the algorithm described in Section 3. Again we restrict the possible guard locations to a set $G \subseteq P$, and only require a set $W \subseteq P$ of witnesses to be sufficiently lit. We denote this problem by $\mathrm{AGPF}(G, W)$; therefore $\mathrm{AGPF}(P, P)$ is the problem of actual interest.

$\mathrm{AGPF}(G, W)$ is modelled by the following LP:

$$\min \quad \sum_{g \in G} x_g \tag{10}$$

$$\text{s.t.} \quad \sum_{g \in G \cap \mathcal{V}(w)} \varrho(g, w) x_g \geqslant 1 \quad \forall w \in W \tag{11}$$

$$x_g \geqslant 0 \qquad \forall g \in G \tag{12}$$

whose associated dual (10)–(12) reads as follows:

$$\max \quad \sum_{w \in W} y_w \tag{13}$$

$$\text{s.t.} \quad \sum_{w \in W \cap \mathcal{V}(g)} \varrho(g, w) y_w \leqslant 1 \quad \forall g \in G \tag{14}$$

$$y_w \geqslant 0 \qquad \forall w \in W \tag{15}$$

As in the general case we are interested in the primal and dual separation problem. Given an optimal solution x^* to (10)–(12) with associated dual solution y^*, the primal separation problem asks for a witness $w \in W$ such that $\sum_{g \in G \cap \mathcal{V}(w)} \varrho(g, w) x_g < 1$. The dual separation problem asks for a guard $g \in G$ with $\sum_{w \in W \cap \mathcal{V}(g)} \varrho(g, w) y_w > 1$. Hence, the primal problem is a minimization, the dual problem a maximization problem.

Note, in contrast to the original Art Gallery problem, we do not seek integer solutions here.

In the following Section 5 we will give an algorithm that for a given ε yields a $(1+\varepsilon)$-approximation to this problem, in case of convergence.

5 Approximation

Instead of using the function ϱ defined in Section 1, we round ϱ down, and use a step function τ. For a

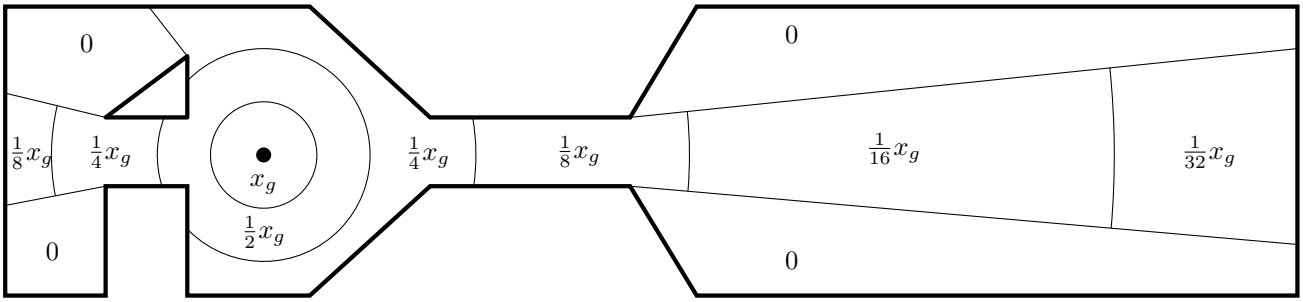


Figure 2: The visibility Arrangement $\mathcal{A}(\{g\})$ induced by $T_{\{g\},x}$ for a point g with value x_g, assuming $\alpha = 1$.

given ε, the function τ is defined as follows:

$$\tau(g,w) := \lfloor\!\lfloor \varrho(g,w) \rfloor\!\rfloor_{1+\varepsilon} , \tag{16}$$

where $\lfloor\!\lfloor \cdot \rfloor\!\rfloor_b$ denotes the hyperfloor with base b, i.e.,

$$\lfloor\!\lfloor x \rfloor\!\rfloor_b := \max\{b^z : b^z \leqslant x,\ z \in \mathbb{Z}\} . \tag{17}$$

Therefore τ is piecewise constant and fulfills

$$\frac{1}{1+\varepsilon}\varrho < \tau \leqslant \varrho . \tag{18}$$

Instead of solving (10)–(12) directly, we solve the system AGPF$\tau(G,W)$ defined as

$$\min \quad \sum_{g\in G} x_g \tag{19}$$

$$\text{s.t.} \quad \sum_{g\in G\cap\mathcal{V}(w)} \tau(g,w)x_g \geqslant 1 \quad \forall w \in W \tag{20}$$

$$x_g \geqslant 0 \qquad \forall g \in G \tag{21}$$

with associated dual LP

$$\max \quad \sum_{w\in W} y_w \tag{22}$$

$$\text{s.t.} \quad \sum_{w\in W\cap\mathcal{V}(g)} \tau(g,w)y_w \leqslant 1 \quad \forall g \in G \tag{23}$$

$$y_w \geqslant 0 \qquad \forall w \in W \tag{24}$$

Lemma 2 *A solution x for AGPF$\tau(G,W)$ is also a solution for AGPF(G,W).*

Proof. This follows directly from (18): As x fulfills all inequalities of type (20), it also fulfills those of type (11). □

Lemma 3 *Let x^* be an optimal solution for AGPF(G,W) with value OPT. Then $(1+\varepsilon)x^*$ is feasible for AGPF$\tau(G,W)$ and has objective value $(1+\varepsilon)$OPT.*

Proof. Again this follows directly from (18): As x^* fulfills all inequalities of type (11), $(1+\varepsilon)x^*$ fulfills those of type (20). □

Corollary 4 *An optimal solution for AGPF$\tau(P,P)$ is an $(1+\varepsilon)$-approximation for AGPF(P,P).*

Proof. An optimal solution $x^{*\tau}$ for AGPF$\tau(P,P)$ has no larger objective value than the feasible solution $(1+\varepsilon)x^*$, i.e., the objective value is bounded by $(1+\varepsilon)$OPT (Lemma 3). Lemma 2 yields feasibility and we obtain the approximation factor. □

5.1 Algorithm

We can use our LP-based AGP procedure, as described in Section 3, to find $(1+\varepsilon)$-approximative solutions for the Art Gallery Problem with Fading Illumination—assuming there are algorithms to solve the two separation problems. These are:

1. Given a solution x for (19)–(21), find a point $w \in P$ with
$$\sum_{g\in G: w\in\mathcal{V}(g)} \tau(g,w)x_g < 1$$
or decide no such point exists.

2. Given a solution y for (22)–(24), find a point $g \in P$ with
$$\sum_{w\in W\cap\mathcal{V}(g)} \tau(g,w)y_w > 1$$
or decide no such point exists.

By the symmetry of visibility, and using the illumination function

$$T_{G,x}(p) := \sum_{g\in G\cap\mathcal{V}(p)} \tau(g,p)x_g , \tag{25}$$

we see that the primal (resp. dual) separation can be solved if we can solve

$$\min_{p\in P} T_{G,x}(p) \quad (\text{resp. } \max_{p\in P} T_{W,y}(p)). \tag{26}$$

It is easy to see that, for a single point g, $T_{\{g\},x}$ is a piecewise constant function over P, as shown in Figure 2. It induces an arrangement $\mathcal{A}(\{g\})$ within P with faces of constant value. P is split into two parts: The first is $P \setminus \mathcal{V}(g)$, where $T_{\{g\},x}$ is constant 0. For

the second part, notice that τ is a monotically decreasing step function taking the values $(1+\varepsilon)^{-z}$ for $z = 0, 1, 2, \ldots$, each at distances $((1+\varepsilon)^{z-1}, (1+\varepsilon)^z]$ from g (exception: $[0, 1]$ for $z = 0$). This introduces $O(\log_{\varepsilon+1} D)$ concentric bands of equal coverage value around g. The second part of the arrangement is therefore the intersection of concentric circles with $\mathcal{V}(g)$.

Now consider the arrangement $\mathcal{A}(G)$ defined by overlaying the individual arrangements for all $g \in G$. Then, $T_{G,x}$ is constant over each face, edge, and vertex of this arrangement.

Therefore, problem (26) can be solved by computing $\mathcal{A}(G)$, and then iterating over all faces, edges, and vertices of it to find a point minimizing (resp. maximizing) $T_{G,x}$. In fact, it is easy to see that the minimum is always attained inside a face, and the maximum either on a vertex or a point in G. This can be used to speed up the search; however the asymptotic runtime does not improve.

5.2 Runtime Complexity

As the algorithm simply enumerates all elements of $\mathcal{A}(G)$, the complexity of this arrangement dominates the runtime. It is constructed from

- n straight line segments defining P,
- $O(n)$ straight line segments defining $\mathcal{V}(g)$ per $g \in G$, and
- $O(\log_{\varepsilon+1} D)$ circles per $g \in G$,

therefore $O((n + \log_{\varepsilon+1} D)|G|)$ straight line segments and circles in total. Therefore the complexity of $\mathcal{A}(G)$ is $O((n + \log_{\varepsilon+1} D)^2|G|^2)$.

Lemma 5 *The separation problems for AGPF$\tau(G, W)$ can be solved in time $O((n + \log_{\varepsilon+1}(\mathrm{diam}P))^2|G|^2)$ resp. $O((n + \log_{\varepsilon+1}(\mathrm{diam}P))^2|W|^2)$.*

5.3 Limitations

We should mention again that the aforementioned approach is *not* an approximation scheme, as it is not guaranteed that the procedure converges to a final solution. In fact, it might simply not terminate at all. However, if the procedure terminates, then it will have found a $(1+\varepsilon)$-approximative solution. We have conducted exhaustive experiments [1] without fading, and found that this happens most of the time.

6 Art Gallery with Fixed Guards and Fading

Finally we consider the case where the lights can only be placed on a fixed set of positions, that is, $G \subset P$ is given as part of the input and we seek to optimize AGPF(G, P). This includes the case of *vertex guards*, where guards can only be placed on polygon vertices.

Theorem 6 *AGPF(G, P) admits a fully polynomial-time approximation scheme (FPTAS).*

Proof. With fixed G, the LP formulation of AGPF$\tau(G, P)$ has a fixed number of columns, but infinitely many rows. It is well-known that the ellipsoid method [3] can be used to find an optimal solution to such an LP, provided there is an oracle to solve the primal separation problem in polynomial time.

In Section 5, we have shown how to implement such an oracle with complexity $O((n + \log_{\varepsilon+1} D)^2|G|^2)$, which is polynomial in the encoded size of the problem input ($\log D$ is polynomial in the encoded coordinates of P's vertices) and in $1/\varepsilon$. Therefore we can find an optimal solution to AGPF$\tau(G, P)$ in polynomial time, which in turn is a $(1+\varepsilon)$-approximation for AGPF(G, P). □

7 Conclusion

We have introduced the Art Gallery Problem with Fading, which is a generalization of both the Art Gallery Problem and the Stage Illumation Problem by Eisenbrand *et al.*[2]. We presented an efficient algorithm to approximate the primal and dual separation problems stemming from a doubly-infinite LP formulation of the problem. It can be used for practical algorithm implementations, and gives rise to an FPTAS for the case with restricted guard positions.

Acknowledgments

This work was supported by the Deutsche Forschungsgemeinschaft (DFG) under contract number KR 3133/1-1 (Kunst!).

References

[1] T. Baumgartner, S. P. Fekete, A. Kröller, and C. Schmidt. Exact solutions and bounds for general art gallery problems. In *Proc. Algorithm Engineering and Experiment (ALENEX '10)*, pages 11–22, 2010.

[2] F. Eisenbrand, S. Funke, A. Karrenbauer, and D. Matijevic. Energy-aware stage illumination. *Proc. Symposium on Computational Geometry (SCG '05)*, pages 336–345, 2005.

[3] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[4] A. Kröller, T. Baumgartner, S. P. Fekete, and C. Schmidt. Exact solutions and bounds for general art gallery problems. *To appear in the Journal of Experimental Algorithms*, 2012.

[5] J. O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, New York, 1987.

A fixed-parameter Algorithm for Guarding 1.5D Terrains

Farnoosh Khodakarami* Farzad Didehvar† Ali Mohades‡

Abstract

In the 1.5D terrain guarding problem we are given a polygonal region in the plane determined by an x-monotone polygonal chain, and the objective is to find the minimum number of guards for the given input terrain. The only previous theoretical results about this problem are about approximation. We turn to fixed-parameter tractability. We study "depth of terrain onion peeling" as parameter and show that 1.5D terrain guarding problem is fixed-parameter tractable with respect to this parameter.

1 Introduction

A 1.5D terrain is a polygonal chain in the plane that is an x-monotone polygonal chain. An x-monotone chain in R^2 is a chain that intersects any vertical line at most once. A terrain T consists of a set of n vertices $v_1, v_2, \dots v_n$, where $v_i = (x_i, y_i)$. The vertices are ordered in increasing order with respect to their x-coordinates. There is an edge connecting each (v_i, v_{i+1}) pair where $i = 1, 2, \dots n-1$. For two points p and q in T, we say that p and q are visible, if the line segment connecting p and q lies entirely above or on the terrain. We write $v_p < v_q$ if v_p is to the left of v_q (i.e., $x_p < x_q$). For any two vertices $v_p, v_q \in T$ with $v_p < v_q$, the subterrain $T_{[p,q]}$ of T is simply the portion of T between v_p and v_q (including v_p and v_q).

A set G of points on the terrain is called a guarding set if every point on the terrain is seen by some point in G. The optimization version of the terrain guarding problem is the problem of finding a minimum guarding set for a given terrain. There are two standard versions of the terrain guarding problem:

1. The discrete version: The goal is to select a minimum set of vertices of terrain as guards.

2. The continuous version: The goal is to select a smallest set of guards on terrain T. This means that guards can be placed anywhere on the terrain.

*Department of Mathematics and Computer Science, Amirkabir University, f.khodakarami@aut.ac.ir

†Department of Mathematics and Computer Science, Amirkabir University, Didehvar@aut.ac.ir

‡Laboratory of Algorithms and Computational Geometry, Department of Mathematics and Computer Science, Amirkabir University of Technology, mohades@aut.ac.ir

A terrain guarding problem is motivated from guarding or covering a road with either security cameras or lights, and also other applications include finding a configuration for line-of-sight transmission networks for radio broadcasting, cellular telephony and other communication technologies [2, 3].

1.1 Previous Work on Terrain Guarding

Chen et al. [1] claimed that 1.5D-terrain guarding problem is NP-hard, but the proof was never completed formally [8]. King and Krohn [2] proved that the decision version of this problem is NP-hard.

Most of the past research has gone into developing approximation algorithms. Ben-Moshe et al. [3] gave the first constant-factor approximation algorithm for the 1.5D-terrain guarding problem. Clarkson and Varadarajan gave another constant factor approximation for the problem based on solving a linear programming relaxation and rounding in [4]. A 4-approximation was proposed by King in [6] but further analysis increased the approximation factor to 5. Elbassioni et al. [5] gave a 4-approximation that also works for the weighted and partial versions of the problem. Gibson et al. obtain a polynomial time approximation scheme for the standard 1.5D terrain guarding problem by using a local search technique.

1.2 Our Result

The visibility graph of T has a node for every vertex of T and an edge for every pair of visible vertices in T. Dominating set is a well-studied NP-complete optimization problem. The minimum dominating set problem in visibility graphs corresponds to the guarding problem in polygons [16]. So to find the 1.5D terrain guarding problem, we can use minimum dominating set of visibility graph of that terrain.

In this paper, we consider the discrete terrain guarding problem. We propose "depth of terrain onion peeling" as parameter and show that the 1.5D terrain guarding problem is fixed-parameter tractable when parameterized by this parameter. To do so, we define terrain onion peeling structure and use this structure to prove the treewidth of 1.5D terrain visibility graph is bounded by linear function of "depth of terrain onion peeling". Then, by using treewidth-based algorithm for minimum dominating set as developed in [9] on terrain visibility graph show that

This is an extended abstract of a presentation given at EuroCG 2012. 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 in a conference with formal proceedings and/or in a journal.

terrain guarding problem is fixed-parameter tractable respect to the "depth of terrain onion peeling".

This paper is organized as follows. Section 2 contains the formal definitions of the fixed-parameter algorithm and tree decomposition which are used in the paper. In Section 3 we introduce the characterization of terrain onion peeling. The main results of the paper are presented in section 4. Finally, Section 5 offers a discussion of our results.

2 Fixed-Parameter Algorithms

A problem with input size n and a positive integer parameter k is fixed-parameter tractable if it can be solved by an algorithm that runs in $O(f(k).n^c)$ time, where f is a computable function depending only on k, and c is a constant independent of k; such an algorithm is (informally) said to run in fpt-time. The class of all fixed-parameter tractable problems is denoted by FPT. We refer to [13, 15, 14] for more information on fixed-parameter algorithms and parameterized complexity.

There are many graph problems that can be solved in linear or polynomial time with a dynamic programming algorithm when the input graph has bounded treewidth. Typically, treewidth-based algorithms proceed in two stages: The first stage finds a tree decomposition of bounded width of the input graph, and the second stage solves the problem using dynamic programming approaches on the tree decomposition. For combinatorial optimization problems, this is a useful approach for obtaining fixed-parameter tractable algorithms.

Definition 1 *A tree decomposition of a graph $G = (V, E)$, is a pair $TD = (\{X_i | i \in I\}, T = (I, F))$ where each node $i \in I$ has associated to it a subset of vertices $X_i \subseteq V$, called the bag of i, such that*

1. *Each vertex belongs to at least one bag.*
2. *For all edges, there is a bag containing both its endpoints, i.e. for all $\{v, w\} \in E$: there is an $i \in I$ with $v, w \in X_i$.*
3. *For all vertices $v \in V$, the set of nodes $\{i \in I | v \in Xi\}$ induces a subtree of T.*

The width of a tree decomposition $(\{X_i \in I\}, T = (I, F))$ is $max_{i \in I}|X_i| - 1$. The treewidth of a graph G is the minimum width over all tree decompositions of G.

3 Terrain onion peeling

The convex hull of a set S of points is the smallest convex polygon P for which each point in S is either on the boundary of P or in its interior. We denote the convex hull of S by $CH(Q)$.

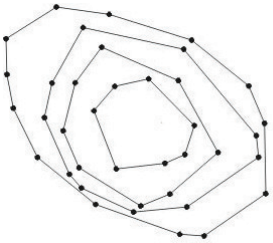


Figure 1: convex onion peeling, the depth of onion is 4.

The convex onion peeling of a set of points is a structure that consists of a sequence of nested convex hulls of points [11, 12]. This structure on a set S of n points is obtained by the following procedure: Compute the $CH(S)$, and let S' be the set of points remaining in the interior of the hull. Compute the $CH(S')$ and recursively repeat this process until no more points remain. One ends up with a sequence of nested convex hulls, called the convex onion peeling of S. The number of layers is called the depth of onion, which is equal to the number of iterations of the convex hull computation process. Figure 1 shows the onion layers and depth of onion.

We used the idea of onion peeling in order to define m-layered terrain. First of all we need to define terrain onion peeling, which is similar to convex onion peeling and obtain by the following procedure: compute $CH(S)$, Let a (resp. b) denote the point of the convex layer of convex hull with minimum (resp. maximum) x-coordinate. The upper chain of $CH(S)$ runs clockwise from a to b. We denote the upper chain of $CH(S)$ by $UCH(S)$. Let S' be the set of points S minus the points on the $UCH(S)$. Compute the $UCH(S')$ and recursively repeat this process until no more points remain. The depth of terrain onion peeling is equal to the number of layers in convex hull computation process, See Figure 2.

Points are divided into several groups (nested layers), using the terrain onion peeling technique. Figure 2 illustrates an example of constructing a terrain onion peeling, which consists of sequence of upper convex hull chains (layers) and each vertex assign to one layer. As you can see in this figure the vertices of terrain are divided into 3 layers. If vertex v_i is assigne to layer m then it is presented by v_i^m.

Lemma 1 *Let v_i^b, v_j^a, v_l^c be vertices of terrain T such that $b, c \geq a$ and $i < j < l$ then v_l^c and v_i^b are invisible pair in a visibility graph.*

Proof. Assume, for the purpose of contradiction, that v_i^b and v_l^c are visible pair, so line segment (v_i^b, v_l^c) lies entirely above or on the terrain. Under the assumption that $i < j < l$, we can conclude (v_i^b, v_l^c) lies above v_j^a.

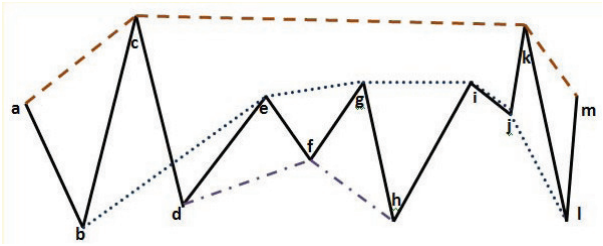


Figure 2: terrain onion peeling: layers are as follow $layer_1 = \{a, c, k, m\}$, $layer_2 = \{b, e, g, i, j, l\}$ and $layer_3 = \{d, f, h\}$, depth of 1.5D terrain onion peeling is 3

Since v_j^a is in layer a, v_j^a is vertex of $UCH(\bigcup_{i \geq a} layer_i)$, which is upper chain of convex hull for the points with layer labels greater than a. Also, v_j^a lies between v_i^b and v_l^c on the terrain T and v_j^a is on the $UCH(\bigcup_{i \geq a} layer_i)$ where $b, c \geq a$. Therefore, v_j^a is above v_i^b and v_l^c, which is a contradiction. □

4 tree decomposition of 1.5D terrain visibility graph

Now that we have presented structural characterization terrain onion peeling, we are ready to see an algorithm for construct tree decomposition of 1.5D terrain visibility graph.

Our Algorithm consists of three main steps:

Step1 Construct the terrain onion peeling and divide vertices into nested layers.

Step2 Construct initial tree decomposition $TD_0 = (\{X_i \in I\}, T = (I, F))$ where T consists of just one node i which $X_i = V$.

Step3 Transform initial tree decomposition into a tree decomposition with width at most $2k$, which is TD_k.

In order to construct TD_k from TD_0, we should perform the following iterative operation:

In iteration l, computes the TD_l based on TD_{l-1} and layer l of onion peeling. In order to construct tree decomposition TD_l from TD_{l-1}, replace each node $i \in I$ of tree decomposition TD_{l-1} with a path of m_i vertices, m_i is number of upper convex hull edges of layer l in bag i. The new bags are decomposed from bag X_i, consider $e = (v_p^l, v_q^l)$ be an edge in convex hull of layer l, then $X_{i_{pq}} = \{v_t^j | v_t^j \in X_i \& j < l\} \cup \{v_t^j | v_t^j \in X_i \& v_t^j \in T_{pq}\}$. Figure 3 illustrates an example of constructing tree decomposition based on sequence of upper convex hull chains (layers) for the terrain in figure 2.

we need to prove that our final result is preserving the tree decomposition properties. The proof is by induction on l which is level of iteration in step3.

For each property, the basis is one node tree decomposition TD_0, which is construct in step2. Verifying that each property holds for TD_0 is trivial. For the inductive step, we assume that the lemma holds for TD_{l-1}:

1. In Step3, the bags of TD_{l-1} are decomposed into subbags, that contain the vertices from previous layer ($lable <= l$) and the vertices of subterrain corresponding to convex hull of level l. So, the union of bags in TD_{l-1} is equal to TD_l.

2. In iteration l of step3, Based on convex hull edges in $layer_l$, vertices which are present into different subterrain are separated into different bags. By lemma 1, these vertices are not visible. This implies that, property 2 holds for TD_l.

3. Let v_p be a vertex in our terrain, we have two cases to consider, depending on whether v_p is in a bag in TD_{l-1} or path of more than one node. In the first case, if the node is in the convex hull of layer l then the set of nodes $\{i \in I | v \in Xi\}$ induces a path of nodes and otherwise it is present only in one node. In the second case, if v_p present in path of more than one nodes, this means that this vertex is in convex hull of previous layers($lable <= l$). So, this vertex will be shown in all nodes of path which are replaced by any nodes in $TD_{[l-1]}$. Thus property 3 of tree decomposition holds for TD_l.

Theorem 2 *Let T be 1.5D terrain and depth of terrain onion peeling is bounded by k then, the treewidth of the 1.5D terrain visibility graph is bounded by $2k$.*

Proof. We must now show that the width of tree decomposition, which we construct, is bounded by $2k$. The bags contains 2 types, vertices from previous layers($lable \leq l$) and the vertices of subterrain corresponding to convex hull of level l.

In the step k of the algorithm the length of vertices in each subterrain is at most 2 because we process the inner convex hull. In order to complete the proof, we need to show that the number of vertices from previous layers($lable \leq k$) is less than $2(k-1)$. We prove this claim by induction on the depth of layers. The basis is easy, since there is no previous layer for $layer_1$. For the inductive step, consider TD_{l-1} has $2(l-2)$ vertices from previous layers. Since in step l of our algorithm the subterrain is contain at most 2 vertices of this layer, we can apply the inductive hypothesis to conclude that each bag of TD_l has at most $2(l-2)+2 = 2(l-1)$ nodes from previous layers. Thus, treewidth of the terrain is $2k$. □

Concept of treewidth provides a powerful tool for determining the fixed-parameter tractability of NP-hard combinatorial optimization problems.

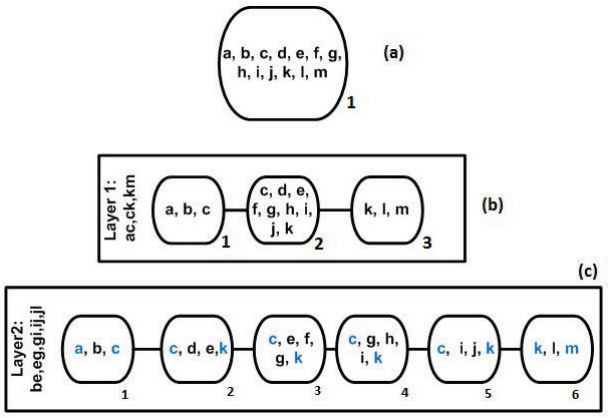


Figure 3: The operation of tree decomposition construction for terrain in Figure 2. Part (a) shows TD_0, which is expanded TD_1 (b) and TD_2 (c) by the iterations of the step3. In the first iteration the node in TD_0 is replaced by the path of 3 nodes in TD_1 based on the convex hull of the $layer_1 = (a, c, k)$. In the next iteration the nodes of TD_1 are replace by paths based on the convex hull of the $layer_2 = (b, e, i, j, l)$, the new bags contains the vertices from the previous layer as well as vertices of subterrain convex hull edges, for example (2-5) in TD_2 contains (c, k), which are in $layer_2$ and present in node 2 of TD_1, and also vertices of subterrains.

The concept of tree decompositions can be used to solve various graph problems. Alber et al. [9] describing fixed-parameter algorithm to solve the minimum dominating set problem constructively in time $O(4^k n)$, for a graph that is given together with a tree decomposition having width k and n nodes.

By theorem 1 and the treewidth-based algorithm for minimum dominating set, we can conclude, 1.5D guarding problem is fixed-parameter tractable with respect to "depth of terrain onion peeling" as parameter.

5 Conclusion

In this paper we show that treewidth of 1.5D terrain onion peeling is bounded by the depth of the terrain onion peeling and present an algorithm for constructing tree decomposition. From this, we derive that 1.5D guarding problem is fixed-parameter tractable respect to "depth of terrain onion peeling" as parameter.

References

[1] D. Z. Chen, V. Estivill-Castro, and J. Urrutia. *Optimal guarding of polygons and monotone chains.* Canadian Conference on Computational Geometry, 1995.

[2] E. Krohn and J. King. *Terrain guarding is NP-Hard.* Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, 1580–1593, 2010.

[3] B. Ben-Moshe, M. J. Katz, and Joseph S. B. Mitchell. *A constant-factor approximation algorithm for optimal 1.5d terrain guarding.* Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms, 515–524, 2005.

[4] K. L. Clarkson, K. Varadarajan. *Improved Approximation Algorithms for Geometric Set Cover.* Discrete & Computational Geometry, vol. 37, 43–58, 2005.

[5] K. Elbassioni, E. Krohn, D. Matijevic, J. Mestre, and D. Severdija. *Improved approximations for guarding 1.5-dimensional terrains.* In Susanne Albers and Jean-Yves Marion, editors, 26th International Symposium on Theoretical Aspects of Computer Science, 2009.

[6] J. King. *A 4-Approximation Algorithm for Guarding 1.5-Dimensional Terrains.* Lecture Notes in Computer Science, vol. 3887, 629–640, 2006.

[7] M. Gibson, G. Kanade, E. Krohn, K. Varadarajan. *An Approximation Scheme for Terrain Guarding.* Proceedings of the 12th International Workshop and 13th International Workshop on Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 140–148, 2009.

[8] E. D. Demaine and J. O'Rourke. *Open problems from cccg 2005.* In CCCG, 2005.

[9] J. Alber, H. L. Bodlaender, H. Fernau, T. Kloks, and R. Niedermeier. *fixed-parameter algorithms for Dominating Set and related problems on planar graphs.* Algorithmica, vol. 33 part 4, 461–493, 2002.

[10] R. Niedermeier. *Reflections on multivariate algorithmics and problem parameterization.* Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science, vol. 5 of LIPIcs, 17-32, 2010.

[11] B. Chazelle. *On the convex layers of a planar set.* IEEE Transactions on Information Theory, vol. 31, 509–517, 1985.

[12] F. Preparata, M. Shamos, *Computational Geometry: An Introduction.* Texts and Monographs in Computer Science. Springer-Verlag, New York, 1985.

[13] R.G. Downey, M. R. Fellows *Parameterized Complexity* Springer-Verlag, 1999.

[14] J. Flum and M. Grohe. *Parameterized Complexity Theory.* Springer-Verlag, 2006.

[15] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms.* Oxford University Press, 2006.

[16] S. Ghosh *Visibility algorithms in the plane.* Cambridge University Press, Cambridge, United Kingdom, 2007.

Partial Searchlight Scheduling is Strongly PSPACE-complete

Giovanni Viglietta*

Abstract

The problem of searching a polygonal region for an unpredictably moving intruder by a set of stationary guards, each carrying an orientable laser, is known as the Searchlight Scheduling Problem. Determining the complexity of deciding if the entire area can be searched is a long-standing open problem. Recently, the author introduced the Partial Searchlight Scheduling Problem, in which only a given subregion of the environment has to be searched, and proved that its 3-dimensional decision version is **PSPACE**-hard, even when restricted to orthogonal polyhedra.

Here we extend and refine this result, by proving that 2-dimensional Partial Searchlight Scheduling is strongly **PSPACE**-complete, both in general and restricted to orthogonal polygons in which the region to be searched is a rectangle.

1 Introduction

The *Searchlight Scheduling Problem* (SSP), first studied in [3], is a pursuit-evasion problem in which a polygon has to be searched for a moving intruder by a set of stationary guards. The intruder moves unpredictably and continuously with unbounded speed, and each guard carries an orientable *searchlight*, emanating a 1-dimensional ray that can be continuously rotated about the guard itself. The polygon's exterior cannot be traversed by the intruder, nor penetrated by searchlights. The intruder is caught whenever it is hit by a searchlight. Because the intruder's location is unknown until it is actually caught, each guard has to sway its searchlight according to a predefined *schedule*. If the guards always catch the intruder, regardless of its path, by following their schedules in concert, they are said to have a *search schedule*.

SSP is the problem of deciding if a given set of guards has a search schedule for a given polygon (with holes). The computational complexity of this decision problem has been only marginally addressed in [3], but has later gained more attention, until in [2] the space of all possible schedules has been shown to be discretizable and reducible to a finite graph, which can be explored exhaustively in order to find a search schedule, if one exists. Since the graph may have double exponential size, this technique easily places SSP in **2-EXP**. Whether SSP is **NP**-hard or even in **NP** is left in [2] as an open problem.

More recently, in [4], the author introduced the *Partial Searchlight Scheduling Problem* (PSSP), in which the guards content themselves with searching a smaller subregion given as input. That is, a search schedule should only guarantee that the given *target region* is eventually cleared, either by catching the intruder or by confining it outside. A 3-dimensional variation of PSSP is studied in [4], in which the input polygonal environment is replaced by an orthogonal polyhedron, and the 1-dimensional rays become 2-dimensional half-planes, which rotate about their boundary lines. Such a problem is shown to be strongly **PSPACE**-hard.

In the present paper we take a further step along this line of research, by proving that 2-dimensional PSSP is strongly **PSPACE**-complete, both in general and restricted to orthogonal polygons in which the region to be searched is a rectangle.

2 Asynchronous NCL machines

Our reduction is based on a model of computation described in [1] and [4], called *Nondeterministic Constraint Logic* (NCL).

An *asynchronous NCL machine* is a 3-regular graph, each of whose vertices is either an *AND vertex* or an *OR vertex*. Of the three edges incident to an AND vertex, one is called its *output edge*, and the other two are its *input edges*. Each edge of an asynchronous NCL machine can be oriented toward either one of its incident vertices (or none), and a configuration of edge orientations is *legal* if

- for each AND vertex, either its output edge is directed inward, or both its input edges are directed inward;
- for each OR vertex, at least one of its three incident edges is directed inward.

Accordingly, a *legal move* consists in the reversal of an edge's orientation that preserves legality of the configuration. Moves can occur at any time independently (i.e., *asynchronously*), and each reversal of an edge can take an arbitrarily long but finite time, during which that edge is not oriented toward any vertex.

Given an asynchronous NCL machine with two *distinguished edges* e_a and e_b, and a *target orientation*

*Department of Computer Science, University of Pisa, viglietta@gmail.com

for each, we call EE-ANCL the problem of deciding if there exist legal configurations A and B such that e_a has its target orientation in A, e_b has its target orientation in B, and there is an asynchronous sequence of legal moves from A to B. EE-ANCL is shown to be **PSPACE**-complete in [4], by a reduction from its *synchronous* version, thoroughly studied in [1].

3 PSPACE-completeness of PSSP

To prove that PSSP belongs to **PSPACE** we use the discretization technique of [2], and to prove that PSSP is **PSPACE**-hard we give a reduction from EE-ANCL.

Lemma 1 *PSSP* $\in$ ***PSPACE***.

Proof. As explained in [2], a technique known as *exact cell decomposition* allows to reduce the space of all possible schedules to a finite graph G. Each searchlight has a linear number of *critical angles*, which yield an overall partition of the polygon into a polynomial number of *cells*. Searchlights take turns moving, and can stop or change direction only at critical angles. Thus, a vertex of G encodes the *status* of each cell (either *contaminated* or *clear*) and the critical angle at which each searchlight is oriented.

As a consequence, G can be navigated nondeterministically by just storing one vertex at a time, which requires polynomial space. Notice that deciding if two vertices of G are adjacent can be done in polynomial time: An edge in G represents a move of a single searchlight between two consecutive critical angles, and the updated status of each cell can be easily evaluated. Indeed, cells' vertices are intersections of lines through input points, hence their coordinates can also be efficiently stored and handled as rational expressions involving the input coordinates.

Now, in order to verify that a path in G is a witness for SSP, one checks if the last vertex encodes a status in which every cell is clear. But the very same cell decomposition works also for PSSP: The analysis in [2] applies even if just a subregion of the polygon has to be searched, and a path in G is a witness for PSSP if and only if its last vertex encodes a status in which every cell that has a non-empty intersection with the target subregion is clear.

Since **PSPACE** = **NPSPACE**, due to Savitch's theorem, our claim follows. □

For the **PSPACE**-hardness part, we first give a reduction in which the target region to be cleared is an orthogonal hexagon. Then, Section 4 will explain how we would have to modify our construction, should we insist on having a rectangular (hence convex) target region.

Lemma 2 *EE-ANCL* $\preceq_{\boldsymbol{P}}$ *PSSP restricted to orthogonal polygons.*

Proof. We show how to transform a given asynchronous NCL machine G with two distinguished edges e_a and e_b into an instance of PSSP.

A rough sketch of our construction is presented in Figure 1. All the vertices of G are placed in a row (a), and are connected together by a network of thin *corridors* (b), turning at right angles, representing edges of G. Each *subsegment* of a corridor is a thin rectangle, containing a *subsegment guard* in the middle (not shown in Figure 1). Two subsegments from different corridors may indeed cross each other like in (c), but in such a way that the crossing point is far enough from the ends of the two subsegments and from the two subsegment guards (so that no subsegment guard can see all the way through another subsegment). All the vertices of G and all the *joints* between consecutive subsegments (i.e., the turning points of each corridor) are connected via extremely thin *pipes* (d) to the upper area (e), which contains the target region (shaded in Figure 1).

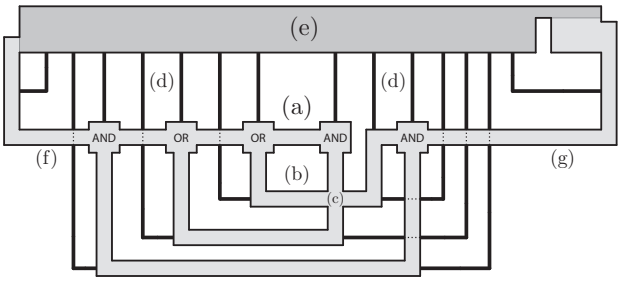


Figure 1: Construction overview.

Two corridors (f) and (g) also reach the upper area, and they correspond to the distinguished edges of G, e_a and e_b, respectively. That is, if $e_a = \{u, v\}$, and the target orientation of e_a is toward v, then the corridor corresponding to e_a connects vertex u in our construction to the upper area (e), rather than to v. The same holds for e_b. Indeed, observe that we may assume that e_a and e_b are reversed only once (respectively, on the first and last move) in a sequence of moves that solves EE-ANCL on G. As a consequence, contributions to vertex constraints given by distinguished edges oriented in their target direction may be ignored.

Each pipe turns at most once, and contains one *pipe guard* in the middle, lying on the boundary. Notice that straight pipes never intersect corridors, but some turning pipes do. Figure 2 shows a turning pipe, with its pipe guard (a) and an intersection with a corridor (b) (proportions are inaccurate). The *intersection guards* (c) separate the pipe from the corridor with their lasers (dotted lines in Figure 2), without "disconnecting" the pipe itself. Although a pipe narrows every time it crosses a corridor, its pipe guard

can always see all the way through it, because it is located in the middle. The small *nook* (d) is unclearable because no guard can see its bottom, hence it is a constant source of recontamination for the target region (e), unless the pipe guard is covering it with its laser. (Each straight pipe also has a similar nook.)

In our construction, corridor guards implement edge orientations in G: Whenever all the subsegment guards in a corridor connecting vertices u and v have their lasers oriented in the same "direction" from vertex u to vertex v, it means that the corresponding edge $\{u, v\}$ in G is oriented toward v.

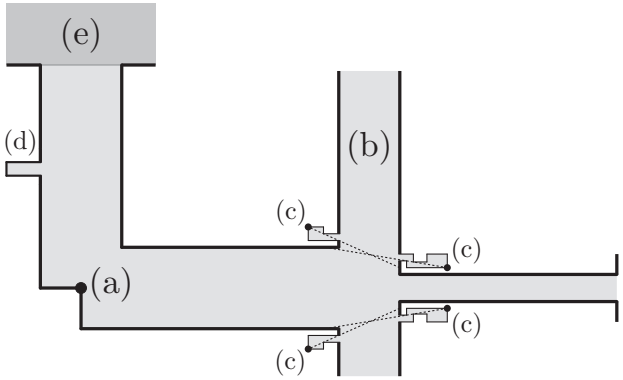


Figure 2: Intersection between a pipe and a corridor.

Figure 3 shows an OR vertex. The three subsegment guards from incoming corridors (a) can all "cap" pipe (b) with their lasers, and nook (c) guarantees that the pipe is recontaminated whenever all three guards turn their lasers away.

AND vertices are implemented as in Figure 4. The two subsegment guards (a) correspond to input edges, and are able to cap one pipe (e) each, whereas guard (c) can cover them both simultaneously. But that leaves pipe (d) uncovered, unless it is capped by guard (b), which belongs to the corridor corresponding to the output edge. Again, uncovered pipes are recontaminated by unclearable nooks (f).

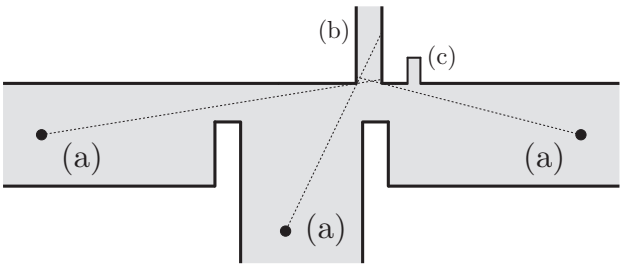


Figure 3: OR vertex.

Joints between consecutive subsegments of a corridor may be viewed as OR vertices with two inputs, shaped like in Figure 3, but without the corridor coming from the left.

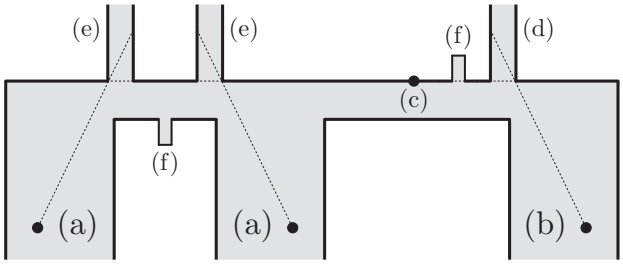


Figure 4: AND vertex.

Finally, Figure 5 shows the upper area of the construction, reached by the distinguished edges e_a and e_b (respectively, (a) and (b)), and by all the pipes (c). The guard in (d) can cap all the pipes, one at a time, and its purpose is to clear the left part of the target region, while the small rectangle (e) on the right will be cleared by the guard in (f). The two pipes (g) implement additional OR vertices with two inputs, and prevent (d) and (f) from acting, unless the respective distinguished edges are in their target orientations. Nook (h) will contaminate part of the target region, unless (d) is aiming down. Nooks (i) prevent area (e) from staying clear whenever guard (f) is not aiming up. The guard in (j) separates the two parts of the target region with its laser, so that they can be cleared in two different moments.

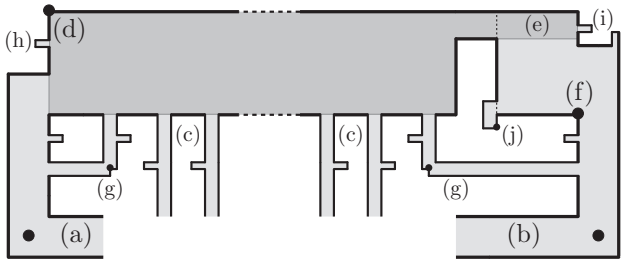


Figure 5: Target region.

Suppose G is a solvable instance of EE-ANCL. Then we can "mimic" the transition from configuration A to configuration B (see Section 2) by turning subsegment guards. Specifically, if edge $e = \{u, v\}$ in G changes its orientation from u to v, then all the subsegment guards in the corridor corresponding to e turn their lasers around, one at a time, starting from the guard closest to u. Before this process starts, each pipe has one end capped by some subsegment guard, and in particular pipe (g) on the left of Figure 5 is capped by the guard in (a). Hence, guard (d) is free to turn and cap all the pipes one by one, stopping for a moment to let each pipe's internal guard clear the pipe itself (which now has both ends capped) and cover its nook (see Figure 2). As a result, the left part of the target region can be cleared by rotating (d) clockwise, from right to down. Then the subsegment

guards start rotating as explained above, until configuration B is reached. If done properly, this keeps all the pipes capped and clear, thus preventing the left part of the target region from being recontaminated. When B is reached, guard (f) can turn up to clear (e) and finally solve our PSSP instance.

Conversely, suppose that G is not solvable. Observe that rectangle (e) in Figure 5 has to be cleared by guard (f) as a last thing, because it will be recontaminated by nooks (i) as soon as (f) turns away. On the other hand, as soon as a pipe has both ends uncapped by external guards, some portion of the target region necessarily gets recontaminated by some nook, regardless of where the pipe guard is aiming its laser. But guard (d) can cap just one pipe at a time and, while it does so, nook (h) keeps some portion of the target region contaminated. Thus, the entire process must start from a configuration A in which all the pipes are simultaneously capped and guard (d) is free to turn right (i.e., e_a is in its target orientation), then proceed without ever uncapping any pipe (i.e., preserving legality), and finally reach a configuration B in which guard (f) is free to turn up (i.e., e_b is in its target orientation). By assumption this is impossible, hence our PSSP instance is unsolvable. □

By putting together Lemma 1 and Lemma 2, we immediately obtain the following:

Theorem 3 *Both PSSP and its restriction to orthogonal polygons are strongly **PSPACE**-complete.* □

The term "strongly" is implied by the fact that all the vertex coordinates generated in the **PSPACE**-hardness reduction of Lemma 2 are numbers with polynomially many digits (or can be made so through negligible adjustments).

4 Convexifying the target region

We can further improve our Theorem 3 by making the target region in Lemma 2 rectangular.

Our new target region has the same width as the previous one, and the height of rectangle (e) in Figure 5. In order for this to work, we have to make sure that some portion of the target region is "affected" by each contaminated pipe that is not capped by guard (d), no matter where *all* the pipe guards are oriented. To achieve this, we make pipes reach the upper area of our construction at increasing heights, from left to right, in a staircase-like fashion.

Assume we already placed pipe (a) in Figure 6, and we need to find the correct height at which it is safe to connect pipe (b). First we find the rightmost intersection (c) between a laser emanating from the pipe guard of (a) and the lower border of the target region. Then we set the height of pipe (b) so that it is capped by guard (d) when it aims slightly to the right of (c).

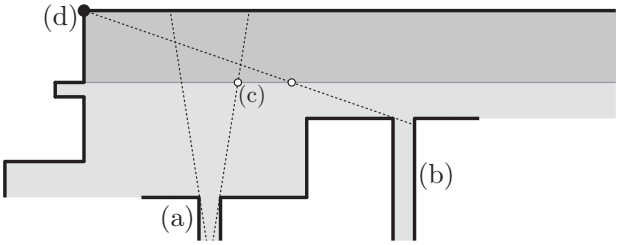


Figure 6: Rectangular target region.

This is always feasible, provided that pipes are thin enough, which is not an issue.

After we have set all pipes' heights from left to right, the construction is complete and the proof of Lemma 2 can be repeated verbatim, yielding:

Theorem 4 *Both PSSP and its restriction to orthogonal polygons with rectangular target regions are strongly **PSPACE**-complete.* □

5 Further research

Observe that the target region constructed in [4] for 3-dimensional PSSP is an arbitrarily small ball centered at a given point. We could even modify PSSP by asking if any neighborhood of a given point is clearable at all (as opposed to a well-defined polygonal target region), and this problem would stay **PSPACE**-hard for polyhedral environments. Our question is whether this holds true for 2-dimensional polygons, as well.

Similarly, we may investigate the complexity of PSSP on other restricted inputs, such as simply connected polygons, or target regions coinciding with the whole environment. The latter is in fact SSP, which has been mentioned in Section 1 as an interesting long-standing open problem. In [4], the author proved that the 3-dimensional version of SSP is **NP**-hard, but determining the true complexity of either version still seems a deep problem and a possibly laborious task.

References

[1] R. A. Hearn and E. D. Demaine. *PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation.* Theor. Comput. Sci. **343** (2005), 72–96, special issue "Game Theory Meets Theoretical Computer Science."

[2] K. J. Obermeyer, A. Ganguli, and F. Bullo. *A complete algorithm for searchlight scheduling.* Int. J. Comput. Geom. Ap. **21** (2011), 101–130.

[3] K. Sugihara, I. Suzuki, and M. Yamashita. *The searchlight scheduling problem.* SIAM J. Comput. **19** (1990), 1024–1040.

[4] G. Viglietta. *Searching polyhedra by rotating half-planes.* Preprint (2011), available at `http://arxiv.org/abs/1104.4137`.

Triangle-Triangle Tolerance Tests

Rainer Erbes* Anja Mantel* Elmar Schömer† Nicola Wolpert*

Abstract

The motivation for this work comes from the digital mock-up process in car industry where constructions and motions in the design of a car are evaluated. One important task is to verify that moving components keep a given distance to other (stationary) components in their environment. The geometry of a component is described by a high-quality triangle mesh and its trajectory is specified by a discrete sequence of positions and orientations. Solving this task, the basic operation is a triangle-triangle tolerance test: Given two triangles, determine whether their shortest Euclidean distance is greater than a given tolerance value ε. A special case is the triangle-triangle intersection test choosing $\varepsilon = 0$. For the latter a lot of work has been done to solve it efficiently [2, 3, 4, 5]. We present efficient solutions for the generalized question. Our solutions are optimized for the treatment of millions of triangle-triangle tolerance tests in parallel on multi-core CPUs and on the GPU.

1 Introduction and previous work

In the digital mock-up, early in the construction phase of a car, constructions and motions of component parts have to be evaluated. For example it has to be checked that in the moving car the vibrating engine always keeps a given safety distance to its circumjacent parts. An example for a wide motion is the engine marriage. It has to be verified that the assembly of the engine into the car body is possible in compliance with a safety distance. In our work we consider two objects, a static and a dynamic one. The boundary of an object is given by a list of triangles, the motion by a sequence of affine transformations. For a vibrating motion we are interested in all triangles of both objects that violate a given ε-distance to the other part during motion. For a wide motion we want this information for every time step separately.

Every solution to this tolerance query will mainly consist of two parts: In a "broad phase" those pairs of triangles are identified, with the help of data structures like a bounding volume hierarchy or a grid, which are candidates to fall below the tolerance. In the "narrow phase" every candidate pair then has to pass the triangle-triangle tolerance test: Determine whether the shortest Euclidean distance of two triangles is greater than the tolerance value ε.

A special case is the triangle-triangle intersection test choosing $\varepsilon = 0$. For the intersection query a lot of work has been done to solve it efficiently [2, 3, 4, 5].

In the literature very little can be found about the generalized question. Geometrically, testing two triangles T_1 and T_2 for distance greater ε is much more involved than testing them for intersection. Consider triangle T_1. Inflate every vertex of T_1 to a sphere of radius ε and take the convex hull H_1 of the three spheres. For $\varepsilon > 0$ the boundary of H_1 consists of parts of planes, spheres and cylinders, see also Figure 1. The triangles T_1 and T_2 have a distance greater ε iff H_1 and T_2 do not intersect.

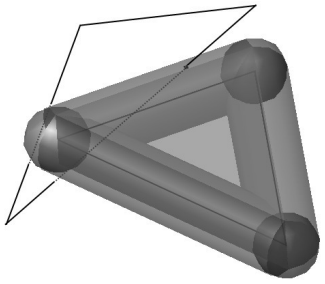


Figure 1: A triangle and the ε-offset of a second triangle intersect. $\Leftrightarrow$ The tolerance value ε between the two triangles is violated.

The proximity query package PQP [8] provides, besides collision detection and distance calculation algorithms, a tolerance verification algorithm. It calculates whether the distance between two complex objects falls below a given tolerance value. The broad phase is realized by using bounding volume hierachies of oriented bounding boxes as well as by rectangular swept spheres. Details are given by Gottschalk et al. [7] and by Larsen at al. [6]. For the narrow phase the minimal distance between the triangles is calculated and compared with ε. PQP stops computation as soon as one pair of triangles is closer than ε. This is different to our problem, because we are interested in all triangles (for all motion steps) that violate the tolerance value. Since this causes much more computing effort, our solutions are optimized for the treatment of millions of triangle-triangle tolerance tests in parallel on multi-core CPUs and on the GPU.

*Hochschule für Technik Stuttgart,
rainer.erbes/anja.mantel/nicola.wolpert@hft-stuttgart.de
†Johannes Gutenberg-Universität Mainz,
schoemer@informatik.uni-mainz.de

2 Ratio between broad and narrow phase

We have implemented two algorithms for the broad phase. One is designed for a vibrating motion and works on the GPU. It uses so called "speed boxes" as proposed by Busold [1]. We are given one static and one moving object. We enclose every static triangle in its AABB and then enlarge the AABB by the tolerance value ε. For every dynamic triangle an AABB is constructed that bounds the triangle for all steps of the motion sequence. Finally, all pairs of static and dynamic intersecting AABBs are determined. The input of the narrow phase is a list of pairs of static and dynamic triangles. Each triangle pair has to be tested for tolerance violation during the vibrating motion.

The second broad phase algorithm is based on a spatial subdivision and works on the multi-core CPU. Both objects are stored in a spatial grid. For every motion step the grid cells are determined which are closer than ε. The input of the narrow phase is a list of pairs of triangles together with an affine transformation. Every transformed triangle pair has to be tested for tolerance violation.

In a first experiment we use the CPU code of the triangle-triangle distance test provided by PQP on the multi-core CPU to execute the tolerance test in the narrow phase for both implementations.

We analyze the ratio between the broad and the narrow phase. In order to have a clean test scenario, we determine in the broad phase the lists of triangle-pairs to be tested for several specific ε. We then write every list into a file and remove double entries. We read the list from the file and start the narrow phase. The grid algorithm and the tolerance test are performed on the multi-core CPU (Intel Core i7-980X, 3.33GHz, 6 Cores + hyper-threading), the speed box algorithm on the GPU (NVIDIA GeForce GTX 480).

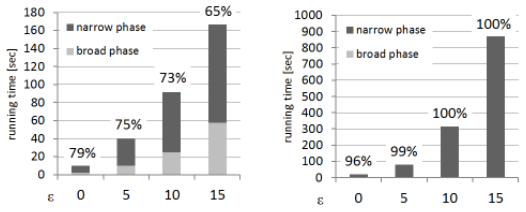


Figure 2: Ratio between narrow and broad phase for a wide (left) and a vibrating motion (right).

The benchmark scenario consists of an engine mount with 24,000 triangles, an exhaust system with 224,000 triangles, and a respective motion track. The left diagram in Figure 2 shows the results for a wide motion with 2,000 discrete motion steps. On the x-axis we vary the tolerance value ε. Both, the grid-based broad and the narrow phase, are calculated on the multi-core CPU. The grid cells have size 5 because this value leads to the best overall running time. It can be seen that a great amount of the running time is spent on the triangle-triangle tolerance test. For a vibrating motion with 14,000 discrete motion steps, the fraction of the narrow phase is even larger (right diagram). This is due to the fact that we calculate the broad phase using speed boxes on the GPU whereas the triangle-triangle test still runs on the multi-core CPU.

It has to be mentioned again that the aim of the PQP tolerance test, which uses a triangle-triangle distance test, is to decide tolerance violations between two objects, not to compute all involved triangles. Our conclusion is that it is inevitable to look for a specialized efficient triangle-triangle tolerance test to process millions of tests in parallel.

3 The narrow phase

In the following we will investigate several triangle-triangle tolerance tests and compare them. Each test deals with one pair of triangles for a specific transformation. The tests we present differ by the kind of "early out" they use. They apply different strategies to shorten computation, either if for sure there is a tolerance violation or if for sure there is none.

3.1 Feature distance

Our first basic tolerance test is to calculate the distance between two triangles and then compare it with the tolerance value. We compute the minimal distance between two triangles by checking all feature distances. Every triangle has 7 features: 3 vertices, 3 edges, and 1 face. For the two triangles we compute all 9 vertex-vertex distances, 18 vertex-edge distances, 9 edge-edge distances, 6 point-face distances, and 6 edge-face intersections. The 48 feature distances are computed one by one and each intermediate result is compared with the tolerance value ε. As soon as one distance is smaller or equal to ε, computing the remaining feature distances can be omitted. Thus, we have an early out as soon as we know that the triangles violate the tolerance.

3.2 Separating plane

Another approach is based on the separating axis theorem. It relies on the fact that two convex objects are disjoint iff there is a plane that separates the objects. Gottschalk et al. [7] use a separating axis test to check two oriented bounding boxes for intersection. They limit the number of planes to be investigated to only 15. This comes from the fact that for two disjoint convex polytopes there exists a separating plane parallel to one of the polytopes' faces or parallel to a pair of edges, one edge from each polytope.

We cannot apply this technique to our triangle-triangle tolerance test in the same manner because

in order to test triangles T_1 and T_2 for tolerance violation we have to test T_2 and the ε-offset H_1 of T_1 for intersection. As seen before, H_1 is not a polytope.

We limit the number of planes by considering all the directions where possibly the shortest distance between the triangles can occur. If H_1 and T_2 are disjoint, the shortest distance between T_1 and T_2 is greater than ε and there exists a separating plane orthogonal to the shortest connection of the two closest features. Thus, we test the planes orthogonal to the 9 vertex-vertex connections, the planes orthogonal to the 2x9 vertex-edge connections and the planes orthogonal to the 9 edge-edge connections. Additionally, we test the two supporting planes of the triangles. This covers the case that a closest feature is the face of one triangle. So we limit the number of contemplable planes to 38. A single separating plane test projects the two triangles onto the normal of the plane and tests if the resulting intervals have a distance greater than ε. As soon as a separating plane is found, the remaining planes do not have to be tested any more. Thus, we have an early out as soon as we know that the triangles do not violate the tolerance.

3.3 Combined

We have seen two approaches with different early outs. Next we combine the separating plane approach with some ideas from the feature distance approach to have both kinds of early outs. Before trying to construct the separating plane orthogonal to a vertex-vertex connection, we first evaluate the distance between the two vertices. Thus, we have an early out in case the vertex-vertex connection already violates the tolerance and we have an early out if there is a separating plane perpendicular to the vertex-vertex connection. In the edge-vertex case, we first evaluate the length of the edge-vertex connection (provided its endpoint is inside the edge) for the test on tolerance violation and then use the direction of the edge-vertex connection for the separating plane test.

3.4 Dual Test

Our next approach works in dual space. It performs several separating plane tests at a time.

Remember that H_1 is the convex hull of three spheres. W.l.o.g. we assume the center point of one sphere to be the origin. Then the dual of H_1 is the intersection of a sphere and two quadrics containing the origin, denoted as H_1^D, see also [9]. Dualizing the vertices of the triangle T_2 leads to three planes. Consider a separating plane p in primal space between H_1 and T_2 and let P_o be the halfspace of p containing the origin. Due to our construction, H_1 is inside P_o. T_2 and therefore all its three vertices are outside P_o. Dualizing p leads to a point p^D inside H_1^D. Looking at the three planes of the dualized vertices of T_2, the point p^D is outside all three halfspaces containing the origin and therefore inside the intersection T_2^D of all three halfspaces not containing the origin. We can conclude that there exists a separating plane between H_1 and T_2 iff H_1^D intersects T_2^D.

Our test is organized as follows: First, for the vertex-vertex test a separating plane and tolerance violation test is performed as described in 3.3. Then we test whether there is a separating plane between H_1 and T_2 which is parallel to one of the edges of T_2. A separating plane parallel to an edge of T_2 exists iff one of the edges (all of them are half-lines) of T_2^D intersects H_1^D. Analogously we look whether there is a separating plane between H_2 and T_1 parallel to one of the edges of T_1 and test the half-lines of T_1^D for intersection with H_2^D.

This summarizes the separating plane tests for the 2x9 vertex-edge connections, the 9 edge-edge connections, and the 2 normal directions to 2x3 intersection tests of a half-line with three quadrics. For an intersection test of a half-line with three quadrics we use interval calculation and Sturm sequences.

4 Experimental Results

We have implemented our triangle-triangle tolerance tests and developed different optimized versions for the multi-core CPU and for the GPU. We use the same test scenarios as described in Section 2.

Wide motion: For each motion step we are interested in all tolerance violating triangles. Because of this task, a tolerance test for two triangles is superfluous and not executed if both triangles have been detected as tolerance violating for the same motion step before. We call this a "logic ε-violation". Figure 3 (a) presents our performance results for the wide motion path with 2000 motion steps. On the path 400 successive steps cause a collision of the two objects. We zoom into this part of the path, again sample 2000 motion steps, and show the results in (b).

Due to the construction of the two paths, the broad phase in (b) collects more triangle pairs. In both cases a high percentage of the triangle pairs have a logic ε-violation and almost all remaining ones (and only their tolerance tests are measured for the running time) have no violation and therefore a separating plane. The difference in the running time beween the combined and the dual approach on the one hand and the separating plane approach on the other hand is caused by the different order in which they consider the feature connections. The first two start with vertex-vertex, the latter with the triangle planes.

Vibrating motion: We are interested in all triangles which violate tolerance during the motion. Diagram 4 (a) shows the number of triangle pairs to be checked in the narrow phase. Each pair has to pass 14,000 triangle-triangle tolerance tests, one for each

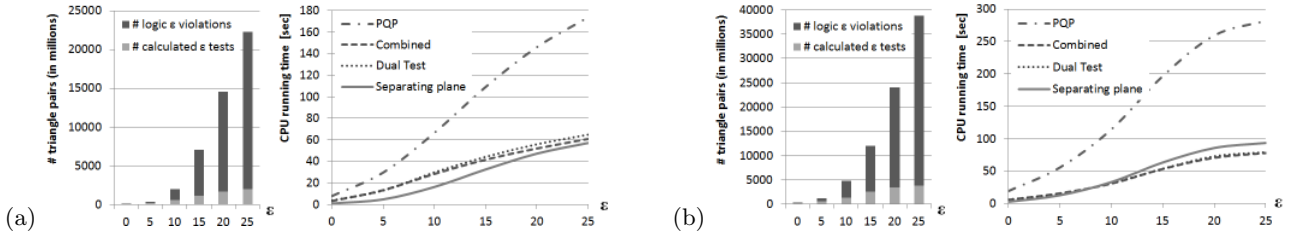


Figure 3: Number of triangle pairs after the broad phase and performance of different tolerance tests on the multi-core CPU for (a) the complete wide motion, (b) a part of the motion where the two objects intersect

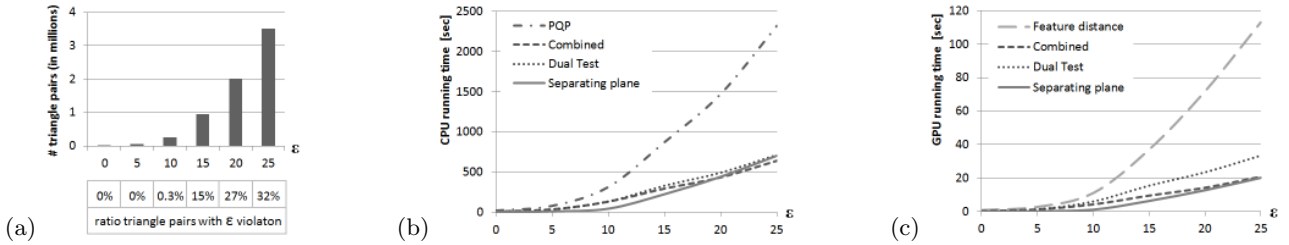


Figure 4: Performance result vibrating motion: (a) Number of triangle-triangle tolerance tests, (b) Performance of different tolerance tests on the multi-core CPU and (c) the GPU

motion step, but we stop computation as soon as we find the first tolerance violation. With increasing ε also the percentage of triangle pairs with tolerance violation increases. The next two diagrams show the performance of the algorithms on the multi-core CPU (b) and the GPU (c). For small ε, and therefore for a small number of ε-violations, the separating plane approach is the best. The more violations we have, the better becomes the combined approach.

Figure 5 proves that the performance of the tolerance query between two complex objects can be improved significantly by using an optimized triangle-triangle tolerance test. The only difference to Figure 2 is that we now use our implementations of the separating plane approach to execute the triangle-triangle tolerance tests. This causes a reduction of the time spend on the tolerance tests and a decrease of the overall running time: For the wide motion (left), which is completely computed on the multi-core CPU, we are in average 2.5 times faster. For the vibrating motion the broad phase is computed on the GPU. Here we are in average 6 times faster if we use our optimized tolerance test on the CPU and even 103 times faster if we let both phases run on the GPU (right).

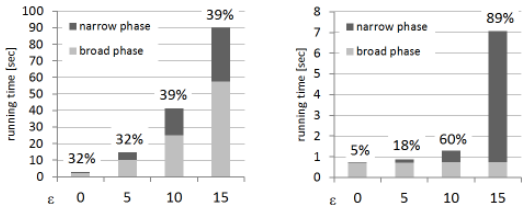


Figure 5: Ratio between narrow and broad phase for a wide (left) and a vibrating motion (right) with optimized triangle-triangle tolerance tests.

References

[1] C. Busold. *Parallele Abstandsberechnung zwischen bewegten und festen Dreiecken* Bachelor thesis, University of Mainz, 2009

[2] T. Möller. *A fast triangle-triangle intersection test.* Journal of Graphics Tools, 1997; 2(2): 2530.

[3] M. Held. *ERIT - A Collection of Efficient and Reliable Intersection Tests.* Journal of Graphics Tools 1997; 2(4): 2544.

[4] P. Guigue and O. Devillers. *Fast and robust triangle-triangle overlap test using orientation predicates.* Journal of Graphics Tools 2003; 8(1): 2542.

[5] O. Tropp, A. Tal, and I. Shimshoni. *A fast triangle to triangle intersection test for collision detection.* Journal of Visualization and Computer Animation, 17(5):527535, 2006.

[6] E. Larsen, S. Gottschalk, M. Lin, and D. Manocha. *Fast proximity queries with swept sphere volumes.* Technical Report TR99-018, Department of Computer Science, University of North Carolina, 1999.

[7] S. Gottschalk, M. Lin, and D. Manocha. *OBB-Tree: A Hierarchical Structure for Rapid Interference Detection.* ACM SIGGRAPH, 1996.

[8] E. Larsen, S. Gottschalk, M. Lin, and D. Manocha. *PQP - A Proximity Query Package* http://gamma.cs.unc.edu/SSV/, Copyright 1999.

[9] N. Geismann, M. Hemmer, and E. Schömer: *The Convex Hull of Ellipsoids (Video).* 17th ACM Symposium on Computational Geometry, pp. 321-322

Optimizing the computation of sequences of determinantal predicates

Ioannis Z. Emiris* Vissarion Fisikopoulos* Luis Peñaranda*

Abstract

Orientation is the core predicate in many important geometric algorithms, such as convex hull and triangulation computations. This operation reduces to compute the sign of a determinant. We propose a method that improves the amortized complexity of the determinants involved in a convex hull computation. Moreover, we study how can we use the computation done in a convex hull construction to improve the construction of subsequent convex hulls. Our two main tools are the dynamic determinant computation and the reuse of determinantal minors. We finally validate our approach by implementing and testing one of our methods.

Keywords Orientation, determinant, convex hull, triangulation, CGAL implementation.

1 Introduction

In general dimension d, the orientation of $d+1$ points is computed as the determinant of a matrix containing the coordinates of the points as columns, plus one last row full of ones. As the dimension grows, a higher percentage of the computation time is consumed by these predicates. In this paper we study the computation of sequences of determinantal predicates in a single, or in a sequence of, *exact* convex hull computations. A typical example of the latter is the computation of many regular triangulations of the same pointset, which arise in algorithms that compute secondary [16] and resultant polytopes [8]. Our study can be extended to other determinantal predicates such as inCircle/inSphere and Volume.

Our contribution is twofold. First, we propose a method that improves the amortized complexity of the determinants involved in a convex hull computation (Sect. 2). This method uses the dynamic determinant evaluation procedure introduced in [18] to solve problems in graphs. Moreover, we study sequences of convex hull computations and propose a method that uses the computation done in a convex hull construction to improve the construction of subsequent convex hulls (Sect. 3). Second, we implement a simple method that optimizes the computation of subsequent determinantal predicates in both single and sequence of convex hull computations. The experiments show in the first scenario a speedup up to dimension 5 and in the second a speedup of 100 times when the dimension is 6 (Sect. 4).

Let us review previous work. There is a variety of algorithms and implementations for computing the determinant of a $d \times d$ matrix. Denoting their complexity by $O(d^\omega)$, the best known ω is 2.697263 [13]. However good asymptotic complexity does not imply good behavior in practice for small and medium dimensions. For instance, LinBox [6] implements algorithms with state-of-the-art asymptotic complexity, but introduces a significant overhead in medium dimensions, and seems most suitable in very high dimensions (typically > 100). There exists also a variety of algorithms for determinant sign computation [1, 4] with good asymptotic complexity. Eigen [11] seems to be suitable for medium to high dimensions, whereas CGAL [5] determinants proved to be efficient in low to medium dimensions. Decomposition methods have complexity $O(d^3)$, but they require the construction of intermediate objects, which adds a constant cost in computations and makes them slow in practice. There exist other simple methods such as [17] with complexity $O(d^4)$ and [2] with complexity $O(dM(d))$ where $M(d)$ is the complexity of matrix multiplication. Albeit simpler to implement, they also construct intermediate objects and thus suffer the same practical problem as decomposition methods. Laplace expansion, the simplest determinant method, does not need intermediate matrices and has complexity $O(d!)$ (note however that $d! < d^3$ for $d < 6$).

Concerning of sequences of determinants, the reference software for computing triangulations of a set of points, TOPCOM [16], computes all Orientation determinants that will be needed and stores their signs.

2 Convex Hull Computation

This section discusses the problem of dynamic computation of determinants produced in geometric computations such as convex hull and triangulation algorithms. These algorithms rely on the signs of determinantal Orientation predicates. The orientation of $d+1$ d-dimensional points is the sign of the determinant of a $(d+1) \times (d+1)$ matrix, where each column contains the d coordinates of each point, plus a 1 on

*National and Kapodistrian University of Athens, Department of Informatics and Telecommunications, Athens, Greece. {emiris,vissarion,lpenaranda}@di.uoa.gr. Partial support from project "Computational Geometric Learning", which acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open grant number: 255827.

the last place.

In the *dynamic determinant problem*, a $d \times d$ matrix A is given. Then, allowing some preprocessing, we should be able to handle updates of elements of A and queries for the current value of the determinant.

Lemma 1 [18, Thm.2] *The dynamic determinant problem can be solved in $O(d^\omega)$ time for preprocessing, $O(d^2)$ time for one column updates and $O(1)$ time for queries.*

We want to apply this result to incremental convex hull algorithms. Let $\mathcal{A} \subset \mathbb{R}^d$ be a pointset with $\mathrm{CH}(\mathcal{A})$ of dimension d, where $\mathrm{CH}(\cdot)$ denote the convex hull. In particular, we focus on the Beneath-and-Beyond (BB) algorithm [7, Sect. 8.4], where the construction of $\mathrm{CH}(\mathcal{A})$ is essentially the construction of a placing triangulation of $\mathrm{CH}(\mathcal{A})$ [15, Sect. 4.3]. Given $\mathcal{A}$, the algorithm at the first step computes a d-simplex and at every step it adds points by keeping a triangulated convex hull of the inserted points. At each step, a new point p is connected with all its visible facets. In order to determine if a facet f is visible from p we have to compute an Orientation predicate that involves p and the points of f. For each visible facet f a new full dimensional face (cell) σ is created, by connecting p to the points of f. Every cell σ has a vertex v that is not a vertex of f. At every step if we know the value of the Orientation determinant involves the vertices of cell σ we only need to compute the new value if we change v with p in this determinant. We can use Lem. 1 to compute this determinant in $O(d^2)$.

The idea is to store the value of Orientation in its corresponding cell together with an inverse $d \times d$ matrix that is used for updates [18, Sect. 4]. Denote t the number of cells of the resulting placing triangulation of $\mathrm{CH}(\mathcal{A})$ that stores the determinantal values and the inverse matrices. Note that every cell constructed by the algorithm corresponds to an Orientation predicate involving its points. The total number of predicates computed is thus t. The first predicate, corresponding to the initial d-simplex, is computed in $O(d^\omega)$. Then, the following holds.

Theorem 2 *The Orientation predicates of the BB algorithm can be computed in $O(d^2)$ amortized time and $O(d^2 t)$ space, where d is the dimension of the points and t is the number of cells of the resulting placing triangulation of the convex hull.*

This result improves the amortized computational complexity of the determinants involved in convex hull computation using the BB from $O(d^\omega)$ to $O(d^2)$.

The method presented in this section can be adapted to other algorithms which compute determinants of matrix updates. For instance, walk algorithms for point location in triangulations compute orientations involving vertices of neighboring cells.

3 Sequences of Convex Hull Computations

In this section we study the problem of computing a sequence of regular triangulations of a d-dimensional pointset $\mathcal{A}$ for given lifting vectors w. This can be done by computing the convex hull of the lifted $\mathcal{A}$ according to w and project its upper hull. This idea has already appeared in [9] where, given a system of $n+1$ polynomials in n variables, the proposed algorithm computes the Newton polytope (or a projection of it) of the resultant of this system. Given the input polynomials, it first defines a pointset $\mathcal{A}$ in dimension $2n$ (*i.e.*, $d = 2n$) and it needs to compute the regular triangulations of $\mathcal{A}$ given some lifting vectors. Similar problems often appear in combinatorial geometry, as in computations of secondary polytopes [16] or in the computation of volumes (*i.e.*, the Volume predicate) of the cells of a triangulation. Again motivated by [9], we want to compute the volume of the cells of regular triangulations of $\mathcal{A}$ given some lifting vectors.

The basic observation is that in every convex hull computation the input points differ only in their last coordinate, which comes from different lifting vectors. Thus, we expect many Orientation predicates that appear in this computation to be similar. Consider now the $d \times |\mathcal{A}|$ matrix with the points of $\mathcal{A}$ as columns. Define P as the extension of this matrix by adding a lifting vector $\vec{w}$ as the last row. We use the Laplace expansion along the last row for computing the determinant of the square submatrix formed by any $d+1$ columns of P; wlog these are the first $d+1$ columns $a_1, \ldots, a_{d+1}$. Let $\langle a_1, \ldots, a_{d+1} \rangle \setminus i$ be the vector $\langle a_1, \ldots, a_{d+1} \rangle$ without its i-th element; $P_{\langle a_1, \ldots, a_{d+1} \rangle \setminus i}$ is the $d \times d$ matrix obtained from the d first elements of the columns whose indices are in $\langle a_1, \ldots, a_{d+1} \rangle \setminus i$.

Orientation is the sign of the determinant of $P^{hom}_{\langle a_1, \ldots, a_{d+2} \rangle}$, constructed by columns $a_1, \ldots, a_{d+2}$ when we add $\vec{1} \in \mathbb{R}^{d+2}$ as a last row. Computing a regular triangulation is a long sequence of such predicates with different a_i's. We expand along the last two rows and compute the determinants $|P_{\langle a_1, \ldots, a_{d+2} \rangle \setminus \{i,j\}}|$ for $\binom{d+2}{2}$ combinations of $i, j \in \{1, \ldots, d+2\}$. The Volume predicate equals the determinant of $P^{hom}_{\langle a_1, \ldots, a_{d+1} \rangle}$, constructed by columns $a_1, \ldots, a_{d+1}$ when we replace its last row by $\vec{1} \in \mathbb{R}^{d+1}$.

Example 1 *Consider the following matrix, corresponding to a poinset $\mathcal{A}$, given the lifting vector $\vec{w} = \{w_1, \ldots, w_9\}$.*

$$P = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 0 & 1 & 2 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ w_1 & w_2 & w_3 & w_4 & w_5 & w_6 & w_7 & w_8 & w_9 \end{pmatrix}$$

Consider the computation of Orientation as a compu-

tation of $\binom{6}{4} = 15$ (4×4) *minors using the Laplace expansion. If we have computed* $|P^{hom}_{\langle 1,2,3,4,5,6\rangle}|$ *then the computation of* $|P^{hom}_{\langle 1,2,3,4,5,7\rangle}|$ *needs only* $\binom{6}{4}-\binom{5}{4} = 10$ *new minors. More interestingly, by giving a new lifting vector* $\vec{w'}$ *we can compute* $|P'^{\,hom}_{\langle 1,2,3,4,5,6\rangle}|$ *without computing any new minors.*

Our contribution consists in maintaining a data structure with the computed minors which are independent of $\vec{w}$. Once computed, they can be reused at subsequent steps of the algorithm. The main advantage of our scheme is that, for a new $\vec{w}$, the only change in P is its last row, hence computing the new determinants is done by reusing stored minors.

Lemma 3 *The complexity of Orientation and Volume predicates is* $O(d)$ *when all minors have been already computed.*

Proof. In Volume predicate, we expand along the row w, performing $O(d)$ arithmetic operations. For Orientation, we reduce the computation of the expansion of the last two rows to $O(d)$ by hashing the minors of the matrix with the row $\vec{1}$. □

4 Experiments on Hashing Determinants

We implemented in C++ the *hashing determinants* scheme, which consists in computing determinants using the Laplace expansion as in Sect. 3, storing all the computed minors in a hash table.

To avoid constructing a new matrix each time we compute a determinant, we keep a table T with all the points. The evaluation of the determinant using the Laplace expansion is performed recursively by using only the indices on T.

For the hash table implementation, we looked for a hashing function that takes as input a vector of integers (*i.e.*, the indices in T of the points that form the matrix) and returns a hash value that minimizes collisions. We considered many different hash functions, including some variations of the well-known FNV hash [10]. We obtained the best results with the implementation of Boost Hash [12], which has constant lookup cost in practice. For convex hull computation in general dimension we rely on CGAL's package `triangulation`, under development [3].

All the experiments that follow ran on an Intel Core i5-2400 3.1GHz, with 6MB L2 cache and 8GB RAM, under 64-bit Debian GNU/Linux. We performed experimental tests of our implementation in the scenario analysed in Sect. 3: we computed a sequence of convex hulls in the context of computing resultant polytopes [9]. Fig. 1 (left) shows a speedup of 100 times for 6 dimensional convex hulls. For 4-dimensional convex hulls, we gain a speedup of 18 times (we refer to [9] for graphs not shown here). An interesting point is that, with our technique, we computed in $<$ 2min an instance where $|\mathcal{A}| = 37$, that would take $>$ 1hr to compute otherwise. Thus, when the dimension and $|\mathcal{A}|$ becomes larger, this method allows us to compute instances that would be intractable otherwise.

The computation of one convex hull can also benefit from the hashing determinants scheme when using Laplace expansion. Even if a determinant is never computed twice, minors of size $< d$ can be reused. Based on this observation, we performed tests on the computation of one convex hull (Fig. 1, right). We generated random rational points uniformly distributed in the cube $[-100, 100]^d$, where the dimension d ranges between 2 and 6. Experiments show that our method performs better up to dimension 5, as expected. In dimension 6, hashing of minors does not help because Laplace expansion becomes very slow. We expect much better results for higher dimensions by implementing the algorithm of Sect. 2.

The drawback of our approach is the storage complexity, which is in $O(n^d)$. The hash table can be cleared at any moment to limit memory consumption, at the cost of dropping all previously computed minors. In our experiments, we obtained a good tradeoff between efficiency and memory consumption by clearing the hash table when it reaches 10^6 minors.

A more sophisticated approach, inspired from the paging problem in computer systems, consists in replacing a determinant in the data structure with the newly computed one when the data structure has reached a threshold size. A *replacement strategy* specifies the choice of which determinant to evict (for instance, the less used determinant or the less recently used determinant). However, the hash table data structure is not the most suitable for this. A good candidate to replace it are *tries* [14], which permit to index the stored values using a tree of depth d (the size of the matrices), thus yielding a $O(d)$ lookup and insertion time (since d is small, this is comparable to the cost of our hashing function). Tries are more suitable to implement replacement strategies. For instance, a *trie* permits us to store, on each node, the number of lookups on its successors. This information would permit to prune the less used subtrees at a given depth.

5 Future Work

We expect to quantify the gain of the BB algorithm when using dynamic determinants as suggested in Sect. 2 by performing a complexity analysis in which the dimension is not treated as a constant (to our knowledge, this analysis was not performed before).

For the computation of a sequence of convex hulls, we should consider implementing other determinant algorithms, to be more competitive dimensions > 5, and *tries*, to improve memory handling. For the computation of one single convex hull, we may implement

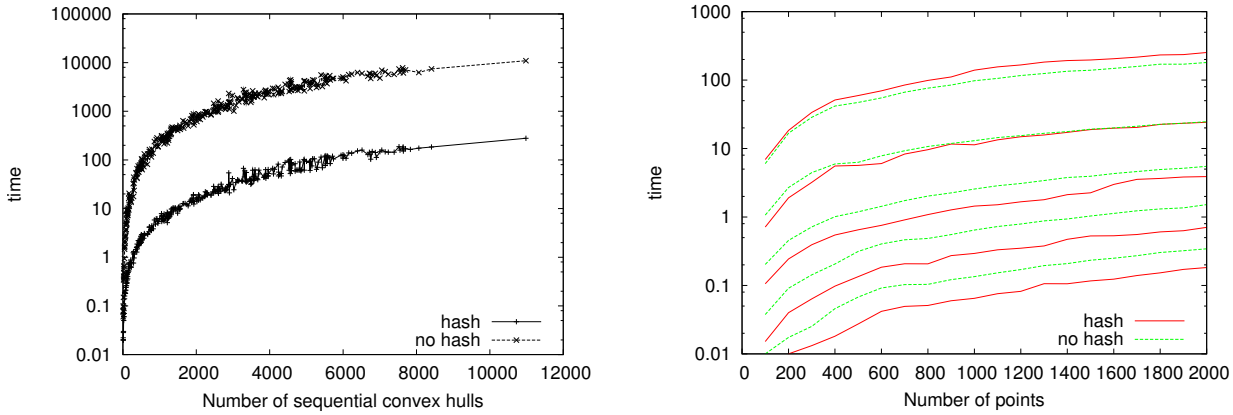


Figure 1: To the left, time (in seconds) for computing many 6-dimensional convex hulls of a set of points changing their lifting, hashing and non-hashing minors; the number of input points range from 10 to 45. To the right, time (in seconds) for computing one convex hull of a point set $\mathcal{A}$ as function of the number of input points. Graphs correspond to different ambient dimensions of the input points, *i.e.*, from bottom to top $\dim(\mathcal{A}) = 2, \ldots, 6$.

the algorithm presented in Sect. 2. Finally, it would be interesting to develop an interface that permits using the hashed determinants scheme transparently (*i.e.*, without knowledge of points' indices), to permit its use in any algorithm that uses determinants. This would also allow us to submit all our implementations to CGAL.

Finally, we should consider the extension of our approach to the 1-skeleton representation of a triangulation [3] which reduces the memory consumption with a penalty of slower running times.

Acknowledgments We thank O. Devillers and S. Hornus for discussions on `triangulation`, G. Rote for his suggestion on [2, 17], as well as the anonymous reviewers for their useful comments.

References

[1] J. Abbott, M. Bronstein, and T. Mulders. Fast deterministic computation of determinants of dense matrices. In *ISSAC*, pp. 197–203, 1999.

[2] R.S. Bird. A simple division-free algorithm for computing determinants. *Inf. Process. Lett.*, 111:1072–1074, Nov. 2011.

[3] J.-D. Boissonnat, O. Devillers, and S. Hornus. Incremental construction of the Delaunay triangulation and the Delaunay graph in medium dimension. In *SoCG*, pp. 208–216, 2009.

[4] H. Brönnimann, I.Z. Emiris, V. Pan, and S. Pion. Sign determination in Residue Number Systems. *Theor. Comp. Science, Spec. Issue on Real Numbers & Computers*, 210(1):173–197, 1999.

[5] CGAL: Computational geometry algorithms library. http://www.cgal.org.

[6] J.-G. Dumas, T. Gautier, M. Giesbrecht, P. Giorgi, B. Hovinen, E. Kaltofen, B.D. Saunders, W.J. Turner, and G. Villard. Linbox: A generic library for exact linear algebra. In *ICMS*, pp. 40–50, 2002.

[7] H. Edelsbrunner. *Algorithms in combinatorial geometry*. Springer-Verlag New York, Inc., New York, NY, USA, 1987.

[8] I.Z. Emiris, V. Fisikopoulos, and C. Konaxis. Exact and approximate algorithms for resultant polytopes. In *Proc. Europ. Workshop Computat. Geometry*, Assisi, Italy, 2012.

[9] I.Z. Emiris, V. Fisikopoulos, C. Konaxis, and L. Peñaranda. An output-sensitive algorithm for computing projections of resultant polytopes. arXiv:1108.5985v2 [cs.SC], 2011.

[10] G. Fowler, L.C. Noll, and P. Vo. FNV hash. www.isthe.com/chongo/tech/comp/fnv/, 1991.

[11] G. Guennebaud, B. Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[12] D. James. Boost functional library. www.boost.org/libs/functional/hash, 2008.

[13] E. Kaltofen and G. Villard. On the complexity of computing determinants. *Computational Complexity*, 13:91–130, 2005.

[14] D.E. Knuth. *The Art of Computer Programming, Volume III: Sorting and Searching*. Addison-Wesley, 1973.

[15] J.A. De Loera, J. Rambau, and F. Santos. *Triangulations: Structures for Algorithms and Applications*, volume 25 of *Algorithms and Computation in Mathematics*. Springer, 2010.

[16] J. Rambau. TOPCOM: Triangulations of point configurations and oriented matroids. In A.M. Cohen, X-S. Gao, and N. Takayama, eds., *Math. Software: ICMS*, pp. 330–340. World Scientific, 2002.

[17] G. Rote. Division-free algorithms for the determinant and the Pfaffian: algebraic and combinatorial approaches. In *Comp. Disc. Math.*, pp. 119–135, 2001.

[18] P. Sankowski and M. Mucha. Fast dynamic transitive closure with lookahead. *Algorithmica*, 56:180–197, 2010. 10.1007/s00453-008-9166-2.

Lines Through Segments in Three Dimensional Space*

Efi Fogel† Michael Hemmer† Asaf Porat† Dan Halperin†

Abstract

We present an efficient output-sensitive algorithm and its exact implementation to solve the following problem: Given a set $\mathcal{S}$ of n line segments in three-dimensional space, find all the lines that simultaneously intersect quadruples of line segments in $\mathcal{S}$. We refer to this problem as *the lines-through-segments problem*, or LTS for short. The algorithm properly handles all degenerate cases. Since we do not assume general position, we compute all lines that intersect *at least* four segments in $\mathcal{S}$. The algorithm runs in $O((n^3 + I)\log n)$ time, and requires $O(n + I)$ working space, where I is the output size; I is bounded by $O(n^4)$. We use CGAL arrangements and in particular its support for two-dimensional arrangements in the plane and on the sphere to efficiently compute the intersecting lines in an exact manner. We also report on the performance of our algorithm and its implementation compared to others. The source code of the LTS program as well as the input examples for the experiments can be obtained from `http://acg.cs.tau.ac.il/projects/lts`.

1 Introduction

LTS is a fundamental problem that arises in a variety of domains. For instance, solving the LTS problem can be used as the first step towards solving the more general problem of finding all lines tangent to four polyhedral objects taken from a set of polyhedral objects. The latter is ubiquitous in many fields of computation such as computer graphics (visibility computations), computational geometry (line transversal), robotics and automation (assembly planning), and computer vision.

*This work has been supported in part by the 7th Framework Programme for Research of the European Commission, under FET-Open grant number 255827 (CGL—Computational Geometry Learning), by the Israel Science Foundation (grant no. 1102/11), by the German-Israeli Foundation (grant no. 969/07), and by the Hermann Minkowski–Minerva Center for Geometry at Tel Aviv University.

†School of Computer Science, Tel-Aviv University, 69978, Israel. `efifogel@gmail.com`, `mhsaar@gmail.com`, `asafpor@gmail.com`, `danha@post.tau.ac.il`.

The number of lines that intersect four lines in $\mathbb{R}^3$ is 0, 1, 2, or infinite. Brönnimann et al. [3] showed that the number of lines that intersect four arbitrary line segments in $\mathbb{R}^3$ is 0, 1, 2, 3, 4, or infinite. In addition, they showed that the lines lie in at most four maximal *connected components*.[1]

A straightforward method to find all the lines that intersect four lines, given a set of n lines, examines each quadruplet of lines using the Plücker coordinate representation. Hohmeyer and Teller [7] and Redburn [6] exploited this method. It was later used by Everett et al. [4] as a building block for the problem of finding line transversals (the set of lines that intersect all given line segments). Naturally, the running time of this method is $O(n^4)$.

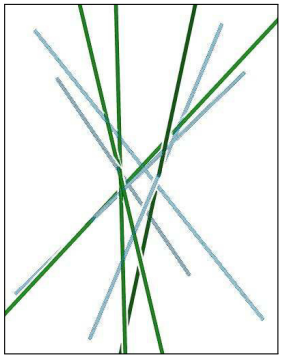


Figure 1: Four lines (drawn in green) that intersect four line segments (drawn in blue with a halftone pattern).

The combinatorial complexity of all the lines that intersect four line segments of a set of n line segments is $\Theta(n^4)$. The lower bound can be easily established by placing two grids of line segments in two parallel planes, respectively. However, in many cases the number of output lines is considerably smaller. The size of the output tends to be even smaller, when the input consists of line segments (as opposed to lines), which is typically the case in practical problems.

We present an efficient output-sensitive algorithm, and its complete and robust implementation that solves the LTS problem in three-dimensional Euclidean space. The implementation is complete in the sense that it handles all degenerate cases and guarantees exact results. Examples of degenerate cases are: A line segment may degenerate to a point, several segments may intersect, be coplanar, parallel, concurrent, lie on the same supporting line, or even overlap. To the best of our knowledge, this is the first algorithm (and implementation) for the LTS problem that is (i) output sensitive and (ii) handles all degenerate cases. The algorithm utilizes the idea of McKenna and O'Rouke [5] to represent the set of lines that intersect three lines as a rectangular hyperbola with vertical and horizontal asymptotes in $\mathbb{R}^2$. However, as opposed to their algorithm, which takes $O(n^4\alpha(n))$ time, our algorithm is output sensitive and its asymptotic time and space complexities are $O((n^3+I)\log n)$ and $O(n + I)$, respectively, where n is the input size

[1] Two lines tangent to the same four line segments are in the same connected component iff one of the lines can be continuously moved into the other while remaining tangent to the same four line-segments.

and I is the output size; I is bounded by $O(n^4)$.

Our algorithm is implemented on top of the Computational Geometry Algorithm Library (CGAL) [8]. The implementation is mainly based on the *2D Arrangements* package of the library [10]. This package supports the robust and efficient construction and maintenance of arrangements induced by curves embedded on certain orientable two-dimensional parametric surfaces in three-dimensional space [2, 9], and robust operations on them.[2] The implementation uses in particular 2D arrangements of rectangular hyperbolas with vertical and horizontal asymptotes in the plane and 2D arrangements of geodesic arcs on the sphere [1].

2 Representation

In this section we discuss the encoding of all lines that intersect two fixed line segments, S_1 and S_2, and a third line segment S_3. However, due to space limitation, we only discuss the generic case, where the lines underlying the input line segments are pairwise disjoint and their direction vectors are linearly independent. Then, we give one simple non-generic case.

We represent a line $L \subset \mathbb{R}^3$ by a point $p \in \mathbb{R}^3$ and a direction $d \in \mathbb{R}^3 \setminus \{\mathcal{O}\}$ as $L(t) = p + t \cdot d$, where $\mathcal{O}$ denotes the origin and $t \in \mathbb{R}$. Clearly, this representation is not unique. A segment $S \subset L \subset \mathbb{R}^3$ is represented by restricting t to the interval $[a, b] \subset \mathbb{R}$. We refer to $S(a)$ and $S(b)$ as the source and target points, respectively, and set $a = 0$ and $b = 1$. We denote the underlying line of a line segment S by $L(S)$. Two lines are *skew* if they are not coplanar. Three or more lines are *concurrent* if they all intersect at a common point.

Given two lines L_1 and L_2 we define a map $\Psi_{L_1L_2}$ as follows: $\Psi_{L_1L_2}(p) = \{(t_1, t_2) \in \mathbb{R}^2 \,|\, L_1(t_1),\ L_2(t_2),\ \text{and } p \text{ are collinear}\}$. That is, $\Psi_{L_1L_2}$ maps a point in $\mathbb{R}^3$ to a set in $\mathbb{R}^2$. This set, which might be empty, corresponds to all lines that contain p and intersect L_1 and L_2. Now consider the pair $(t_1, t_2) \in \mathbb{R}^2$. If $L_1(t_1) \neq L_2(t_2)$, then this pair uniquely defines the line that intersects L_1 and L_2 at $L_1(t_1)$ and $L_2(t_2)$, respectively. Thus, for skew lines L_1 and L_2 there is a canonical bijective map between $\mathbb{R}^2$ and all lines that intersect L_1 and L_2.

The characterization of $\Psi_{S_1S_2}(S_3) = \Psi_{L(S_1)L(S_2)}(S_3) \cap [0, 1]^2$ serves as the theoretical foundation of the algorithm that solves the LTS problem presented in Section 3. In the generic case, which we sketch in the next subsection, $\Psi_{S_1S_2}(S_3)$ consists of at most three arcs lying on a rectangular hyperbola with a vertical and a horizontal asymptote. In degenerate cases, which we omit here, the base curve may be a single line, or a pair of a horizontal and a vertical line. In some cases, e.g., S_1, S_2, and S_2 are coplanar, $\Psi_{S_1S_2}(S_3)$, is a two-dimensional region that is bounded by curves of the above kind.

[2] Arrangements on surfaces are supported as of CGAL version 3.4, albeit not documented yet.

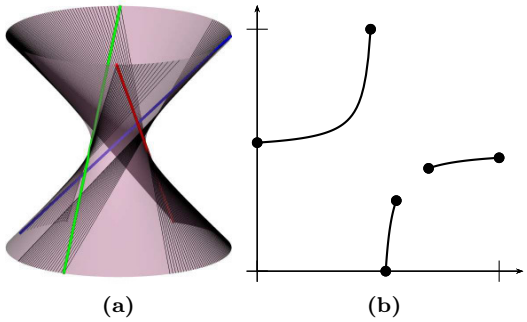


Figure 2: (a) Three surface patches the lines of which intersect three pairwise skew line segments, S_1, S_2, and S_3, in $\mathbb{R}^3$. These surface patches are contained in a hyperboloid of one sheet. (b) The point set $\Psi_{S_1S_2}(S_3)$.

If S_1 and S_2 intersect, it is impossible to encode the lines that go through the intersection point by $\Psi_{S_1S_2}(S_3)$. This case requires a special representation as discussed in Section 2.2.

2.1 Generic Case

We assume that the direction vectors of the underlying lines of the input line segments are linearly independent. Thus, we can apply a rational affine transformation, such that the three segments are given by $S_i(t_i) = p_i + t_i \cdot d_i,\ i \in \{1, 2, 3\}$, where $p_1 = (a, b, c)$, $p_2 = (d, e, f)$, $p_3 = \mathcal{O}$ and $d_i = e_i$ (where e_i denotes the unit vector along the ith axis).

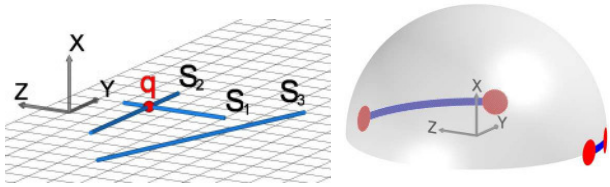
We further assume that the underlying lines are pairwise disjoint, which implies that $b \neq 0$, $d \neq 0$, and $c \neq f$. Consider the points $L_1(t_1)$, $L_2(t_2)$, and $L_3(t_3)$. These points are collinear iff $|(L_1(t_1) - L_2(t_2)) \times (L_3(t_3) - L_2(t_2))| = 0$. These are three dependent equations in three unknowns. Eliminating t_3 we obtain the following expression for t_2 in terms of t_1:

$$t_2(t_1) = \frac{e \cdot t_1 + (a \cdot e - d \cdot b)}{t_1 + a}. \qquad (1)$$

It implies that $\Psi_{L_1L_2}(L_3)$ is a rectangular hyperbola with a vertical asymptote and a horizontal asymptote. Nonetheless, we are interested in $\Psi_{S_1S_2}(S_3)$. Recall that all t_i, $1 \leq i \leq 3$, are restricted to $[0, 1]$. Similar to Equation 1, we can eliminate t_2 and solve for t_1 in terms of t_3. It is then easy to show that $\Psi_{S_1S_2}(S_3)$ consists of at most three maximal connected components, where each component represents a patch of a ruled surface as depicted in Figure 2.

2.2 S_1 and S_2 Intersect

Assume L_1 and L_2 intersect, and let $q = L_1(\tilde{t_1}) = L_2(\tilde{t_2})$ be the intersection point. The point $(\tilde{t_1}, \tilde{t_2})$ represents all lines that contain q. We represent



(a) (b)

Figure 3: (a) Three line segments, S_1, S_2, and S_3, such that S_1 and S_2 intersect at q (and S_3 does not). (b) The mapping $\Xi_q(S_3)$, where $\Xi_q(S_3) = \{\Xi_q(p) \,|\, p \in S_3\}$, which consists of two geodesic arcs on $\mathbb{H}^2$.

these lines by points on a semi open upper hemisphere centered at q. We define the additional map $\Xi_q : \mathbb{R}^3 \setminus \{q\} \to \mathbb{H}^2$ and $\Xi_q(p) \longmapsto d = s(p-q)/|p-q|$, with $s \in \{\pm 1\}$, such that $d \in \mathbb{H}^2 = \{p \,|\, p \in \mathbb{S}^2$ and p is lexicographically larger than $\mathcal{O}\}$. In the generic case a segment S maps to one or two geodesic arcs on $\mathbb{H}^2$. If S_3 is a point, or $L(S_3)$ contains q and S_3 does not, $\Xi_q(S)$ consists of a single point. If $q \in S_3$, we define $\Xi_q(S_3) = \mathbb{H}^2$; see Figure 3.

3 The Algorithm

We describe the algorithm that handles only a reduced set of cases due to space limitation. Henceforth, we assume that none of the input line segments degenerates to a single point and no three line segments are concurrent or coplanar. The complete algorithm, which is omitted here, handles all cases.

The input is a set $\mathcal{S} = \{S_1, \ldots, S_n\}$ of n line segments in $\mathbb{R}^3$. The output is a set of at most $O(n^4)$ lines in $\mathbb{R}^3$, such that each line intersects four line segments in $\mathcal{S}$. The intersected line segments are provided as part of the output.

The idea is to process each pair, (S_i, S_j) with $i < j-2$, and construct the planar arrangement, $\mathcal{A}_{ij}$, induced by the set $\{\Psi_{S_iS_j}(S_k) \,|\, i < k < j\}$. As we limit ourself to the generic case here, $\mathcal{A}_{ij}$ is induced only by hyperbolic arcs. Each intersection point of two such arcs represents a line that intersects S_i and S_j as well as the two segments that are mapped through $\Psi_{S_iS_j}$ to the hyperbolic arcs. Considering only pairs of segments S_i and S_j with $i < j-2$ and constructing the corresponding arrangement using only the line segments of the set $\{S_k \,|\, i < k < j\}$, ensures that every output line is reported exactly once.

If S_i and S_j intersect, we also construct an arrangement of geodesic arcs on the half-sphere $\mathbb{H}^2$ centered at their intersection point. The arrangement is induced by the central projections of the other segments on the sphere. Again, an intersection of two geodesic arcs encode lines that intersect four segments.

Using the sweep line algorithm, each arrangement, $\mathcal{A}_{ij}$, is constructed in $O(n \log n + I_{ij})$ time, where I_{ij} in the number of intersections in $\mathcal{A}_{ij}$. Since there are $O(n^2)$ arrangements, the total runtime is $O(n^3 \log n + I)$. As only one arrangement must be retained at a time, the required storage space is $O(n \log n + J)$, where J is the maximum number of intersections in a single arrangement. J is bounded by $O(n^2)$.

We remark that the complete algorithm, which is implemented, is necessarily more involved. In particular, in some degenerate cases $\Psi(S_i, S_j)(S_k)$ is a two dimensional patch that is bounded by hyperbolic arcs and line segements. In these cases we use the overlay algorithm of CGAL to add the two dimensional patches to the arrangement generated by the sweep.

4 Experimental Results

We have conducted several experiments on three types of data sets. The first comprises random input. The second produces the worst-case combinatorial output and has many degeneracies. The third consists of transformed versions of the former and has many near-degeneracies. We report on the time consumption of our implementation, and compare it to the implementation of J. Redburn [6]. All experiments were performed on a Pentium PC clocked at 2.40 GHz.

4.1 Random Input

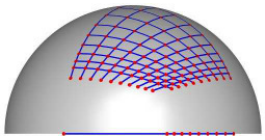
Input	Time		Lines
	LTS	Red	
Short	1.06	300.4	0
Medium	2.82	314.0	20,742
Long	5.15	327.0	64,151

The Random data set consists of 50 line segments drawn uniformly at random. In particular, the endpoints are selected uniformly at random within a sphere. We experimented with three different radii, namely, **Short**, **Medium**, and **Long** listed in increasing lengths. We verified that the line segments are in general position. The table above shows the number of output lines and the time in seconds it took to perform the computation using our implementation, referred to as **LTS**, and an instance of a program developed by J. Redburn [6] that relies on unlimited precision, referred to as **Red**. One can clearly observe how the time consumption of our implementation decreases with the decrease of the output size, which in turn decreases with the decrease in the line-segment lengths. Adversely, the time consumption of Redburn's implementation hardly changes.

4.2 Grid

The Grid data set, which attains the maximal number of output lines, comprises 40 line segments arranged in two grids of 20 lines segments each lying in two planes parallel to the yz plane. Each grid consists of ten vertical and ten horizontal line segments. Thus,

each plane contains 100 intersection points, which implies that the output includes exactly $100^2 = 10,000$ lines, each containing one intersection point of each plane. Due to the degenerate nature of the input, the output also includes several planar patches each lying in one of the two planes that contain the input grids, which we also compute but not elaborate on any further here. Using our implementation it took 20.74 seconds to compute the output. Unfortunately, Redburn's implementation could not handle this case.



We remark that every output line is represented by a vertex of an arrangement on the sphere. The figure to the right depicts one sample arrangement on the sphere centered at the intersection point of two line segments, S_1 and S_{40}, lying in the same plane. The arrangement is induced by the set of geodesic arcs $\{\Xi_{S_1 \cap S_{40}}(S_i) \,|\, i = 2, \ldots, 39\}$.

4.3 Transformed Grid

We conducted three additional experiments using a transformed version of the Grid data set. First, we slightly perturbed the input line segments, such that every two line segments became skew and the directions of every three line segments became linearly independent (input **I**). Secondly, we translated the (perturbed) horizontal line segments of one grid along the plane that contains this grid (input **II**), increasing the distance between the (perturbed) vertical and horizontal line segments of the grid. This drastically reduced the number of output lines. We then applied a similar translation to the other plane (input **III**), further reducing the size of the output. Table 1 shows the number of output lines and the time it took to perform the computation using our implementation. The output sensitivity of our algorithm is prominent. The table also shows the time it took to perform the computation using two instances of the program developed by Redburn. One instance, relies on a number type with unlimited precision, while the other resorts to double-precision floating-point numbers. As expected, when limited precision numbers were used, the output was only an approximation. Notice that the influence of the output size on the time consumption of Redburn's implementation is negligible.

Table 1: Perturbed Grid. Time is measured in seconds.

Input	Unlimited Prec.			Double Prec.	
	Time		Lines	Time	Lines
	LTS	Red		Red	
I	23.72	140.17	12,139	0.70	12,009
II	11.83	132.80	5,923	0.69	5,927
III	6.90	128.80	1,350	0.70	1,253

5 Acknowledgement

We thank Michael Hoffmann for helpful discussions on assembly partitioning, which inspired us to conduct the research discussed in this article. We also thank Linqiao Zhang who provided us with Redburn's code that was used for the experiments. Zhang used it as part of an implementation of an algorithm that constructs the visibility skeleton [11].

References

[1] E. Berberich, E. Fogel, D. Halperin, M. Kerber, and O. Setter. Arrangements on parametric surfaces II: Concretizations and applications. *Math. in Comput. Sci.*, 4:67–91, 2010.

[2] E. Berberich, E. Fogel, D. Halperin, K. Mehlhorn, and R. Wein. Arrangements on parametric surfaces I: General framework and infrastructure. *Math. in Comput. Sci.*, 4:45–66, 2010.

[3] H. Brönnimann, H. Everett, S. Lazard, F. Sottile, and S. Whitesides. Transversals to line segments in three-dimensional space. *Disc. Comput. Geom.*, 34:381–390, 2005. 10.1007/s00454-005-1183-1.

[4] H. Everett, S. Lazard, W. Lenhart, J. Redburn, and L. Zhang. On the degree of standard geometric predicates for line transversals. *Comput. Geom. Theory Appl.*, 42(5):484–494, 2009.

[5] M. McKenna and J. O'Rourke. Arrangements of lines in 3-space: a data structure with applications. In *Proc. 4th Annu. ACM Symp. Comput. Geom.*, pages 371–380, New York, NY, USA, 1988. ACM Press.

[6] J. Redburn. *Robust computation of the non-obstructed line segments tangent to four amongst n triangles.* PhD thesis, Williams College, Massachusetts, 2003.

[7] S. Teller and M. Hohmeyer. Determining the lines through four lines. *j. of graphics, gpu, and game tools*, 4(3):11–22, 1999.

[8] T. CGAL Project. *CGAL User and Reference Manual.* CGAL Editorial Board, 3.9 edition, 2011. `http://www.cgal.org/Manual/3.9/doc_html/cgal_manual/contents.html`.

[9] R. Wein, E. Fogel, B. Zukerman, and D. Halperin. Advanced programming techniques applied to CGAL's arrangement package. *Comput. Geom. Theory Appl.*, 38(1–2):37–63, 2007. Special issue on CGAL.

[10] R. Wein, E. Fogel, B. Zukerman, and D. Halperin. 2D arrangements. In *CGAL User and Reference Manual.* CGAL Editorial Board, 3.9 edition, 2011. `http://www.cgal.org/Manual/3.9/doc_html/cgal_manual/packages.html#Pkg:Arrangement2`.

[11] L. Zhang, H. Everett, S. Lazard, C. Weibel, and S. Whitesides. On the size of the 3D visibility skeleton: Experimental results. In *Proc. 16th Annu. Eur. Symp. Alg.*, volume 5193/2008 of *LNCS*, pages 805–816, Karlsruhe Allemagne, 2008. Springer.

Analysis of the Incircle predicate for the Euclidean Voronoi diagram of axes-aligned line segments*

Manos N. Kamarianakis† Menelaos I. Karavelas‡

Abstract

In this paper we study the most-demanding predicate for computing the Euclidean Voronoi diagram of axes-aligned line segments, namely the Incircle predicate. In particular, we show that the Incircle predicate can be answered by evaluating the signs of algebraic expressions of degree at most 6; this is half the algebraic degree we get when we evaluate the Incircle predicate using the current state-of-the-art approach.

1 Introduction

The Euclidean Voronoi diagrams of a set of line segments is one of the most well studied structures in computational geometry. There are numerous algorithms for its computation [5, 14, 16, 21, 7, 1, 15]. There are implementations that assume that numerical computations are performed exactly [19, 13], i.e., they follow the Exact Geometric Computation (EGC) paradigm [22], as well as algorithms that use floating-point arithmetic [10, 20, 9]; the latter class of algorithms does not guarantee exactness, but rather topological correctness.

Efficient and exact predicate evaluation in geometric algorithms is of vital importance. It has to be fast for the algorithm to be efficient. It has to be complete in the sense that it has to cover all degenerate cases, which, despite that fact that they are "degenerate" from the theoretical/analysis point-of-view, they are commonplace in real world input. In the EGC paradigm context, exactness is the bare minimum that is required in order to guarantee the correctness of the algorithm. The efficiency of predicates is typically measured in terms of the algebraic degree of the expressions (in the input parameters) that are computed during the predicate evaluation, as well as the number (and possibly type) of arithmetic operations involved. Degree-driven approaches for either the evaluation of predicates, or the design of the algorithm as a whole, has become an important question in algorithm/predicate design over the past few years [3, 17, 2, 4, 6, 18].

In this paper we are interested in the most demanding predicate of the Euclidean Voronoi diagram of axes-aligned line segments, namely the Incircle predicate. Axes-aligned segments are typical input instances in applications such as VLSI design [8]. Given three sites S_1, S_2, and S_3 we denote their Voronoi circle by $V(S_1, S_2, S_3)$ (if it exists). There are at most two Voronoi circles defined by the triplet (S_1, S_2, S_3); the notation $V(S_1, S_2, S_3)$ refers to the Voronoi circle that "discovers" the sites S_1, S_2 and S_3 in that (cyclic) order, when we walk on the circle's boundary in the counterclockwise sense. Given a fourth object O, which we call the *query object*, the Incircle predicate $\mathsf{Incircle}(S_1, S_2, S_3, O)$ determines the relative position O with respect to the disk D bounded by $V(S_1, S_2, S_3)$. The predicate is positive if O does not intersect D, zero if O touches the boundary but not the interior of D, and negative if the intersection of O with the interior of D is non-empty.

The Voronoi circle of three sites does not always exist. In this paper, however, we assume that the Incircle predicate is called during the execution of an incremental algorithm for computing the Euclidean Voronoi diagram of line segments, and thus the first three sites are always related to a Voronoi vertex in the diagram. Since we can circularly rotate the first three arguments of the Incircle predicate, there are only eight possible distinct configurations for the Incircle predicate: $PPPX$, $PPSX$, $PSSX$ and $SSSX$, where P stands for point, S stands for segment, and X stands for either P or S.

The predicates for the Euclidean Voronoi diagram of line segments, in the context of an incremental construction of the diagram, have already been studied by Burnikel [3]. Assuming that the input is either rational points, or segments described by their endpoints as rational points, Burnikel shows that the Incircle predicate can be evaluated using polynomial expressions of degree 40 in the input quantities (see the line dubbed "General [3]" in Table 1). Considering Burnikel's approach for the case of axes-aligned line segments, and performing the appropriate simplifications in his calculations, we arrive at a new set of algebraic degrees for the various configurations of the Incircle predicate (see line dubbed "Axes-aligned [3]" in Table 1); now the most demanding case the is $PPSX$ case, which gives algebraic degree 8 and 12, when the query object is a point and a segment, respectively.

*The full version of the paper may be found in [11].

†Department of Applied Mathematics, University of Crete, manos@tem.uoc.gr

‡Department of Applied Mathematics, University of Crete, mkaravel@tem.uoc.gr

This is an extended abstract of a presentation given at EuroCG 2012. 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 in a conference with formal proceedings and/or in a journal.

	$PPPP$	$PPSP$	$PSSP$	$SSSP$
General [3]	4	12	16	32
Axes-aligned [3]	4	8	4	2
Axes-aligned [this paper]	4	6	4	2

	$PPPS$	$PPSS$	$PSSS$	$SSSS$
General [3]	8	24	32	40
Axes-aligned [3]	6	12	4	2
Axes-aligned [this paper]	6	6	4	2

Table 1: Maximum algebraic degrees for the eight types of the Incircle predicate according to: [3] for the general and the axes-aligned segments case, and this paper. Top/Bottom table: the query object is a point/segment.

In Section 3 we analyze the $PPSX$ configurations for the Incircle predicate, and show how we can reduce the algebraic degrees for this case from 8 and 12, to 6. This is done by means of three key ingredients: (1) we reduce the $PPSP$ case to the $PPPS$ case, (2) we express the Incircle predicate as a difference of distances, instead of as a difference of squares of distances, and (3) we formulate the Incircle predicate as an algebraic problem of the following form: we compute a linear polynomial $L(x) = l_1x + l_0$ and a quadratic polynomial $Q(x) = q_2x^2 + q_1x + q_0$, such that the result of the Incircle predicate is the sign of $L(x)$ evaluated at a specific root of $Q(x)$.

2 Evaluation of the sign of $L(x) = l_1x + l_0$ at a specific root of $Q(x) = q_2x^2 + q_1x + q_0$

Let $L(x) = l_1x + l_0$ and $Q(x) = q_2x^2 + q_1x + q_0$ be a linear and a quadratic polynomial, respectively, such that $Q(x)$ has non-negative discriminant. Let the algebraic degrees of l_1, l_2, q_2, q_1 and q_0 be δ_l, $\delta_l + 1$, δ_q, $\delta_q + 1$, and $\delta_q + 2$, respectively. We are interested in the sign of $L(r)$, where r is one of the two roots $x_1 \leq x_2$ of $Q(x)$. Below we assume, without loss of generality, that $l_1, q_2 > 0$.

The obvious approach is to solve for r and substitute into the equation of $L(x)$. Let $\Delta_Q = q_1^2 - 4q_2q_0$ be the discriminant of $Q(x)$. Then $r = (-q_1 \pm \sqrt{\Delta_Q})/(2q_2)$, which, in turn, yields $L(r) = (l_1q_1 + 2l_0q_2 \pm \sqrt{\Delta_Q})/(2q_2)$. Computing $\text{sign}(L(r))$ is dominated by the computation of $\text{sign}(l_1q_1 + 2l_0q_2 \pm l_1\sqrt{\Delta_Q})$. This ammounts to evaluating expressions of algebraic degree at most $2(\delta_l + \delta_q + 1)$.

Observe now that evaluating the sign of $L(r)$ is equivalent to evaluating $\text{sign}(Q(x^\star))$, and possibly $\text{sign}(Q'(x^\star))$, where $x^\star = -\frac{l_0}{l_1}$ stands for the unique root of $L(x)$. Since $Q(x^\star) = (l_1^2q_0 - l_1q_1l_0 + q_2l_0^2)/l_1^2$ and $Q'(x^\star) = (l_1q_1 - 2q_2l_0)/l_1$, we conclude that, in order to evaluate $\text{sign}(L(r))$, we need to consider expressions of algebraic degree at most $2\delta_l+\delta_q+2$, which is smaller than the algebraic degree of the approach described early in this section, when $\delta_q > 0$.

3 The $PPSX$ case

Let A and B be the two points and CD be the segment defining the Voronoi circle. Without loss of generality, we may assume that CD is x-axis parallel, since otherwise we can reduce $\mathsf{Incircle}(A, B, CD, Q)$ to $\mathsf{Incircle}(\mathcal{R}(B), \mathcal{R}(A), \mathcal{R}(CD), \mathcal{R}(Q))$, where $\mathcal{R} : \mathbb{E}^2 \to \mathbb{E}^2$ denotes the reflection transformation about the line $y = x$. Notice that $\mathcal{R}$ preserves circles and line segments, reverses orientations, and is inclusion preserving. Finally, $\mathcal{R}$ maps an x-axis parallel segment to a y-axis parallel segment, and vice versa. Hence, $\mathsf{Incircle}(A, B, CD, QS) = \mathsf{Incircle}(\mathcal{R}(B), \mathcal{R}(A), \mathcal{R}(CD), \mathcal{R}(QS))$.

The query object is a point. Let Q be the query point. For the Voronoi circle $V(A, B, CD)$ to be defined, both A and B must be on the same side with respect to ℓ_{CD}. Consider now Q: if Q does not lie on the side of ℓ_{CD} that A and B lie, we have $\mathsf{Incircle}(A, B, CD, Q) > 0$. Testing the sideness of Q against ℓ_{CD} simply means testing the sign of $y_Q - y_C$, which is a quantity of algebraic degree 1.

Suppose now that Q lies on the same side of ℓ_{CD} as A and B, and let $\sigma = \mathsf{Orientation}(B, A, Q)$. In the special case $\sigma = 0$ (i.e., Q lies on the line ℓ_{BA}), we observe that Q lies inside the Voronoi circle $V(A, B, CD)$ if and only if Q lies on ℓ_{BA} and between A and B. This can be determined by evaluating the signs of the differences $x_A - x_B$, $x_Q - x_A$ and $x_Q - x_B$, if $x_A \neq x_B$, or the signs of the differences $y_A - y_B$, $y_Q - y_A$ and $y_Q - y_B$, if $x_A = x_B$, which are all quantities of algebraic degree 1.

If $\sigma \neq 0$, we are going to reduce $\mathsf{Incircle}(A, B, CD, Q)$ to $\mathsf{Incircle}(A, B, Q, CD)$ (see also Fig. 1). Suppose first that $\sigma < 0$, i.e., Q lies to the right of the oriented line ℓ_{BA}. Since A, B and CD appear on $V(A, B, CD)$ in that order when we traverse it in the counterclockwise sense, we conclude that Q lies inside $V(A, B, CD)$ (resp., lies on $V(A, B, CD)$) if and only if the circle defined by A, B and Q, does not intersect with (resp., touches) the segment CD. Hence, $\mathsf{Incircle}(A, B, CD, Q) = -\mathsf{Incircle}(A, B, Q, CD)$. In a similar manner, if $\sigma > 0$, i.e., Q lies to the left of the oriented line ℓ_{BA}, Q lies inside $V(A, B, CD)$ (resp., lies on $V(A, B, CD)$) if and only if the circle defined by B, A and Q intersects the line segment CD. Hence, $\mathsf{Incircle}(A, B, CD, Q) = \mathsf{Incircle}(B, A, Q, CD)$.

Since the Incircle predicate, in the $PPPS$ case, is of degree 6 (cf. Table 1), while $\mathsf{Orientation}(B, A, Q)$ is of degree 2, we deduce that $\mathsf{Incircle}(A, B, CD, Q)$ can also be answered using quantities of algebraic degree at most 6.

The query object is a segment. Let K be the center of $V(A, B, CD)$. K is an intersection point of

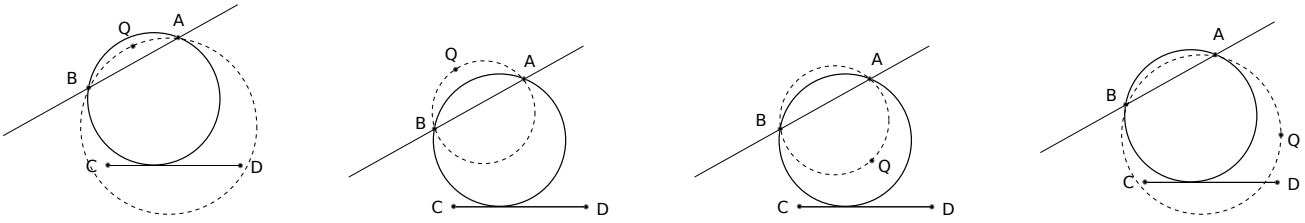


Figure 1: Reducing $\mathsf{Incircle}(A, B, CD, Q)$ to $\mathsf{Incircle}(A, B, Q, CD)$. Left/Right two: Q lies to the left/right of the oriented line ℓ_{BA}.

the bisector of A and B and the parabola with focal point A and directrix the supporting line ℓ_{CD} of CD. Solving the corresponding system of equations we deduce that, in the general case where A and B are not equidistant from ℓ_{CD} (i.e., if $y_A \neq y_B$), the x-coordinate of the Voronoi center x_K, is a root of a quadratic polynomial $P(x) = p_2x^2 + p_1x + p_0$, while the y-coordinate of the Voronoi center y_K, is a root of a quadratic polynomial $T(y) = t_2y^2 + t_1y + t_0$. Moreover, y_K and x_K are linearly dependent, i.e., $y_K = \frac{\alpha_1}{\beta}x_K + \frac{\alpha_0}{\beta}$. The algebraic degrees of p_2, p_1, p_0, t_2, t_1 and t_0 are 1, 2, 3, 2, 3 and 4, respectively. Furthermore, the degrees of α_1, α_0 and β are 1, 2 and 1, respectively. The roots $x_1 \leq x_2$ of the polynomial $P(x)$ (resp., $y_1 \leq y_2$ of $T(y)$) correspond to the centers of the two possible Voronoi circles $V(A, B, CD)$ and $V(B, A, CD)$. The roots of $P(x)$ and $T(y)$ of interest are shown in the following two tables.

Relative positions of A, B and CD	Root of $P(x)$ of interest
$y_C < y_A < y_B$ or $y_A < y_B < y_C$	x_1
$y_C < y_B < y_A$ or $y_B < y_A < y_C$	x_2

Relative positions of A, B	Root of $T(y)$ of interest
$x_A < x_B$	y_2
$x_A > x_B$	y_1

Let QS be the query segment. The first step is to compute $\mathsf{Incircle}(A, B, CD, Q)$ and, if needed, $\mathsf{Incircle}(A, B, CD, S)$. If at least one of Q and S lies inside $V(A, B, CD)$, we get $\mathsf{Incircle}(A, B, CD, QS) < 0$. Otherwise, we need to determine if the line ℓ_{QS} intersects $V(A, B, CD)$. If ℓ_{QS} does not intersect the Voronoi circle, we have $\mathsf{Incircle}(A, B, CD, QS) > 0$. If ℓ_{QS} intersects the Voronoi circle we have to check if Q and S lie on the same or opposite sides of the line $\ell^{\perp}_{QS}(K)$ that goes through the Voronoi center K and is perpendicular to ℓ_{QS}. Notice that since QS is axes-aligned, the line $\ell^{\perp}_{QS}(K)$ is either the line $x = x_K$ or the line $y = y_K$. Answering the $\mathsf{Incircle}$ predicate is equivalent to comparing the distance of K from the line ℓ_{QS} to the segment CD:

$$\mathsf{Incircle}(A, B, CD, \ell_{QS}) = d(K, \ell_{QS}) - d(K, CD). \quad (1)$$

Let us now examine and analyze the right-hand side difference of (1). Since the segment CD is x-axis parallel, $d(K, CD) = |y_K - y_C|$. Recall that y_K is a specific root of the quadratic polynomial $T(y)$. Therefore, determining the sign of $y_K - y_C$ reduces to evaluating the signs of $T(y_C)$ and $T'(y_C)$. Assume first that the segment QS is x-axis parallel. In this case, the equation of ℓ_{QS} is $y = y_Q$, and, hence, $d(K, \ell_{QS}) = |y_K - y_Q|$. As before, we can determine the sign of $y_K - y_Q$ by evaluating the signs of $T(y_Q)$ and $T'(y_Q)$. Hence, $\mathsf{Incircle}(A, B, CD, \ell_{QS}) = |y_K - y_Q| - |y_K - y_C| = J_1y_K + J_0$, where J_1 and J_0 are given in the following table.

$y_K - y_Q$	$y_K - y_C$	J_1	J_0
≥ 0	≥ 0	0	$y_C - y_Q$
	< 0	2	$-y_Q - y_C$
< 0	≥ 0	-2	$y_Q + y_C$
	< 0	0	$-y_C + y_Q$

Clearly, if $J_1 = 0$ we have $\mathsf{Incircle}(A, B, CD, \ell_{QS}) = \text{sign}(J_0)$. Otherwise, evaluating $\mathsf{Incircle}(A, B, CD, \ell_{QS})$ can be done as in Subsection 2. Since the algebraic degrees of J_1 and J_0 are 0 and 1, respectively, we can resolve the $\mathsf{Incircle}$ predicate using expressions of algebraic degree at most 4.

Consider now the case where QS is y-axis parallel. The equation of ℓ_{QS} is $x = x_Q$, and, thus, $d(K, \ell_{QS}) = |x_K - x_Q|$. As in the x-axis parallel case, x_K is a specific known root of the quadratic polynomial $P(x)$, i.e., determining the sign of $x_K - x_Q$ amounts to evaluating the signs of $P(x_Q)$ and $P'(x_Q)$. Using the fact that $y_K = \frac{\alpha_1}{\beta}x_K + \frac{\alpha_0}{\beta}$, we get $\mathsf{Incircle}(S_1, S_2, S_3, \ell_{QS}) = |x_K - x_Q| - |y_K - y_C| = \frac{1}{\beta}(L_1x_K + L_0)$, where L_1 and L_0 are given in the following table.

$x_K - x_Q$	$y_K - y_C$	L_1	L_0
≥ 0	≥ 0	$-\alpha_1 + \beta$	$\beta(y_C - x_Q) - \alpha_0$
	< 0	$\alpha_1 + \beta$	$\beta(-y_C - x_Q) + \alpha_0$
< 0	≥ 0	$-\alpha_1 - \beta$	$\beta(y_C + x_Q) - \alpha_0$
	< 0	$\alpha_1 - \beta$	$\beta(-y_C + x_Q) + \alpha_0$

If $L_1 = 0$, $\mathsf{Incircle}(S_1, S_2, S_3, \ell_{QS}) = \text{sign}(L_0)\text{sign}(\beta)$. Otherwise, given that x_K is a known root of $P(x)$, determining the sign of $L_1x_K + L_0$ can be done as in Subsection 2. Since the algebraic degrees of L_1 and L_0 are 1 and 2, respectively, evaluating the sign $L_1x_K + L_0$ reduces to computing the signs of expressions of algebraic degree at most 5.

As we mentioned at the beginning of this subsection, if $\mathsf{Incircle}(A, B, CD, \ell_{QS}) \leq 0$, we need to check the position of Q and S with respect to the either line

$x = x_K$ (if QS is x-axis parallel), or the line $y = y_K$ (if QS is y-axis parallel). To check the position of I, $I \in \{Q, S\}$, against the line $x = x_K$, we simply have to compute the signs of $P(x_I)$ and $P'(x_I)$. The algebraic degrees of these quantities are 3 and 2, respectively. In a symmetric manner, to check the position of I, $I \in \{Q, S\}$, against the line $y = y_K$, we simply have to compute the signs of $T(y_I)$ and $T'(y_I)$; their algebraic degrees are 4 and 3, respectively.

For the special case $y_A = y_B$, we easily get $x_K = \frac{1}{2}(x_A + x_B)$ and $y_K = \frac{U_2}{U_1}$, where the algebraic degrees of U_2 and U_1 are 2 and 1, respectively. If QS is x-axis parallel, we need to determine the sign of the quantity $d(K, \ell_{QS}) - d(K, CD) = |y_K - y_Q| - |y_K - y_C|$, or, equivalently, the signs of U_1 and $|U_2 - U_1 y_Q| - |U_2 - U_1 y_C|$, which are of algebraic degree 1 and 2, respectively. If QS is y-axis parallel, we need to evaluate the sign of $d(K, \ell_{QS}) - d(K, CD) = |x_K - x_Q| - |y_K - y_C|$, or, equivalently, the signs of U_1 and $|U_1(x_A + x_B - 2x_Q)| - 2|U_2 - U_1 y_Q|$, which are also of algebraic degree 1 and 2, respectively.

Recalling that, in order to evaluate Incircle(A, B, CD, QS), the first step is to evaluate Incircle(A, B, CD, Q), and, if needed, Incircle(A, B, CD, S), we conclude that in order to evaluate the Incircle predicate in the $PPSS$ case, we need to compute the sign of expressions of algebraic degree at most 6.

4 Future work

Our analysis is so far theoretical. We would like to implement the approach presented in this paper and compare it against the generic implementation in CGAL [12]. Finally, we would like to study and implement the rest of the predicates involved in the computation of the Voronoi diagram, when the line segment are axes-aligned.

Acknowledgments. The first author has been partially supported by a fellowship of the Onassis Foundation. The second author has been partially supported by the FP7-REGPOT-2009-1 project "Archimedes Center for Modeling, Analysis and Computation".

References

[1] J.-D. Boissonnat, O. Devillers, R. Schott, M. Teillaud, and M. Yvinec. Applications of random sampling to on-line algorithms in computational geometry. *Discrete Comput. Geom.*, 8:51–71, 1992.

[2] J.-D. Boissonnat and F. Preparata. Robust plane sweep for intersecting segments. *SIAM J. Comput.*, 29(5):1401–1421, 2000.

[3] C. Burnikel. *Exact Computation of Voronoi Diagrams and Line Segment Intersections.* Ph.D thesis, Universität des Saarlandes, Mar. 1996.

[4] O. Devillers, A. Fronville, B. Mourrain, and M. Teillaud. Algebraic methods and arithmetic filtering for exact predicates on circle arcs. *Comp. Geom: Theory & Appl., Spec. Issue*, 22:119–142, 2002.

[5] R. L. Drysdale, III and D. T. Lee. Generalized Voronoi diagrams in the plane. In *Proc. 16th Allerton Conf. Commun. Control Comput.*, pages 833–842, 1978.

[6] I. Z. Emiris and M. I. Karavelas. The predicates of the Apollonius diagram: algorithmic analysis and implementation. *Computational Geometry: Theory and Applications*, 33(1-2):18–57, January 2006. Special Issue on Robust Geometric Algorithms and their Implementations.

[7] S. J. Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica*, 2:153–174, 1987.

[8] P. Gupta and E. Papadopoulou. Yield analysis and optimization. In C. Alpert, D. Mehta, and S. Sapatnekar, editors, *The Handbook of Algorithms for VLSI Physical Design Automation*, chapter 7.3. Taylor & Francis CRC Press, November 2008.

[9] M. Held. VRONI: An engineering approach to the reliable and efficient computation of Voronoi diagrams of points and line segments. *Comput. Geom. Theory Appl.*, 18:95–123, 2001.

[10] T. Imai. A topology oriented algorithm for the Voronoi diagram of polygons. In *Proc. 8th Canad. Conf. Comput. Geom.*, pages 107–112. Carleton University Press, Ottawa, Canada, 1996.

[11] M. N. Kamarianakis and M. I. Karavelas. Analysis of the Incircle predicate for the Euclidean Voronoi diagram of axes-aligned line segments, July 2011. `arXiv:1107.5204v1 [cs.CG]`.

[12] M. Karavelas. 2D segment Delaunay graphs. In *CGAL User and Reference Manual*. CGAL Editorial Board, 3.9 edition, 2011.

[13] M. I. Karavelas. A robust and efficient implementation for the segment Voronoi diagram. In *Proceedings of the International Symposium on Voronoi Diagrams in Science and Engineering (VD2004)*, pages 51–62, Hongo, Tokyo, Japan, September 13–15, 2004.

[14] D. G. Kirkpatrick. Efficient computation of continuous skeletons. In *Proc. 20th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 18–27, 1979.

[15] R. Klein, K. Mehlhorn, and S. Meiser. Randomized incremental construction of abstract Voronoi diagrams. *Comput. Geom.: Theory & Appl.*, 3(3):157–184, 1993.

[16] D. T. Lee. Medial axis transformation of a planar shape. *IEEE Trans. Pattern Anal. Mach. Intell.*, PAMI-4(4):363–369, 1982.

[17] G. Liotta, F. Preparata, and R. Tamassia. Robust proximity queries: An illustration of degree-driven algorithm design. *SIAM J. Comput.*, 28(3):864–889, 1999.

[18] D. L. Millman and J. Snoeyink. Computing planar Voronoi diagrams in double precision: a further example of degree-driven algorithm design. In *Proceedings of the 26th Annual Symposium on Computational Geometry (SoCG'10)*, pages 386–392, Snowbird, Utah, USA, 2010.

[19] M. Seel. The AVD LEP user manual.

[20] K. Sugihara, M. Iri, H. Inagaki, and T. Imai. Topology-oriented implementation - an approach to robust geometric algorithms. *Algorithmica*, 27(1):5–20, 2000.

[21] C. K. Yap. An $O(n \log n)$ algorithm for the Voronoi diagram of a set of simple curve segments. *Discrete Comput. Geom.*, 2:365–393, 1987.

[22] C. K. Yap and T. Dubé. The exact computation paradigm. In D.-Z. Du and F. K. Hwang, editors, *Computing in Euclidean Geometry*, volume 4 of *Lecture Notes Series on Computing*, pages 452–492. World Scientific, Singapore, 2nd edition, 1995.

Maxmin Length Triangulation in Polygons

Christiane Schmidt*

Abstract

We consider the maxmin length triangulation problem in polygons. We give an NP-hardness proof for the case of polygons with holes and interior points. For simple polygons we present an optimal algorithm using dynamic programming.

1 Introduction

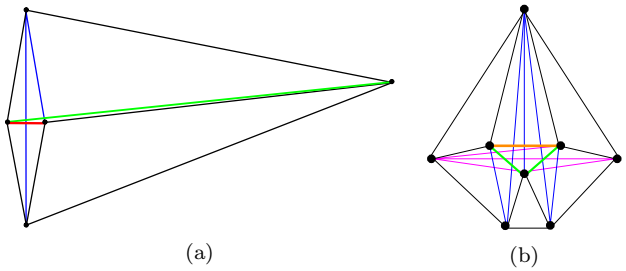
A *triangulation* T of a point set V in the Euclidean plane is a maximal set of noncrossing straight-line segments, where the line segments' endpoints are in V. The maximality ensures that all faces of the graph, except for the unbounded face, are triangles. The problem of computing a triangulation for a given point set has long been considered; for an overview we refer to the textbooks [1, 2].

If a subset S of all edges is given, Lloyd [8] showed that it is NP-complete to decide whether a triangulation $T \subseteq S$ exists.

Often we are interested in computing not just an arbitrary triangulation of the given point set, but to find an optimal triangulation for a certain optimality criterion. The Delaunay triangulation gives the optimal triangulation for a couple of criteria: maxmin angle, minmax smallest enclosing circle and minmax circumscribed circle. Mulzer and Rote [9] proved the *minimum weight triangulation* (MWT) to be NP-hard. The MWT problem asks for a triangulation that minimizes the sum of the Euclidean lengths of all edges. For polygonal domains, in which all edges on the polygon's boundary are enforced and only edges in the interior are allowed, Klincsek [6] presented a dynamic programming approach for the MWT with cubic runtime.

The *minmax length triangulation* is a triangulation of V, such that the longest edge is minimized. Edelsbrunner and Tan [3] presented an $O(n^2)$ algorithm for this problem.

Another length criterion is considered for the *maxmin length triangulation*: we look for a triangulation, such that the shortest edge is maximized. Hu [5] presented a linear-time algorithm for convex polygons (or points in convex positions as the enforced edges on the convex hull result in a convex polygon). The author also proved the graph version of this problem (with some positive edge weights) to be NP-complete. Unfortunately, not much is known for the triangulation of point sets in the plane. Intuitive solutions do not result in an optimal triangulation: the greedy approach would start with the longest edge; Figure 1(a) shows that this is not optimal. In addition, the use of edge flips, starting with some triangulation and aiming for increasing the shortest edge length, will not result in an optimal solution, as we may have to use edges that actually decrease the shortest edge length to obtain the optimal solution, see Figure 1(b).



(a) (b)

Figure 1: Neither a greedy approach nor edge flipping can lead to an optimal solution. (a) Greedy starts with the longest green edge and is not optimal (short red edge is enforced). (b) Edge flips to the optimal (blue) triangulation include the insertion of the shortest, green edges at some point. The starting triangulations includes the orange edge and the pink edges. Black edges occur in both triangulations.

In this paper we prove the maxmin length triangulation for polygons with holes and interior points to be NP-hard. Hence, in contrast to a given point set in the Euclidean plane, for which only the edges on the convex hull are enforced, and arbitrarily many points are located in its interior, we are given a polygon P as well as points in the polygon's interior. In addition, we consider the restricted case of simple polygons, without points in the interior.

This leaves open the cases of polygons with holes without interior points and, more importantly, the case of simple, or even convex polygons with interior points, as this equals the point set case.

2 Notations

We are given a polygon P. The vertices of P are denoted by $v_1, \ldots, v_n$. Let the vertices be ordered clockwise. A triangulation T of P is defined to be a decomposition of P into triangles, using a maximal

*Braunschweig University of Technology, Germany, c.schmidt@tu-bs.de

This is an extended abstract of a presentation given at EuroCG 2012. 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 in a conference with formal proceedings and/or in a journal.

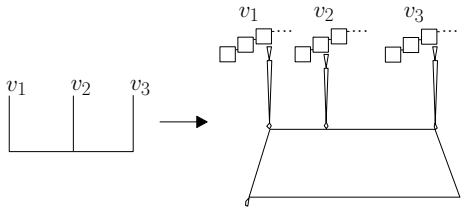


Figure 2: Overview of the construction.

set of noncrossing diagonals, see, for example, [2].

In Section 3 we consider polygons with additional interior points in the polygon's interior.

For a triangulation T we define $w_{min}(T)$ to be the length of a shortest edge in T.

Definition 1 *The maxmin length triangulation problem in a polygon P asks for a triangulation T^* of P, such that for all triangulations $T \neq T^*$: $w_{min}(T^*) \geq w_{min}(T)$.*

For two points p_1, p_2 let $d(p_1, p_2)$ denote the Euclidean distance of the points.

3 Polygons with Holes and Interior Points

Theorem 1 *Maxmin length triangulation is NP-hard for polygons with holes and interior points.*

Proof. The proof is based on a reduction of the NP-hard problem `Planar 3SAT`, a special case of 3SAT in which the variable-clause incidence graph H is planar. In particular, there is a rectilinear embedding of H for which variables $v_1, \ldots, v_n$ lie on a straight line and the clauses are arranged above and below them; see Knuth and Raghunathan [7].

We turn this embedding into a polygon: We represent the variables, clauses and edges by polygonal pieces. Moreover, we introduce interior points. A sketch of this transformation can be seen in Figure 2. We will give the details of the gadgets in the following.

The basic idea for the reduction is the use of very short edges (indicated in green in the following), that need to be crossed by other triangulation edges, as their occurrence in a triangulation would reduce $w_{min}(T)$. In particular, those very short edges are shorter than all polygon edges. In addition, interior points (depicted as violet stars)—sometimes also with very short edges in between—are inserted, which makes it impossible to switch from one triangulation to the other. The basic idea of cutting short edges (diagonals) is based on the following lemma:

Lemma 2 *Let S be a point set, d a diagonal and T a triangulation of S with edges t_i $i \in I$: $d \notin T \Leftrightarrow \exists i \in I : d \cap t_i \neq \emptyset$.*

In the following, black edges are polygon edges. We give a grid for better orientation; the real grid is the integer grid induced if any edge of the shown grid has side length 5. For clarity, we will not depict all edges of a triangulation, but the edges necessary for our argumentation, only.

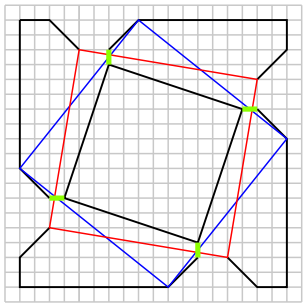


Figure 3: Basic idea for the variable gadget.

The general concept for a variable gadget is given in Figure 3: all 4 green short edges need to be cut (Lemma 2). The possible ways to cut the green edges intersect in turn. Thus, we either use blue or red edges (red edges may end at a vertex incident to the green edges, the intersection property is kept). As we have to be able to branch off polygonal corridors and we need to stick together various of these square-shaped constructions, in order to branch off sufficiently many corridors the slightly varied construction of Figure 4 results. The square-shaped variable patterns are connected to the sides, corridors can branch off to the top and bottom (as we will see later). Again, we need either the red or blue edges to cross all green edges from the left square. Then, the interior points (violet stars) with short distance are introduced to ensure that a blue triangulation in the left square remains a blue triangulation in the right square: Only the possible blue edge to the right square and the short red edge intersect the violet edge. As the short red

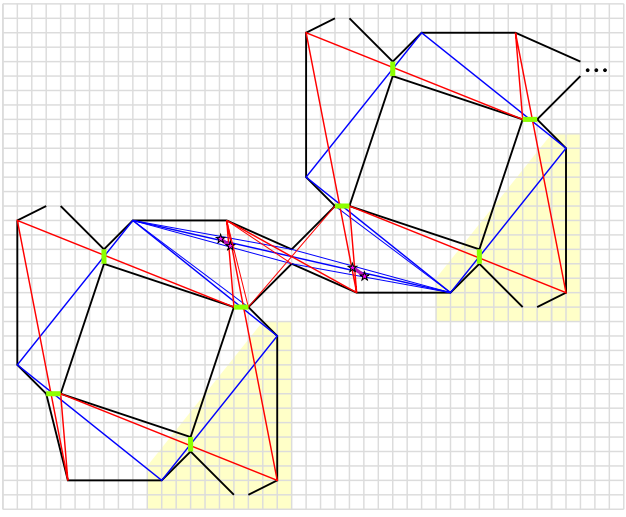
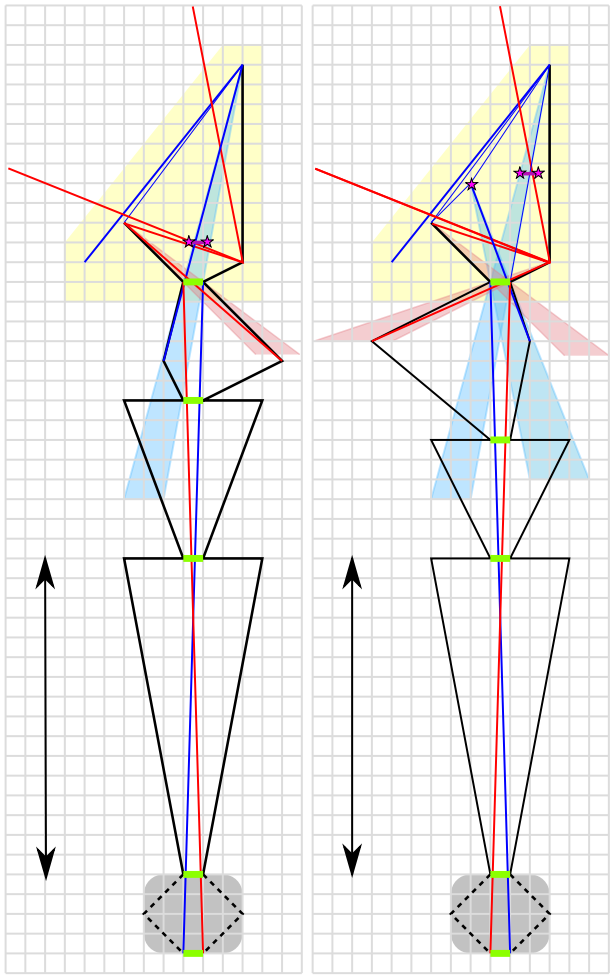


Figure 4: Example for the variable gadget. The yellow highlighted area indicates the area from Figure 5.



(a) (b)

Figure 5: Examples for corridor construction. The gray shaded area indicates the area from Figure 6.

edge intersects with the given blue edge around the square's hole, the blue edge is enforced.

The corridor connections are shown in Figure 5. There are two possible ways to connect a corridor: the blue and red triangulation represent two possible truth settings. One triangulation corresponds to a truth setting that satisfies the clause (i.e., a setting of "true" if v_i is in the clause, a setting of "false" if $\neg v_i$ appears in the clause), and we need to be able to give both the same role in the clause. In case the blue triangulation corresponds to a truth setting that satisfies the clause, we use the construction of Figure 5(a); If the red triangulation corresponds to a truth setting that satisfies the clause, we use the construction of Figure 5(b). In both cases the arrows indicate that this polygonal part can have arbitrary height, which allows us to connect to the clause gadget on the "right" level. The dashed polygonal parts indicate that this takes the form of one of the three possible pieces shown in Figure 6, depending on the position in the clause gadget (left, center, right).

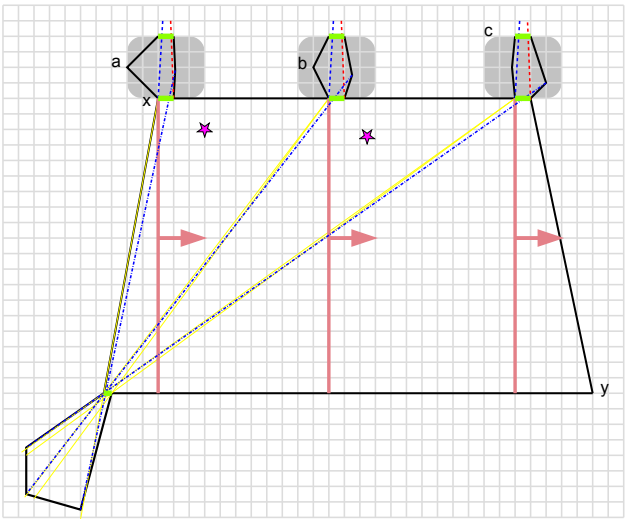


Figure 6: Example for the clause gadget.

Again, we use very short green edges that need to be cut. Only the topmost green edge can be cut by crossing edges with an incident vertex in the variable gadget. The spearheaded triangular shape of the corridors enforces all other green edges (except for the bottommost green edge) to be cut by an edge that is incident to one vertex incident to the topmost green edge and one vertex incident to the bottommost green edge. No straight lines inside of the polygon from a vertex in the spearheaded triangle structure end in a vertex in the clause gadget. Hence, the bottommost green edges need to be treated in the clause gadget. All other green edges cannot be cut in another way.

In the following, we depict the range of possible edge connections by light blue and red cones.

In Figure 5(a) the additional interior points (violet stars) enforce again that one triangulation cannot switch to the other. Note that the two points cannot be used to cut the green edges due to the other triangulation edges for both settings. In Figure 5(b), the pair of interior points serves the same purpose as in (a). The single interior point is added to enforce the mirrored connections inside of the corridor.

An example for a clause gadget is shown in Figure 6. Here we use blue to denote the triangulation that corresponds to a truth setting satisfying the clause (and red for the other setting). The clause gadget features another even shorter (green) edge (bottom-left). In case one variable satisfies the clause we may use the corresponding blue dashed-dotted edge to cut this edge. For a variable not satisfying the clause, edges from a, b and c to either y or the two interior (violet) points are necessary. Edges from an interior point or the lower left corner of the corridor cannot cross the short bottom-left edge. Thus, in case at least one variable has a truth setting satisfying the clause, we are able to cut all short edges, in partic-

ular the lower bottom-left edge in the clause gadget. If all variables have a truth setting not satisfying the clause we cannot cut all green edges in the polygon plus the lower bottom-left edge in the clause gadget. □

4 Simple Polygons

We use straightforward dynamic programming for computing the maxmin length triangulation in a simple polygon: in each step we establish the maxmin triangulation of the polygon induced by vertices being k apart $(v_i, v_{i+1}, \ldots, v_{i+k})$ including the edge $\{v_i, v_{i+k}\}$. Possible triangulations include the edge $\{v_i, v_{i+k}\}$ and the maxmin triangulations from polygons induced by the vertices v_i, v_{i+m} and v_{i+m}, v_{i+k}, where m runs from 1 to $k-1$. Hence, the value of each of these triangulations is given by the minimum of $d(v_i, v_{i+k})$ and the longest shortest edge of triangulations of the polygons induced by the vertices $\{v_i, v_{i+m}\}$ and $\{v_{i+m}, v_{i+k}\}$. We then determine the maximum of all these values over all possible values of m.

A minor modification of this procedure is necessary: in case we simply consider the distance between two vertices we do not account for the possibility that this diagonal may not run through P's interior. Consequently, we need to adapt our distance function. We define

$$d^I(v_i, v_j) = \begin{cases} d(v_i, v_j) & : \quad j = i+1, \text{ or } \{v_i, v_j\} \in P \\ -\infty & : \quad otherwise \end{cases}$$

This rules out any triangulation that includes a non-valid edge, as we will in the end determine the maximum over all possible values. Note that the triangulations of a subpolygon may actually obtain the value $-\infty$.

Algorithm 1 summarizes this approach: Let $E(i, j)$ $(i < j)$ be longest edge of all shortest edges of possible triangulations of the subpolygon involving the nodes $v_i, v_{i+1}, \ldots, v_j$.

Lemma 3 *Algorithm 1 is correct and solves the maxmin length triangulation problem in simple polygons in time $O(n^3)$.*

Note that this also allows for a polynomial time algorithm $(O(n^h))$ for a constant number of holes, h .

5 Conclusion

We proved the maxmin length triangulation to be NP-hard for polygons with holes and interior points. We are confident that variations of the gadgets given in Section 3 show that this problem cannot be approximated within a constant factor. On the other hand there is a simple dynamic programming approach for simple polygons. As it turns out, the intermediate and most important case of point sets is still NP-complete [4].

Algorithm 1: Maxmin length triangulation for Simple Polygons

1.:
For $k = 1, i = 1, 2, \ldots, n$ and $j = i + k$ let $E(i, j) = d^I(v_i, v_j)$.

2.:
$k = k + 1$. For $i = 1, 2, \ldots, n$ and $j = i + k$ let

$$E(i,j) = \max_{i<m<j} \left[\min \left\{ d^I(v_i, v_j), E(i, m), E(m, j) \right\}\right] \quad (1)$$

The index where the maximum in Equation(1) is achieved is denoted by $q(i, j) \quad \forall (i, j)$.

3.:
If $k < n - 1$ go to 2. Otherwise $w_{min}(T^*) = E(1, n)$.

4.:
In order to find the edges in the maxmin length triangulation T^*, we backtrace: $\{v_1, v_n\} \in T^*$. $\forall \{v_i, v_j\} \in T^*,\ j < i + 1 :\ q = q(i, j)$. Then: $\{v_i, v_q\} \in T^*$ and $\{v_q, v_j\} \in T^*$.

Acknowledgments

I wish to thank Joseph S.B. Mitchell for starting discussions on the maxmin length triangulation and Alexander Kröller for helpful discussions.

References

[1] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 2008.

[2] S. L. Devadoss and J. O'Rourke. *Discrete and Computational Geometry*. Princeton University Press, 2011.

[3] H. Edelsbrunner and T.-S. Tan. A quadratic time algorithm for the minmax length triangulation. *SIAM J. Comput.*, 22:527–551, June 1993.

[4] S. P. Fekete. Personal communication, 2012.

[5] S. Hu. A linear time algorithm for max-min length triangulation of a convex polygon. *Inf. Process. Lett.*, 101:203–208, March 2007.

[6] G. Klincsek. Minimal triangulations of polygonal domains. In P. L. Hammer, editor, *Combinatorics 79*, volume 9 of *Annals of Discrete Mathematics*, pages 121 – 123. Elsevier, 1980.

[7] D. E. Knuth and A. Raghunathan. The problem of compatible representatives. *SIAM Journal of Discrete Math.*, 5(3):422–427, 1992.

[8] E. L. Lloyd. On triangulations of a set of points in the plane. In *Foundations of Computer Science, 1977., 18th Annual Symposium on*, pages 228 –240, 31 1977-nov. 2 1977.

[9] W. Mulzer and G. Rote. Minimum weight triangulation is NP-hard. In *Proceedings of the twenty-second annual symposium on Computational geometry*, SCG '06, pages 1–10, New York, NY, USA, 2006. ACM.

On Triangulation Axes of Polygons*

Wolfgang Aigner† Franz Aurenhammer† Bert Jüttler‡

Abstract

We define the *triangulation axis* of a simple polygon P as an anisotropic medial axis of P whose 'unit disks' are line segments or triangles. The underlying triangulation that specifies the anisotropy can be varied, to adapt the axis so as to reflect predominant geometrical and topological features of P. Triangulation axes are piecewise linear skeletons, and typically have much fewer edges and branchings than the Euclidean medial axis or the straight skeleton of P (between $n-2$ and $2n-6$ edges, compared to $2n-3$). Still, they retain important properties, as for example the reconstructability of P from its skeleton. Triangulation axes can be easily computed from their defining triangulations in $O(n)$ time. We investigate the effect of using several optimal triangulations for P. In particular, careful edge flipping in the constrained Delaunay triangulation leads, in $O(n \log n)$ overall time, to an axis competitive to high quality axes requiring $O(n^3)$ time for optimization via dynamic programming.

1 Introduction

Let P be a simple polygon in the plane. A disk $D \subset P$ is called *maximal* (for P) if there is no other disk $D' \subset P$ with $D' \supset D$. The (Euclidean) *medial axis* of P is the union of centers of all maximal disks for P. This tree-like skeletal structure has proved a very useful descriptor of shape. Applications in diverse areas exist, and various construction algorithms have been proposed; see e.g., [2, 4] and references therein.

The medial axis is uniquely defined by the given input polygon P. The same is true for the straight skeleton [3] of P, which can serve as a piecewise-linear alternative to the medial axis. In certain applications, however, it is desirable to have some flexibility in designing a skeletal structure, be it for keeping its size small so as to reflect only the essential parts of P, or for the sake of stability with respect to slight boundary changes of P. Several attempts have been made to adapt and prune the medial axis accordingly; Attali et al. [4] is a good survey.

In the present note, we propose a different idea, namely, of putting some anisotropy on the polygon P.

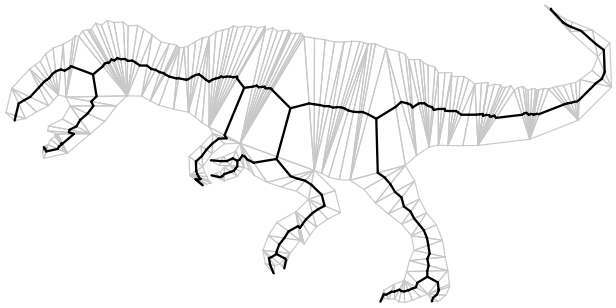


Figure 1: A triangulation axis of a polygonal shape

Distances are measured differently at different locations within P, by varying the shape of the inscribed disks. (Anisotropic Voronoi diagrams where distances are measured individually from each defining point site have been introduced in Labelle and Shewchuk [8].) We divide the polygon P into triangles, and allott to each triangle a continuous family of unit disks, resulting from appropriately defined convex distance functions. (Voronoi diagrams for convex distance functions have been considered first in Chew and Drysdale [6].) The resulting skeleton is a tree similar to the dual of the chosen triangulation of P, which is piecewise linear, and always consists of fewer edges than the medial axis or the straight skeleton. When using the various known types of triangulation (e.g., constrained Delaunay, minimum weight), and also other triangulations optimal in different respects, we gain the needed flexibility, with the ultimate goal of defining a simple, stable, and characteristic skeletal axis structure for P.

2 Triangulation axis

A *triangulation*, T, of a simple polygon P is a partition of P into triangles whose vertices are all from P. Let P have $n \geq 4$ vertices. To define what we call the *triangulation axis*, $M_T(P)$, *of* P *and* T, the triangles which constitute T are categorized into three types: *ear* triangles, *link* triangles, and *branch* triangles—having one, two, or three sides that are diagonals of P, respectively. Depending on its type, a triangle Δ contributes a specific part to $M_T(P)$. If Δ is an ear triangle, then its axis part is the line segment that connects the midpoint of its unique bounding diagonal d of P to the vertex of Δ opposite to d. If Δ is a link tri-

*Supported by ESF Programme EuroGIGA-Voronoi

†Institute for Theoretical Computer Science, Graz University of Technology, Austria, {waigner,auren}@igi.tugraz.at

‡Institute of Applied Geometry, Johannes Kepler University Linz, Austria, Bert.Juettler@jku.at

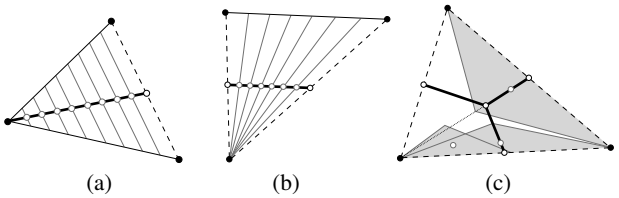


Figure 2: Triangle types: (a) ear triangle, (b) link triangle, and (c) branch triangle

angle, then it contributes to $M_T(P)$ the line segment connecting the midpoints of the two bounding diagonals of P. Finally, if Δ is a branch triangle, then the three line segments connecting its side midpoints to a distinguished point (for example, the centroid of Δ) is taken. See Figure 2, where axis parts are drawn in bold lines. A particular triangulation axis of a polygon is depicted in Figure 1. Observe that link triangles (which typically constitute the majority in T) give rise to homothetic copies of P's boundary parts in the axis.

$M_T(P)$ can be interpreted as an anisotropic medial axis of P. Using suitable convex unit disks, the centers of maximal inscribed disks will delineate the triangulation axis. This is made explicit in Figure 2. Unit disks for ear triangles and link triangles are just line segments of varying slopes, whereas unit disks for branch triangles are of triangular shape.

We found out recent mention of a triangulation axis in Wang [9] for GIS applications, who refer to Ai and van Oosterom [1] for earlier use. However, no systematic study of $M_T(P)$ is provided, and only the constrained Delaunay triangulation [5] of P is used for T.

A nice feature of triangulation axes is their small combinatorial size.

Lemma 1 *Any triangulation axis of a simple polygon P with n vertices has between $n-2$ and $2n-6$ edges.*

Proof. Each triangulation T of P has the same number of triangles, $n-2$. Ear triangles and link triangles yield 1 edge of $M_T(P)$ each, whereas branch triangles contribute 3 edges. The number of edges of $M_T(P)$ thus is $n-2+2b$, with b counting the branch triangles of T. Moreover, we have $0 \le b \le \frac{n}{2} - 2$, the upper bound stemming from the fact that T has $b+2$ ear triangles, so that $(b+2)+b \le n-2$. The claimed bounds follow. □

In comparison, the medial axis of P has $2n-3$ edges, r of which are parabolically curved (one for each reflex vertex of P), and the straight skeleton of P has $2n-3$ straight edges.

A desired property of skeletal axes is the ability of restoring the polygon P from its axis. Both the medial axis and the straight skeleton share this property. However, distances from the axis edges to the boundary of P have to be stored, in addition. (This is commonly called the *medial axis transform* for the former structure.) Triangulation axes are even more well-behaved in this respect: Storing an additional bit for certain edges is sufficient to guarantee a unique reconstruction of P.

Lemma 2 *Given any of its triangulation axes, a simple polygon P can be reconstructed in $O(n)$ time.*

Proof. We reconstruct P triangle by triangle. Note first that triangle types can be recognized from the axis part they yield. Actually, any branch triangle Δ is reconstructable uniquely, as Δ is just a rotated copy of its edge midpoints' triangle, in doubled size; cf. Figure 2(c). If the next triangle is an ear triangle, its reconstruction is trivial, too. In the case of a link triangle, we have two choices of following P's boundary in a way parallel to the respective axis edge; see Figure 3. Storing a 'left/right' bit is necessary in that case. Finally, if the axis does not branch at all (is just a path) then the coordinates of a single (non-ear) vertex of P have to be remembered. □

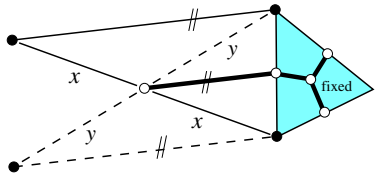


Figure 3: Two possible link triangles to continue

At most $n-4$ bits have to be stored, as any triangulation of P has at least 2 ear triangles. In contrast, the medial axis of P has $n+r-2$ inner vertices, for each we need to store the radius of the maximal disk centered there.

Without extra information, the polygon reconstruction is ambiguous, even when the triangulation is known to be constrained Delaunay. That is, the empty-circle property does not help; Figure 4 gives an example.

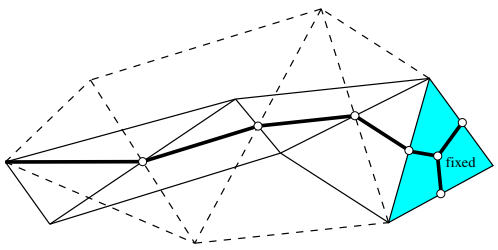


Figure 4: P cannot be restored from its Delaunay axis

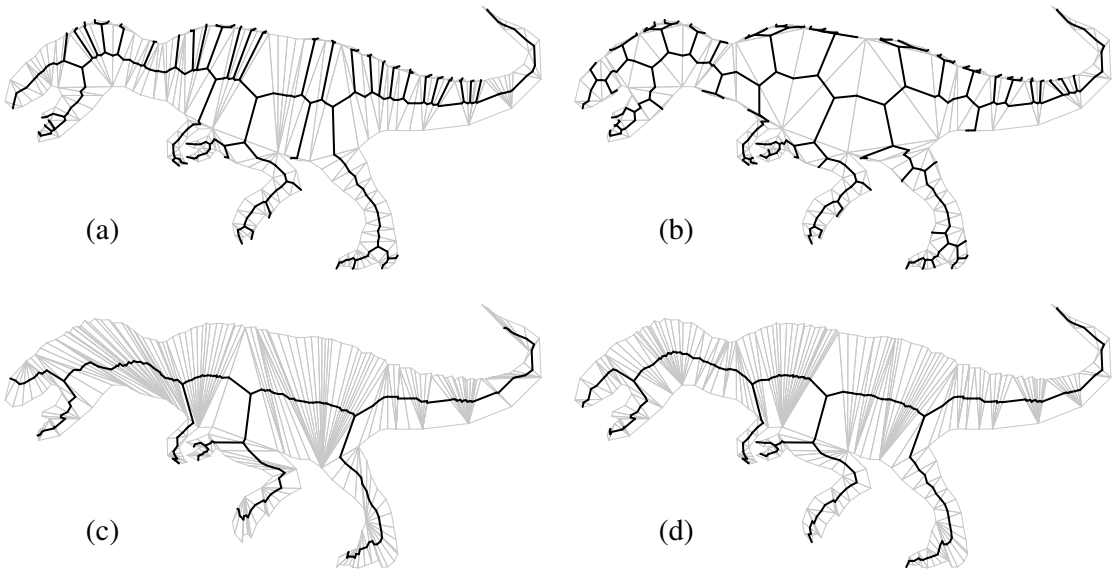


Figure 5: Comparison of optimal triangulation axes: (a) constrained Delaunay, (b) minimal weight, (c) minimal axis length, (d) minimal weight with expensive branch triangles

The next lemma indicates that triangulation axes tend to describe the topology of a polygon more compactly than does the medial axis.

Lemma 3 *For no triangulation T, $M_T(P)$ has more branchings than the medial axis of P.*

Proof. Each convex vertex v of P is a leaf of the medial axis, whereas v is a leaf of $M_T(P)$ only if T contains an ear triangle at v. Reflex vertices of P cannot be leaves in either structure, and the number of branchings in any tree is exactly 2 less than the number of leaves. □

3 Choice of triangulation

The geometry and topology of the triangulation axis $M_T(P)$ strongly depend on the choice of the underlying triangulation T.

The *constrained Delaunay triangulation* of P seems a good choice at first glance, as it tends to avoid edges at flat convex vertices of P, by its empty circle property. However (and similar to the notorious problem with the medial axis of P), still various small and unimportant features are reflected by this 'Delaunay axis' of P; see Figure 5(a).[1] Also, it lacks optimization properties like maximizing angles or minimizing lengths, as can be shown by examples. An advantage is the low construction time, $O(n \log n)$; see e.g., [5].

The *minimum-weight triangulation* of P appears a promising candidate, too, as a triangulation of short total length might have edges nicely aligned within P. Unfortunately, small length implies many branch triangles, as these triangles are capable of covering much polygon area, leaving less area (and thus edge length) for the most frequent type of axis edges, the link edges; see Figure 5(b). On the other hand, weighting each branch triangle by k times its perimeter, for some large constant k (rather than using $k = 1$ for all triangles) gives quite satisfactory results; see Figure 5(d). Note that the type of each triangle can be read of from the vertices of P it uses. The runtime of $O(n^3)$ that results from dynamic programming [7] is a clear disadvantage, though.

The weight of a triangle can also be chosen as the length of the axis part it yields. This *minimum length triangulation axis*, see Figure 5(c), looks comparable to the axis in Figure 5(d), which was slightly better, however, as no inadequate paths close to leaves do occur that stem from keeping the axis shortest possible. Another possibility is to minimizing the number of branch triangles (weight 1, remaining triangles weight 0). The resulting axis also minimizes the number of leaves, hence maximizes the number of links. The optimal solution is highly ambiguous, and its ability of describing the polygon is inferior to the former choices for most instances. (In fact, a good choice is given by the modified minimum-weight triangulation above, for sufficiently large k.)

The axes in examples (c) and (d) nicely resemble the predominant features of the polygon's medial axis. Clearly, if P is poorly sampled, $M_T(P)$ can be geometrically far apart from the latter structure, for any choice of T; see Figure 6. Still, $M_T(P)$ behaves better than an approximation by the Voronoi diagram of P's vertices, as such skeletons may exit and re-enter the polygon at several places (as in Figure 7).

[1] For branch triangles Δ, we used the Steiner point of Δ's edge midpoints, rather than the centroid of Δ, in this running example. This choice gives slightly better results.

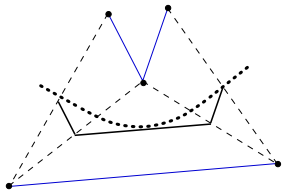


Figure 6: Medial axis deviation within a link triangle

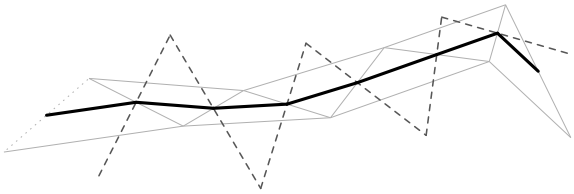


Figure 7: Triangulation axis (bold edges) and Voronoi axis (dashed edges)

4 Edge flipping

Triangulation axes can be modified (and improved) gradually by flipping edges in their defining triangulation. Our goal is to start with the constrained Delaunay triangulation (with low construction time), and to prune away unwanted branches of the Delaunay axis by controlled edge flips. A different pruning approach, based on removing certain axis parts *without* adapting its remaining edges is proposed in Wang [9].

Consult Figure 8. If the bottommost branch of the axis is considered unimportant (e.g., because of its short length) then it can be possibly removed, by flipping in visibility edges to the opposite vertex of the respective branch triangle.

This 'clean up' step can be applied recursively, by establishing a partial order among the branch triangles, starting from the axis' leaves. Each level in this hierarchy takes $O(n)$ flips, because each created edge stays permanent in its level, and there are $O(\log n)$ levels. An overall runtime of $O(n \log n)$ is achieved, including the construction of the initial triangulation. This simple heuristic is surprisingly effective; Figure 1 shows that an axis of quality comparable to the $O(n^3)$ time optimal triangulation approaches in Section 3 is obtained.

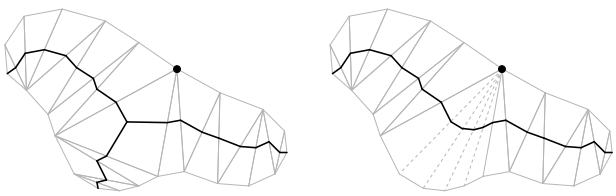


Figure 8: Pruning by flipping

5 Discussion

We have introduced the triangulation axis of a polygon to computational geometry, and we believe in its potential of being a simple, stable, and characteristic skeletal structure.

Several questions are raised by the investigations in the present note. The main issue, of course, is to find a fast and provably good method of adapting the underlying triangulation to the shape of the polygon. We have undertaken first steps towards this goal in Section 4. A convergence result with respect to the medial axis would be desirable, if extraneaous vertices on P's boundary are allowed. Also, is there any notable optimization property of the Delaunay axis of a polygon? Finally, let us mention that the concept of triangulation axis is not restricted to simple polygons, but straightforwardly extends to polygonal regions with holes.

References

[1] T. Ai and P. van Oosterom. GAP-tree extensions based on skeletons. *Proc. 10th Int. Symposium on Advances in Spatial Data Handling*, Springer Verlag, 2002, 501–513.

[2] O. Aichholzer, W. Aigner, F. Aurenhammer, T. Hackl, B. Jüttler, and M. Rabl. Medial axis computation for planar free-form shapes. *Computer-Aided Design* 41 (2009), 339–349.

[3] O. Aichholzer, D. Alberts, F. Aurenhammer, and B. Gärtner. A novel type of skeleton for polygons. *Journal of Universal Computer Science* 1 (1995), 752–761.

[4] D. Attali, J.-D. Boissonnat, and H. Edelsbrunner. Stability and computation of medial axes—a state-of-the-art report. *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration*, T. Müller, B. Hamann, B. Russell (eds.), Springer Series on Mathematics and Visualization, 2008, 109–125.

[5] L.P. Chew. Constrained Delaunay triangulations. *Algorithmica* 4 (1989), 97–108.

[6] L.P. Chew and R.L.S. Drysdale. Voronoi diagrams based on convex distance functions. *Proc. 1st Ann. ACM Symposium on Computational Geometry*, 1985, 235–244.

[7] G.T. Klincsek. Minimal triangulations of polygonal domains. *Annals of Discrete Mathematics* 9 (1980), 121–123.

[8] F. Labelle and J.R. Shewchuk. Anisotropic Voronoi diagrams and guaranteed-quality anisotropic mesh generation. *Proc. 19th Ann. ACM Symposium on Computational Geometry*, 2003, 191–200.

[9] T. Wang. Extraction of optimal skeleton of polygon based on hierarchical analysis. In J. Zhang et al. (eds.), *Int. Archives of Photogrammetry, Remote Sensing, and Spatial Information Sciences*, 28-7/C4, 2009, 272–276.

A Lower Bound for Shallow Partitions

Wolfgang Mulzer* Daniel Werner†

Abstract

We give a lower bound of $\Omega(\log(n/k)/\log\log(n/k))$ for the crossing number of shallow partitions in the plane.

1 Introduction

Range searching is a fundamental problem in computational geometry that has long driven innovation in the field [3]: given a set of n points in d dimensions, find a data structure such that all points inside a given query range can be found efficiently. Depending on the precise nature of the query range and on the dimension, many different versions of the problem can be studied. Consequently, a wide variety of techniques have been developed to address them. Among these tools we can find such classics as range trees and kd-trees [5, Chapter 5], ε-nets and cuttings [7], spanning trees with small crossing number [13], geometric partitions [9], and many more. For several problems, almost matching lower bounds are known (in certain models of computation) [7].

Geometric partitions provide the most effective means for solving the *simplex range searching* problem, where the query range is given by a d-dimensional simplex [6, 9]. They provide a way to subdivide a point set into parts of roughly equal size, such that (i) each part is contained in a simplex; and (ii) any given hyperplane intersects only few of these simplices. This makes it possible to construct a tree-like data structure in which each node corresponds to a simplex in an appropriate geometric partition. With a careful implementation, one can achieve query time $O(n^{1-1/d}+z)$ with linear space [6] (here z is the output size, i.e., the number of reported points).

If the query simplex degenerates to a half-space, we can do better [10]. For this, we need a more specialized version of geometric partitions, called *shallow partitions*. Again, these partitions provide a way for subdividing a d-dimensional point set into parts of roughly equal size, such that each part is contained in a simplex and such that a hyperplane intersects only few of these simplices. This time, however, we restrict ourselves to *shallow* hyperplanes. Such hyperplanes have only few points to one side. Thus, we only have the guarantee that any shallow hyperplane will intersect few simplices of the partition (see below for details). This makes it possible to decrease the number of simplices that are intersected and to achieve better bounds for halfspace range searching. Namely, one can obtain for $d \geq 4$ a linear-space data structure that answers a query in time $O(n^{1-1/\lfloor d/2\rfloor}+z)$, where z is the output size [6] (for $d = 2, 3$, one can achieve query time $O(\log n + z)$ and linear space [2]).

Shallow partitions (as well as their cousins—shallow cuttings) have proved invaluable tools in computational geometry and have found numerous further applications. Nonetheless, there still remain some open questions. As mentioned above, we would like every shallow hyperplane to intersect as few simplices of the shallow partition as possible. But what exactly is possible? For dimension $d \geq 4$, the original bound by Matoušek [9] is known to be asymptotically tight. For lower dimensions, however, Matoušek asked whether his result could be improved. It took almost 20 years until Afshani and Chan [2] provided the first lower bound in three dimensions, almost matching the upper bound. For the plane, however, so far no nontrivial lower bounds appear in the literature.

Here, we will give a construction that provides such a lower bound for shallow partitions in two dimensions. Our result almost matches the upper bound and also gives an alternative proof for the bound of Afshani and Chan [2]. A similar construction has been discovered independently by Afshani [1].

2 Shallow partitions

We begin by providing the details of Matoušek's shallow partition theorem in two dimensions. Let $P \subseteq \mathbb{R}^2$ be a planar n-point set in general position. Let $k \in \{1, \ldots, n\}$ be a parameter. A *k-partition* $\mathcal{P}$ for P consists of two parts: (i) a sequence $P_1, P_2, \ldots, P_{\lceil n/k\rceil}$ of pairwise disjoint subsets of P such that $\bigcup_i P_i = P$ and $|P_i| = k$ for $i = 1, \ldots, \lfloor n/k\rfloor$; and (ii) a sequence $\Delta_1, \Delta_2, \ldots, \Delta_{\lceil n/k\rceil}$ of triangles such that $P_i \subseteq \Delta_i$ for all i.

Now let ℓ be a line that does not contain any point in P, and let ℓ^+ denote the open halfplane above ℓ. We say that ℓ is *k-shallow* if $|\ell^+ \cap P| \leq k$. Given a k-partition $\mathcal{P}$ of P, the *crossing number* of $\mathcal{P}$ is the maximum number of triangles in $\mathcal{P}$ that are intersected by

*Institut für Informatik, Freie Universität Berlin, 14195 Berlin, Germany mulzer@inf.fu-berlin.de

†Institut für Informatik, Freie Universität Berlin, 14195 Berlin, Germany werner@inf.fu-berlin.de. Supported by Deutsche Forschungsgemeinschaft within the Research Training Group (Graduiertenkolleg) "Methods for Discrete Structures".

any k-shallow line. For any given k, the goal is to find a k-partition of P whose crossing number is as small as possible. Matoušek [10, Theorem 3.1] proved the following theorem.

Theorem 1 *Let P be a planar n-point set in general position and let $k \in \{1, \ldots, n\}$. Then there exists a k-partition of P with crossing number $O(\log(n/k))$.* □

Matoušek's original proof uses cuttings and a variant of the iterative reweighting technique (also known as the multiplicative weights update method [4]), and it readily generalizes to higher dimensions. More recently, Har-Peled and Sharir [8, Lemma 3.3] give an approach for proving Theorem 1 with elementary means, but it is not clear whether their technique can be applied to higher dimensions. As mentioned in the introduction, Matoušek [10] asked whether the crossing number in Theorem 1 can be improved to $O(1)$. He conjectured that the answer is no. Afshani and Chan [2] proved that for any k there are arbitrarily large point sets in $\mathbb{R}^3$ such that the crossing number of any k-partition for them is $\Omega\big(\frac{\log(n/k)}{\log\log(n/k)}\big)$. However, their construction does not apply for two dimensions. Hence, we will describe here a different—and arguably simpler—construction that yields the same lower bound for the plane. Independently, Afshani [1] used very similar ideas to obtain the same lower bound.

3 The Lower Bound

Let $a(n, k)$ be the minimum crossing number that a k-partition can achieve for any planar n-point set in general position. For the lower bound, we shall consider the dual setting. We use the standard duality transform along the unit paraboloid that maps the point $p : (p_x, p_y)$ to the line $p^* : y = 2p_x x - p_y$ and vice versa [11].

A point set P dualizes to a set P^* of planar lines. We now define the *k-level* of P^*, $\mathrm{lev}_k(P^*)$ [12]. It is the closure of the set of all points that lie on a line of P^* and that have exactly k lines of P^* beneath them. We observe that $\mathrm{lev}_k(P^*)$ is an x-monotone polygonal curve whose edges and vertices come from the arrangement of P^*. Let $\mathcal{C}$ be the upper convex hull of $\mathrm{lev}_k(P^*)$. For each vertex v of $\mathcal{C}$, we let $P_v^* \subseteq P^*$ denote the set of lines beneath it. We call P_v^* the *conflict set* of v. We have $|P_v^*| = k$,[1] hence v is dual to a k-shallow line v^* in the primal plane.

Now we can interpret shallow partitions in the dual plane:

[1] Note that $\mathrm{lev}_k(P^*)$ may also contain vertices with only $k-1$ lines of P^* beneath them, but these vertices cannot appear on $\mathcal{C}$, since they correspond to a concave bend in $\mathrm{lev}_k(P^*)$.

Proposition 2 *Let $\mathcal{C}$ be an x-monotone downward convex chain, and let L be a set of n lines such that for each vertex v of $\mathcal{C}$ the conflict set L_v has cardinality k. Then there exists a coloring of L such that (i) each color class has size at most k; and (ii) each conflict set L_v contains at most $a(n, k) + 1$ different colors.*

Proof. Consider the primal plane, where $L = P^*$ corresponds to a point set P. By assumption, there exists a k-partition $\mathcal{P}$ of P with crossing number $a(n, k)$. Each vertex v of $\mathcal{C}$ corresponds to a k-shallow line v^*, and at most one triangle of $\mathcal{P}$ can be wholly contained in v^{*+}. Thus, the claim follows from the properties of the duality transform. □

We are now ready to describe the construction. Let $m = 2^\beta$ be a power of 2 and let $\mathcal{C}$ be an x-monotone convex chain with m vertices. We denote these vertices by v_1, …, v_m, from left to right. Now, for $j = 0, \ldots, \beta$, let L_j be a set of $m/2^j$ lines such that the first line in L_j lies exactly below the vertices v_1 to v_{2^j}, the second line lies below v_{2^j+1} to $v_{2\cdot 2^j}$, the third line lies below $v_{2\cdot 2^j+1}$ to $v_{3\cdot 2^j}$, etc. We set $L' := \bigcup_{j=0}^{\beta} L_j$. See Fig. 1.

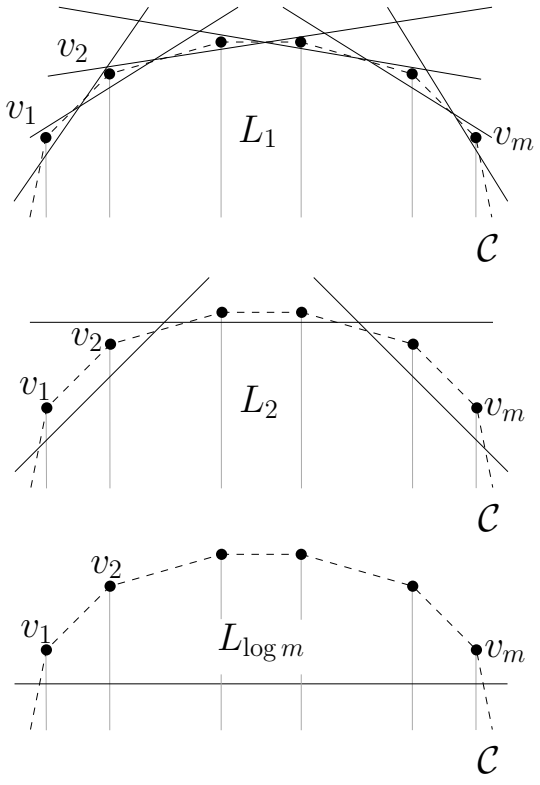


Figure 1: Sets of lines L_j.

Assume for now that k is a multiple of $\beta + 1$, and let L consist of $k/(\beta+1)$ copies of L'. We perturb the lines in L such that they are all distinct while their

relationship with the vertices of $\mathcal{C}$ remains unchanged. It follows that L has exactly $n := (2m-1)k/(\beta+1)$ lines, with exactly k lines in each conflict set L_{v_i} (recall that by definition $\beta = \log m$).

By Proposition 2, there is a coloring of L such that each color class has size at most k and such that each conflict set contains at most $a(n,k)+1$ colors. The structure of L lets us interpret this coloring as follows: let T be a complete binary tree with $2m-1$ nodes and height β. We label the leaves of T with the vertices $v_1, \ldots, v_m$, from left to right. Thus, every node w of T corresponds to an interval of consecutive vertices of $\mathcal{C}$, namely the leaves of the subtree rooted in w. By assigning to w the lines that lie exactly below the vertices in this interval, we obtain a partition of L into sets of size $k/(\beta+1)$. This leads to an interpretation of shallow partitions as multi-colorings of trees.

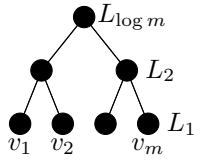


Figure 2: A tree with height $\beta = \log n$. The leaves correspond to the vertices v_i, and the level of height i corresponds to the lines in L_i. A line ℓ is stored in the node whose subtree corresponds to the vertices that have ℓ below them.

Proposition 3 *Let T be a complete binary tree with height $\beta = \log m$ and $2m-1$ nodes, and let k be a multiple of $\log m + 1$. Then there exists a multi-coloring of the nodes of T with the following properties: (i) every node is associated with a multiset of $k/(\beta+1)$ colors; (ii) each color class has at most k elements; (iii) along each root-leaf path there are at most $a(n,k)+1$ distinct colors, where $n = (2m-1)k/(\beta+1)$.*

Proof. Properties (i) and (ii) follow immediately from Proposition 2 and the construction. For Property (iii), observe that the lines encountered along a root-leaf path are exactly the lines below the vertex of $\mathcal{C}$ corresponding to the leaf. □

We can now prove the desired lower bound.

Lemma 4 *Let T be a complete binary tree with height $\beta = \log m$ and $2m-1$ nodes, and let k be a multiple of $\log m + 1$. Consider a multi-coloring of T such that (i) every node is associated with a multiset of $k/(\beta+1)$ colors; and (ii) each color class has at most $2k$ elements. Then there exists a root leaf-path with $\Omega(\log m/\log\log m)$ distinct colors.*

Proof. We subdivide the nodes of T into *slices*. The first slice consists of the first $\lceil\log(3\beta)\rceil$ levels of T, the second slice consists of the following $\lceil\log(6\beta)\rceil$ levels, the third slice has the next $\lceil\log(9\beta)\rceil$ levels, and so on. In general, the ith slice consists of $\lceil\log(3i\beta)\rceil$ consecutive levels of T.

We claim that there exists a root-leaf path that has at least one distinct color for each slice that it crosses, except for the last one. To see this, we first consider a complete subtree T' of T that is has its root in the first level of a slice i and its leaves in the last level of the same slice. As a complete binary tree with $\lceil\log(3i\beta)\rceil$ levels, T' has at least $3i\beta - 1 \geq 2i\beta + 2i$ nodes. Therefore, our multi-coloring needs to assign at least $2(i\beta+i)k/(\beta+1)$ colors in T'. Since each color class has size at most $2k$, this requires at least i *distinct* colors.

We now construct the required root-leaf path slice by slice. Throughout, we maintain the invariant that after i slices have been considered, the path contains at least i distinct colors. This is certainly true at the root. Now suppose that we have constructed a partial path Q_{i-1} that ends at a node z in the last level of the $(i-1)$th slice. If Q_{i-1} contains at least i distinct colors, we arbitrarily extend it to a path Q_i that ends at the bottom of the ith slice. Otherwise, we pick an arbitrary child z' of z. As noted above, the complete subtree that is rooted at z' and restricted to the ith slice contains at least i distinct colors. Thus, we can extend Q_{i-1} through z' to a path Q_i that goes to the bottom of the ith slice and that meets at least i distinct colors. The claim follows.

It remains to calculate a lower bound for the number of slices b. By construction, we must have

$$\sum_{i=1}^{b} \lceil\log(3i\beta)\rceil \geq \beta + 1.$$

Now,

$$\begin{aligned}\sum_{i=1}^{b} \lceil\log(3i\beta)\rceil &\leq \sum_{i=1}^{b} \log(4i\beta)\\ &\leq b(2 + \log b + \log\beta)\\ &\leq 3b\log\beta,\end{aligned}$$

since clearly $b \leq \beta$. Hence,

$$b \geq \frac{\beta+1}{3\log\beta} = \Omega\Big(\frac{\log m}{\log\log m}\Big),$$

as desired. □

We now indicate how to drop the assumption that k is a multiple of $\beta+1$. Indeed, suppose that this is not the case, but $k \geq \beta+1$. We first perform the above construction with $k' := \lfloor k/(\beta+1)\rfloor(\beta+1)$ instead of k. Note that since $k \geq \beta+1$, we have $k \geq k'$. Then we add $k-k'$ suitably perturbed copies of L_β (the set containing a line in conflict with all vertices of $\mathcal{C}$).

Let L be the resulting set of lines. By Proposition 2, there exists a coloring of L such that each color class has at most $k \leq 2k'$ elements and such that each conflict set has at most $a(|L|, k) + 1$ distinct colors. The tree T corresponding to L has the same structure as before, but now each non-leaf node except the leaf is associated with $k'/(\beta + 1)$ colors, while the leaves have $k - k'$ additional colors. This suffices for the argument of Lemma 4 to go through.

Theorem 5 *There is a constant $c > 0$ such that the following holds. For every n and $k \in \{\log n, \ldots, n/4\}$, there exists a planar n-point set P such that the crossing number for any k-partition of P is at least $c \log(n/k)/\log\log(n/k)$. Thus,*

$$a(n, k) = \Omega\left(\frac{\log(n/k)}{\log\log(n/k)}\right).$$

Proof. Let $\beta \in \mathbb{N}$ be maximum with $(2^{\beta+1} - 1)/(\beta + 1) \leq n/2k$. Set $m := 2^\beta$ and $k' := \lfloor k/(\beta+1) \rfloor (\beta+1)$.

From Propositions 2 and 3 and Lemma 4, it follows that by taking the dual we obtain a set P' of $n' := (2m - 1)k'/(\beta + 1) + k - k'$ points such that any k-partition of P' has crossing number at least $c' \log m/\log\log m$, for some constant $c' > 0$.

First note that $\beta < \log n$ and $k \geq \beta + 1$. Hence, $k' \leq k \leq 2k'$ and $k - k' \leq \log n$. Thus, we can conclude that

$$n' = \frac{2m - 1}{\log m + 1} k' + k - k' \leq \frac{n}{2} + \log n \leq n.$$

and

$$n' = \frac{2m - 1}{\log m + 1} k' + k - k' \geq \frac{n}{4k} \cdot \frac{k}{2} = \frac{n}{8}.$$

Thus, by adding at most $7n/8$ points that are contained in no k-shallow halfplane, we can obtain from P' a point set P with n points and crossing number at least $c \log m/\log\log m$. Finally, observe that

$$m \geq \frac{n'}{k'} - k \geq \frac{n}{9k},$$

so P also has crossing number at least $c \cdot \frac{\log(n/k)}{\log\log(n/k)}$, for some $c > 0$. The result follows. □

Note that our construction also implies a similar lower bound in $\mathbb{R}^3$ by embedding the plane into three-dimensional space and perturbing the points slightly. This provides an alternative proof of the result by Afshani and Chan [2].

4 Conclusion and Open Problems

We have given a simple construction that give a lower bound of $\Omega\left(\frac{\log(n/k)}{\log\log(n/k)}\right)$ for the crossing number of any shallow partition of a planar point set. Matoušek's result gives an upper bound of $O(\log(n/k))$. Thus, there still remains a factor of $\log\log(n/k)$ to be settled. Can we show that Matoušek's analysis is tight? Or, perhaps more interestingly, can we construct shallow partitions with crossing number $O\left(\frac{\log(n/k)}{\log\log(n/k)}\right)$?

Acknowledgments

We would like to thank Nabil Mustafa for drawing our attention to shallow partitions and for enlightening conversations. We would also like to thank Timothy Chan for answering our questions about shallow partitions.

References

[1] P. Afshani. Unpublished manuscript.

[2] P. Afshani and T. M. Chan. Optimal halfspace range reporting in three dimensions. In *Proc. 20th Annu. ACM-SIAM Sympos. Discrete Algorithms (SODA)*, pages 180–186, 2009.

[3] P. K. Agarwal and J. Erickson. Geometric range searching and its relatives. In *Advances in discrete and computational geometry*, volume 223 of *Contemp. Math.*, pages 1–56. 1999.

[4] S. Arora, E. Hazan, and S. Kale. The multiplicative weights update method: a meta-algorithm and its applications. *To appear.*

[5] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational geometry: algorithms and applications.* Springer-Verlag, Berlin, third edition, 2008.

[6] T. M. Chan. Optimal partition trees. In *Proc. 26th Annu. ACM Sympos. Comput. Geom. (SoCG)*, pages 1–10. 2010.

[7] B. Chazelle. *The discrepancy method: randomness and complexity.* Cambridge University Press, Cambridge, 2000.

[8] S. Har-Peled and M. Sharir. Relative (p, ϵ)-approximations in geometry. *Discrete Comput. Geom.*, 45(3):462–496, 2011.

[9] J. Matoušek. Efficient partition trees. *Discrete Comput. Geom.*, 8(3):315–334, 1992.

[10] J. Matoušek. Reporting points in halfspaces. *Comput. Geom. Theory Appl.*, 2(3):169–186, 1992.

[11] K. Mulmuley. *Computational Geometry: An Introduction through Randomized Algorithms.* Prentice-Hall, Englewood Cliffs, 1994.

[12] M. Sharir and P. K. Agarwal. *Davenport-Schinzel sequences and their geometric applications.* Cambridge University Press, New York, NY, USA, 1995.

[13] E. Welzl. On spanning trees with low crossing numbers. In *Data structures and efficient algorithms*, volume 594 of *Lecture Notes in Comput. Sci.*, pages 233–249. Springer, Berlin, 1992.

An extension of a Theorem of Yao & Yao

Edgardo Roldán-Pensado* Pablo Soberón†

Abstract

In this paper we study $N_d(k)$, the smallest positive integer such that any nice measure μ in $\mathbb{R}^d$ can be partitioned in $N_d(k)$ parts of equal measure so that every hyperplane avoids at least k of them. A theorem of Yao and Yao [10] states that $N_d(1) \leq 2^d$. Among other results, we obtain the bounds $N_d(2) \leq 3 \cdot 2^{d-1}$ and $N_d(1) \geq C \cdot 2^{d/2}$ for some constant C. We apply these results to a problem on the separation of points and hyperplanes.

1 Introduction

A convex partition of $\mathbb{R}^d$ into n parts is a covering $\mathcal{P} = \{C_1, \ldots, C_n\}$ of $\mathbb{R}^d$ consisting of closed convex bodies with pairwise disjoint interiors. We say that a hyperplane $H \subset \mathbb{R}^d$ avoids a set C if it does not intersect its interior.

The classical Yao-Yao Theorem [10] states the following.

Theorem 1 (Yao and Yao, 1985) *Let μ be a nice measure in $\mathbb{R}^d$, then there is a convex partition $\mathcal{P}$ of $\mathbb{R}^d$ into 2^d parts of equal μ-measure such that every hyperplane in $\mathbb{R}^d$ avoids at least one element of $\mathcal{P}$.*

In the original proof of this theorem the measure has to have a continuous density function bounded away from 0. Later, in [5], this was weakened to the condition that the measure of every hyperplane be 0. In this paper, we ask μ to satisfy the original conditions, as in [10].

We extend this theorem to the case when every hyperplane is required to avoid 2 pieces.

Theorem 2 *Let μ be a nice measure in $\mathbb{R}^d$, then there is a convex partition $\mathcal{P}$ of $\mathbb{R}^d$ into $3 \cdot 2^{d-1}$ parts of equal μ-measure such that every hyperplane in $\mathbb{R}^d$ avoids at least two elements of $\mathcal{P}$.*

For $d = 2$ this follows from a theorem by Buck and Buck [3] which states that $\mathbb{R}^2$ can be divided into six parts of equal measure by three concurrent lines. Our method gives in this case a partition by three lines, two of which are parallel. The proof of this theorem is given in section 4. Some details of Yao and Yao's original proof of their results are needed, so a sketch of this proof is given in section 3.

Let $N_d(k)$ be the smallest positive integer such that the following holds: *For every nice measure μ on $\mathbb{R}^d$ there exists a partition of $\mathbb{R}^d$ into $N_d(k)$ convex parts of equal measure such that every hyperplane avoids the interior of at least k parts.* We call such a partition a k-equipartition. Yao-Yao's Theorem and Theorem 2 are equivalent to the bounds $N_d(1) \leq 2^d$ and $N_d(2) \leq 3 \cdot 2^{d-1}$.

Let $M_d(k, \alpha)$ be the smallest positive integer such that for every nice measure μ on $\mathbb{R}^d$ there is a family of $M_d(k, \alpha)$ convex sets such that every hyperplane avoids the interior of at least k of them and they all have measure at least α. Clearly we have

$$N_d(k) \geq M_d\left(k, \frac{1}{N_d(k)}\right).$$

Proposition 3 *Let $p \leq q$ be non-negative integers, then $M_2(q-p, \frac{p}{2q}) \leq 2q$.*

From the proof of this proposition we also obtain a bound for N_2.

Corollary 4 $N_2(k) \leq 2k + 2$.

This improves the bounds that can be obtained by using Yao-Yao's Theorem and Theorem 2.

For a fixed d, we can determine the asymptotic behaviour of $N_d(k)$.

Theorem 5 $\lim_{k\to\infty} \frac{N_d(k)}{k} = 1.$

These results are proven in section 5. In section 2, they are applied to obtain new bounds on the following problem.

Problem: Determine all pairs $(\alpha, \beta) \in \mathbb{R}^2_+$ such that for any finite set X of points in $\mathbb{R}^d$ and any finite set Y of hyperplanes in $\mathbb{R}^d$, there are sets $A \subset X$ and $B \subset Y$ such that $|A| \geq \alpha|X|$, $|B| \geq \beta|Y|$ and no two points in A are separated by a hyperplane in B.

Trying to solve this question we obtained a lower bound for $N_d(1)$ in terms of the regularised incomplete beta function I (see [6] for a definition).

Theorem 6 *If $1 \geq \alpha \cdot M_d(1, \alpha)$, then*

$$\frac{1}{\alpha} \geq I^{-1}_{\sin^2(t)}\left(\frac{d}{2}, \frac{1}{2}\right) \approx C \cdot 2^{\frac{d}{2}}$$

*Department of Mathematics, University College London, e.roldan@math.ucl.ac.uk

†Department of Mathematics, University College London, pablo.soberon@ciencias.unam.mx

for some constant $C > 0$.

Corollary 7 $N_d(1) \geq C \cdot 2^{\frac{d}{2}}$.

These results follow from Theorems 8 and 11 below. This kind of bound can be obtained for $N_d(k)$ for any k in terms of the regularised incomplete beta function.

Remark. The Yao-Yao Theorem can be generalised in the following way: Given $a_1, a_2, \ldots, a_{2^d} > 0$ such that $\sum a_i = \mu(\mathbb{R}^d)$, there is a partition of $\mathbb{R}^d$ into 2^d convex parts $\{C_1, C_2, \ldots, C_{2^d}\}$ such that $\mu(C_i) = a_i$ for all i and every hyperplane avoids the interior of at least one C_i. A similar generalization exists for Theorem 2 as well.

2 The (α, β) problem

The (α, β) problem deals with how well behaved points and hyperplanes are with each other in terms of separation. This problem was told to us by Imre Bárány in its first version, but it has the difficulty that it is not self-dual. We work with a second version which does have this property.

Problem: Find all pairs $(\alpha, \beta) \in \mathbb{R}^2_+$ such that for any nice probability measures μ_1, μ_2 in S^d there are sets $A, B \subset S^d$ with $\mu_1(A) \geq \alpha$, $\mu_2(B) \geq \beta$ such that either $a \cdot b \geq 0$ for all $a \in A$, $b \in B$ or $a \cdot b \leq 0$ for all $a \in A$, $b \in B$.

Here we may assume without loss of generality that the measures μ_1 and μ_2 are centrally symmetric.

Let $\mathcal{C}'_d$ be the set of pairs (α, β) that satisfy the conditions of the original problem and $\mathcal{C}_d$ be the set of pairs that satisfy the conditions of the second version of the problem. It can be shown that $(\alpha, \beta) \in \mathcal{C}'_d$ if and only if $(\frac{\alpha}{2}, \frac{\beta}{2}) \in \mathcal{C}_d$.

We shall denote by M^d usual probability measure on S^d. Given $0 \leq t \leq \frac{\pi}{2}$ and $x \in S^d$, let $\mathcal{C}(d, x, t)$ be the spherical cap of S^d with centre x and central angle t, define $h_d(t)$ as its M^d-measure.

Given $A \subset S^d$, let $A^\perp$ be the set of points $x \in S^d$ such that there exists $a \in A$ with $a \cdot x = 0$.

Fix a closed subset A of S^d and set $t > 0$ so that $M^d(A) = h_d(t)$. Define A_ε as the set of points $x \in S^d$ for which there exists $a \in A$ with $\arccos(a \cdot x) < \varepsilon$. Theorem 2.1 in [4] states the following: *For every $\varepsilon > 0$, $\mu_d(A_\varepsilon) \geq h_d(t + \varepsilon)$.* If $\varepsilon = \frac{\pi}{4}$ then $S^d \setminus A_\varepsilon$ is one of the two connected components of $S^d \setminus A^\perp$. This gives the following theorem.

Theorem 8 *All points in $\mathcal{C}_d$ lie on or below the curve*

$$\left\{\left(h_d(t), h_d\left(\frac{\pi}{2} - t\right)\right) : 0 \leq t \leq \frac{\pi}{2}\right\}.$$

The function $h_d(t)$ can be described using the regularised incomplete beta function (see [6]) as

$$h_d(t) = \frac{1}{2} I_{\sin^2(t)}\left(\frac{d}{2}, \frac{1}{2}\right).$$

Theorem 9 $\mathcal{C}_1 = \left\{(\alpha, \beta) : \alpha + \beta \leq \frac{1}{2}\right\}$.

Proof. Take α, β with $\alpha + \beta \leq \frac{1}{2}$. Suppose that for every arc segment A with $\mu_1(A) = \alpha$ we have that $\mu_2(A^\perp) > \alpha$. Note that each of the two components of $A^\perp$ is obtained by rotating A by an angle of $\pm\frac{\pi}{2}$. This implies that $\mu_1(S^1) < \mu_2(S^1)$, which is a contradiction. Therefore there exists an arc segment A that satisfies $\mu_2(A^\perp) \leq \alpha$. If B is any of the two components of $S^1 \setminus A^\perp$, then $\mu_2(B) \geq \beta$. This proves one of the inclusions, Theorem 8 gives us the other. □

Lemma 10 *Let $0 \leq \rho \leq 1$. Suppose that for any nice measure μ_1 on S_d there exists a family F of subsets of S^d and a probability measure μ_F on F such that*

- *$\mu_1(A) \geq \alpha$ for all $A \in F$,*
- *For every $b \in S^d$, the set $F_b = \{A \in F : A \cap \{b\}^\perp \neq \emptyset\}$ is μ_F-measurable and $\mu_F(F_b) \leq \rho$.*

Then $(\alpha, \frac{1-\rho}{2}) \in \mathcal{C}_d$.

Proof. Let μ_1 and μ_2 be nice measures. Suppose that we can find μ_F as above, then using Fubini's Theorem

$$\int_F \mu_2(A^\perp) d\mu_F = \int_{S^d} \mu_F(F_b) d\mu_2 \leq \rho.$$

Thus, we can find A_0 such that $\mu_2(A_0^\perp) \leq \rho$. This means that there is a set B such that $\mu_2(B) \geq \frac{1-\rho}{2}$ and the sign of $a \cdot b$ is constant for all $a \in A_0$ and $b \in B$. □

Using this with $N_d(k)$ and $M_d(k, \alpha)$ and a projection from a hyperplane in $\mathbb{R}^{d+1}$ to S^d, we obtain the following.

Theorem 11 *For any two positive integers k and d, $\left(\frac{1}{2N_d(k)}, \frac{k}{2N_d(k)}\right) \in \mathcal{C}_d$. More generally, if $\alpha > 0$, $\left(\frac{\alpha}{2}, \frac{k}{2M_d(k,\alpha)}\right) \in \mathcal{C}_d$.*

With this Theorem and Theorem 8 we obtain the lower bounds in Theorem 6. It should be noted that this also implies lower bounds for $N_d(k)$ for any k.

Applying the results known for $N_d(k)$, we obtain the following.

Corollary 12 *For any two non-negative integers k_1 and k_2, not both equal to 0, let $\alpha = \left(\frac{1}{2^d}\right)^{k_1}\left(\frac{1}{3\cdot 2^{d-1}}\right)^{k_2}$ and $\beta = 1 - \left(1 - \frac{1}{2^d}\right)^{k_1}\left(1 - \frac{1}{3\cdot 2^{d-2}}\right)^{k_2}$. Then $\frac{1}{2}(\alpha, \beta) \in \mathcal{C}_d$.*

This gives in particular that $\left(\frac{1}{2^{d+1}}, \frac{1}{2^{d+1}}\right) \in \mathcal{C}_d$, which was also obtained in [1] with similar methods.

We can get better bounds if further conditions are imposed on one of the measures. Let $\mathcal{C}_d(\Delta)$ be the set of pairs (α, β) such that for any two measures μ_1, μ_2 on S^d such that μ_1 is the integral of a Lipschitz

function f with $\mathrm{Lip}(f) \leq \Delta$, there are sets A, B in S^d satisfying $\mu_1(A) = \alpha$, $\mu_2(B) = \beta$ and either $a \cdot b \geq 0$ for all $a \in A$, $b \in B$ or $a \cdot b \leq 0$ for all $a \in B$, $b \in B$.

Theorem 13 *For every $0 < \lambda \leq 1$ and $0 < r < \frac{1-\lambda}{\Delta}$, let $\alpha = \lambda h_d(r)$ and $\beta = h_{d-1}(\frac{\pi}{2} - 2\arcsin(\sin(r)\sin^{-1}(\frac{1-\lambda}{\Delta} - r)))$. Then $(\alpha, \beta) \in \mathcal{C}_d(\Delta)$.*

The idea for the proof is to use a small caps centred in a small $(d-1)$-sphere contained in S^d and the Lipschitz condition to have sets as in the proof of Lemma 10. If r is close to 0, then the pairs obtained are close to $\left(\lambda h_d(r), h_{d-1}\left[\frac{\pi}{2} - c_2 r\right]\right)$ for a constant c_2 depending on λ and Δ. If instead of this we use a hypercube (of dimension d) in S^d, we obtain bounds of the type $(\frac{c_1}{m^{d-1}}, \frac{1}{2} - \frac{c_2 d}{m})$. These are worse than the ones in Theorem 13 but are easier to grasp.

3 Yao-Yao's original proof

Let $O(d)$ be the space consisting of $d \times d$ matrices u such that $u^T u = I$, the space $SO(d) \subset O(d)$ consists of matrices with determinant 1. A matrix $u \in O(d)$ can be expressed as $u = (u_1, \ldots, u_d)$ where u_i is the i-th row vector of u, in this way every u can be identified with an ordered orthonormal base of $\mathbb{R}^d$.

Fix a base $u = (u_1, \ldots, u_d)$ of $\mathbb{R}^d$. If H is a hyperplane orthogonal to u_1, define the open half-spaces $H^+ = \{x + tu_1 : x \in H, t > 0\}$, $H^- = \{x - tu_1 : x \in H, t > 0\}$.

Let v be a unit vector in $\mathbb{R}^d$ not orthogonal to u_1 and let $p_v : \mathbb{R}^d \to H$ be the projection such that $p_v(x + tv) = x$ for all $x \in H$ and $t \in \mathbb{R}$. We can identify H with $\mathbb{R}^{d-1}$ by means of the base $u_2, \ldots, u_d$. There is a natural way to define measures μ_v^+ and μ_v^- in H: For any measurable $S \subset H$, set $\mu^{\pm}(S) = \mu(p_v^{-1}(S) \cap H^{\pm})$.

In [10] a centre $c \in \mathbb{R}^d$ for μ relative to the base $u_1, \ldots, u_d$ is defined as follows:

- If $d = 1$ then c is the point that splits $\mathbb{R}$ into two parts of equal μ-measure.
- If $d > 1$, let H be the hyperplane orthogonal to u_1 that splits $\mathbb{R}^d$ into two parts of equal μ-measure. Then c lies on H and there exists a unit vector v (with $u_1 \cdot v > 0$) such that c is a centre for both μ_v^+ and μ_v^- relative to the base $(u_2, \ldots, u_d)$.

This induces a partition into 2^d parts, if a hyperplane intersects the line through c parallel to v in H^+ then it avoids one of the elements of the partition contained in H^- and vice versa.

Then it is proved that c is exists and is unique. It is easy to see that the projection vector v is also unique and that v and c vary continuously with u.

If we want each element of the partition to have a pre-described value, then the same proof works by changing the choice of H appropriately.

4 Proof of Theorem 2

Problems involving partitions of measures are topological in nature. A common method to approach them is the test map scheme (e.g. [7], [9]). We follow this scheme and reduce the problem to showing that some equivariant functions always have a zero.

Given $x = (x_1, \ldots, x_n) \in \mathbb{R}^{d_1} \times \cdots \times \mathbb{R}^{d_n}$, we define $g_i(x)$ as the result of changing the sign of the i-th coordinate of x. We always use g_i to denote this function independently of the target space, since it causes no confusion. The j-th coordinate of $x_i \in \mathbb{R}^{d_i}$ will be denoted by $x_i^{(j)}$.

We start with the geometrical part of the proof of Theorem 2 and continue with the topological part.

Proof. Let $u = (u_1, \ldots, u_d)$ be an orthonormal base of $\mathbb{R}^d$, we think of u_1 as the upwards direction. If H is a hyperplane orthogonal to u_1, define the open half-spaces $H^+ = \{x + tu_1 : x \in H, t > 0\}$, $H^- = \{x - tu_1 : x \in H, t > 0\}$.

Let H_1 and H_2 be the hyperplanes orthogonal to u_1 such that the sets $A = H_1^+$, $B = H_1^- \cap H_2^+$ and $C = H_2^-$ have equal μ-measure. Let $\mu_1 = \mu|A \cup B$ and $\mu_2 = \mu|B \cup C$.

Yao-Yao's Theorem applied to μ_1 gives a unique centre $O_1 \in H_1$ and a unique projection vector v_1 pointing downwards (i.e. $u_1 \cdot v_1 < 0$).

Let $J_1 \subset H_1$ be the $(d-2)$-dimensional flat through O_1 orthogonal to u_2. Note that the hyperplane $K_1 = \{J_1 + tv_1 : t \in \mathbb{R}\}$ splits B into two parts of equal μ_1-measure.

Define analogously $O_2 \in H_2$, v_2 pointing upwards, J_2 and K_2. Since K_1 and K_2 each divide B into two parts of equal μ-measure, they intersect in a $d-2$-dimensional flat $J \subset B$ parallel to J_1 and J_2.

The centres O_1 and O_2 as well as the vectors v_1 and v_2 vary continuously with u.

Our aim is to find u such that the vectors v_1 and v_2 are parallel to the line O_1O_2. Once this is done, let $\mathcal{P}_1$ and $\mathcal{P}_2$ be the correspondings Yao-Yao partition to μ_1 and μ_2.

Then we can use the partition $\mathcal{P}$ consisting of the elements of $\mathcal{P}_1$ contained in A, the elements of $\mathcal{P}_2$ contained in C and the non-empty elements $K \cap B$ such that $K \in \mathcal{P}_1$. Every hyperplane avoids at least two elements of $\mathcal{P}$.

Let $r : \mathbb{R}^d \to \mathbb{R}^{d-1}$ be the projection such that $r(O_1) = r(O_2) = O$ and $\{r(u_2), \ldots, r(u_d)\}$ is the canonical basis.

The affine hyperplane $r(J)$ is orthogonal to u_2, so it is of the form $\{v : v \cdot u_2 = \lambda\}$ for some $\lambda \in \mathbb{R}$. Let $x \in \mathbb{R}^{d-2}$ and $y \in \mathbb{R}^{d-2}$ be the vectors consisting of the last $d-2$ coordinates of $r(v_1)$ and $r(v_2)$, respectively.

Let $f : O(d) \to \mathbb{R}^{d-1} \times \cdots \times \mathbb{R}^1$ be defined by

$$f(u) = ((x + y, \lambda), x - y) \in \mathbb{R}^{d-1} \times \mathbb{R}^{d-2}.$$

Note that if $h(u) = 0$ for some u, then the vectors v_1 and v_2 are parallel to the line O_1O_2 and we are done. We prove something more general.

For $v \in \mathbb{R}^{d-1} \times \cdots \times \mathbb{R}^1$, let $v^{(j)} = (v_1^{(j)}, \ldots, v_{d-j}^{(j)})$ and $v^T = (v^{(1)}, \ldots, v^{(d-1)})$.

Claim. *Assume $f : O(d) \to \mathbb{R}^{d-1} \times \cdots \times \mathbb{R}^1$ is a function such that whenever $f(u) = v$, then*

- $f(g_1(u)) = g_2(v)$,
- $f(g_2(u)) = g_{d-1}(v^T)^T$,
- $f(g_{i+2}(u)) = g_i(v^T)^T$ *for* $i = 1, \ldots, d-3$.

Then there exists $u \in O(d)$ such that $f(u) = 0$.

Consider the restriction $f^* = f|SO(d)$. Together with composition, we can think of $\{g_1 \circ g_d, \ldots, g_{d-1} \circ g_d\}$ as a set of generators of the group $\mathbb{Z}_2^{d-1}$. Given the conditions on f, there are natural group actions of $\mathbb{Z}_2^{d-1}$ on $SO(d)$ and $\mathbb{R}^{d-1} \times \cdots \times \mathbb{R}^1$ such that f^* is equivariant.

From here we use a similar proof method to the one Bárány used to prove the Borsuk-Ulam Theorem in [2]. This method is thoroughly explained in Chapter 2.2 of [7] and a generalization is given in [8]. For this we need a continuous and equivariant function $f_0 : SO(d) \to \mathbb{R}^{d-1} \times \ldots, \mathbb{R}^1$ that is generic enough and has exactly 2^{d-1} zeros . Such a function is the one given by $f_0(u) = (v_1, \ldots, v_{d-1})$, where

- $v_1 = (u_3^{(1)}, \ldots, u_d^{(1)}, u_2^{(1)})$,
- $v_2 = u_1^{(1)} \cdot (u_3^{(2)}, \ldots, u_d^{(2)})$,
- $v_{i+2} = (u_3^{(i+2)}, \ldots, u_{d-i}^{(i+2)})$ for $i = 1, \ldots, d-3$.

If $f_0(u) = 0$ then u_i is the i-th element of the canonical basis or its negative. Therefore f_0 has exactly 2^{d-1} zeros in $SO(d)$.

We leave the rest of the present proof to the interested reader. □

5 Other proofs

Proof. [Proof of Theorem 3] Set a parameter $t \geq 0$, we proceed inductively. Let ℓ_1 be a oriented halving line. Once that ℓ_k has been constructed, let ℓ_{k+1} be the oriented halving line such that the sets

$$A_k = \{x \in \mathbb{R}^2 : x \text{ is right of } \ell_{k+1} \text{ and left of } \ell_k\},$$
$$A_{q+k} = \{x \in \mathbb{R}^2 : x \text{ is left of } \ell_{k+1} \text{ and right of } \ell_k\}$$

have μ-measure $\frac{p}{2q} + t$, for $k = 1, \ldots, q$. If $t = 0$ then the sum of the measures of these sets up to $k = q$ is p, but they may not cover p times every point of $\mathbb{R}^2$. There is a smallest t such that each point of $\mathbb{R}^2$ is covered at least p times. In this case ℓ_1 and ℓ_{q+1} are equal as sets, so in total we have q lines and the boundary of every A_k is contained in the union of two of them. If $\ell \subset \mathbb{R}^2$ is a typical line then it intersects each of the lines $\ell_1, \ldots, \ell_q$ once, therefore it intersects exactly $p + q$ elements of F. □

Proof. [Proof of Corollary 4] In the previous proof put $p = 1$ and $q = k + 1$, then the resulting sets form a partition. □

Proof. [Proof of Theorem 5] Clearly $N_d(k) \geq d + k$, as there is a hyperplane through any d given points. The function N_d also satisfies

$$N_d(k_1 + k_2) \leq N_d(k_1) + N_d(k_2). \tag{1}$$

To see this, partition $\mathbb{R}^d$ by a hyperplane that divides its measure in proportions $N_d(k_1) : N_d(k_2)$. Then find a k_1-equipartition of one side and a k_2-equipartition of the other side. We also have the asymptotically stronger equation

$$N_d(k_1 N_d(k_2) + k_2 N_d(k_1) - k_1 k_2) \leq N_d(k_1) N_d(k_2). \tag{2}$$

This can be shown by finding a k_1-equipartition of $\mathbb{R}^d$ and further partition each of its pieces by k_2-equipartitions. Starting with Yao-Yao's Theorem and iterating (2), a sequence of partitions can be found such that $N_d(k_i)/k_i$ tends to 1. This together with (1) implies the desired limit. □

References

[1] N. Alon, J. Pach, R. Pinchasi, R. Radoičić, and M. Sharir. Crossing patterns of semi-algebraic sets. *J. Combin. Theory Ser. A*, 111(2):310–326, 2005.

[2] I. Bárány. Borsuk's theorem through complementary pivoting. *Math. Program.*, 18:84–88, 1980.

[3] R. C. Buck and E. F. Buck. Equipartition of convex sets. *Math. Mag.*, 22(4):pp. 195–198, 1949.

[4] T. Figiel, J. Lindenstrauss, and V. Milman. The dimension of almost spherical sections of convex bodies. *Acta Math.*, 139:53–94, 1977.

[5] J. Lehec. On the Yao-Yao partition theorem. *Arch. Math. (Basel)*, 92:366–376, 2009.

[6] S. Li. Concise formulas for the area and volume of a hyperspherical cap. *Asian J. Math. Stat.*, 4(1):66–70, 2011.

[7] J. Matoušek. *Using the Borsuk-Ulam theorem.* Universitext. Springer-Verlag, Berlin, 2003.

[8] O. R. Musin. Borsuk-Ulam type theorems for manifolds. arXiv:0912.0787v6 [math.CO], 2011.

[9] P. Soberón. Balanced convex partitions of measures in $\mathbb{R}^d$. *Mathematika*, 58:71–76, 2012.

[10] A. C. Yao and F. F. E. Yao. A general approach to d-dimensional geometric queries. *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 163–168. ACM, 1985.

What makes a Tree a Straight Skeleton?*

O. Aichholzer† H. Cheng‡ S.L. Devadoss§ T. Hackl† S. Huber¶ B. Li§ A. Risteski‖

Abstract

Let G be a cycle-free connected straight-line graph with predefined edge lengths and fixed order of incident edges around each vertex. We address the problem of deciding whether there exists a simple polygon P such that G is the straight skeleton of P. We show that for given G such a polygon P might not exist, and if it exists it might not be unique. For small star graphs and caterpillars we give necessary and sufficient conditions for constructing P.

1 Introduction

The straight skeleton $\mathcal{S}(P)$ of a simple polygon P is a skeleton structure like Voronoi diagrams, but consists of straight-line segments only. Its definition is based on a so-called *wavefront propagation* process that corresponds to mitered offset curves. Each edge e of P emits a wavefront that moves with unit speed to the interior of P. Initially, the wavefront of P consists of parallel copies of edges of P. However, during the wavefront propagation, topological changes occur: An *edge event* happens if a wavefront edge shrinks to zero length. A *split event* happens if a reflex wavefront vertex meets a wavefront edge and splits the wavefront into pieces, see Figure 1 (right). The *straight skeleton* $\mathcal{S}(P)$ is defined as the set of loci that are traced out by the wavefront vertices and it partitions P into polygonal faces. Each face $f(e)$ belongs to a unique edge e of P. Each straight skeleton edge belongs to two faces, say $f(e_1)$ and $f(e_2)$, and lies on the bisector of e_1 and e_2. Straight skeletons have many applications, like automated roof construction, computation of mitered offset curves, and solving fold-and-cut problems. See [4] and Chapter 5.2 in [3] for further information and detailed definitions.

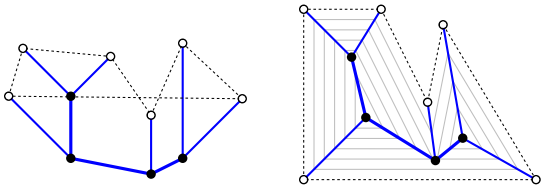


Figure 1: Example of a feasible cycle-free connected abstract geometric graph G (leaves of G are shown as white dots). Left: Arbitrary embedding $E(G)$ and (non-simple) polygon $P_{E(G)}$ (dotted). Right: Suitable polygon $P_{E'(G)}$ for a different embedding $E'(G)$, which is equal to $\mathcal{S}(P_{E'(G)})$. A set of wavefronts of $P_{E'(G)}$ at different points in time are depicted in gray.

Although straight skeletons were introduced to computational geometry in 1995 by Aichholzer et al. [1], their roots actually go back to the 19th century. In textbooks about the construction of roofs (see e.g. [6], pages 86–122) using the angle bisectors (of the polygon defined by the ground walls) was suggested to design roofs where rainwater can run off in a controlled way. This construction is called *Dachausmittlung* and became rather popular. See [5] for related and partially more involved methods to obtain roofs from the ground plan of a house.

Maybe not surprisingly, none of this early works mentions the ambiguity of the non-algorithmic definition of the construction. Using solely bisector graphs does not necessarily lead to a unique roof construction, and actually not even guarantees a plane partition of the interior of the defining boundary [1].

An interesting inverse problem was motivated to us by Lior Pachter and investigations started in [2]: Which graphs are the straight skeleton of some polygon? To give a more formal problem definition we denote with *abstract geometric graphs* the set of combinatorial graphs, where the length of each edge and the cyclic order of incident edges around every vertex is predefined (and cannot be altered). Let $\mathcal{G}$ be the set of cycle-free connected abstract geometric graphs. Denote with $E(G)$ an embedding of $G \in \mathcal{G}$ in the plane, that is, the vertices of G are points in $\mathbb{R}^2$ and the edges of G are straight-line segments of the predefined length, connecting the corresponding points

*Research of O. Aichholzer partially supported by the ESF EUROCORES programme EuroGIGA - ComPoSe, Austrian Science Fund (FWF): I 648-N18. T. Hackl was funded by the Austrian Science Fund (FWF): P23629-N18. S. Huber was funded by the Austrian Science Fund (FWF): L367-N15. H. Cheng, S.L. Devadoss, B. Li, and A. Risteski were funded by NSF grant DMS-0850577.

†Institute for Software Technology, Graz University of Technology, [oaich|thackl]@ist.tugraz.at

‡University of Arizona, Tucson, AZ 85721, howardc@email.arizona.edu

§Williams College, Williamstown, MA 01267, [satyan.devadoss|brian.t.li]@williams.edu

¶Department of Computer Science, Universität Salzburg, Austria, shuber@cosy.sbg.ac.at

‖Princeton University, Princeton, NJ 08544, risteski@princeton.edu

and respecting the predefined cyclic order of incident edges around each vertex. Further, denote with $P_{E(G)}$ the polygon resulting from connecting the leaves of G (with straight-line segments) in cyclic order for the embedding $E(G)$. We call a simple polygon $P_{E(G)}$ *suitable* if its straight skeleton $\mathcal{S}(P_{E(G)}) = E(G)$, for the embedding $E(G)$. If there exists a suitable polygon for a graph $G \in \mathcal{G}$, we call G *feasible*, see Figure 1.

The obvious questions which arise from these definitions are: Which graphs of $\mathcal{G}$ are feasible? Are the suitable polygons for feasible graphs unique? How can one construct a suitable polygon for a feasible graph?

2 Star graphs

All polygon edges whose straight-skeleton faces contain a common vertex u (of the straight skeleton) have equal orthogonal distance t to u, because their wavefront edges reach u at the same time t. That is, the supporting lines of those polygon edges are tangential to the circle with center u and radius t. Thus, in this section we consider a subset of $\mathcal{G}$, the so called star graphs. A *star graph* $S_n \in \mathcal{G}$, for $n \geq 3$ has $(n+1)$ vertices, one vertex u with degree n and n leaves $v_1, \ldots, v_n$ ordered counter clockwise around u. The length of each edge uv_i, with $1 \leq i \leq n$, is denoted by l_i. W.l.o.g. let $l_1 = \max_i l_i$. Observe that the polygon $P_{E(S_n)}$ is star shaped and v_iv_{i+1} (with $v_{n+k} := v_{1+(k-1) \bmod n}$) are its edges.

Observation 1 *If $S_n \in \mathcal{G}$ is a feasible star graph and $P_{E(S_n)}$ is a suitable polygon of S_n, then (1) all straight-skeleton faces are triangles, (2) two consecutive vertices v_i, v_{i+1} can not be both reflex, (3) $l_i < l_{i\pm1}$ for each reflex vertex v_i of $P_{E(S_n)}$, and (4) all edges of $P_{E(S_n)}$ have equal orthogonal distance t to u, with $t \in (0, \min_i l_i]$.*

As a given $S_n \in \mathcal{G}$ is possibly not feasible and a suitable polygon may not be known or does not exist, we define a polyline $L_{S_n}(t, A)$: The vertices $v_1, \ldots, v_{n+1}$ of $L_{S_n}(t, A)$ are the leaves, $v_1, \ldots, v_n$, of S_n, in the same order as for S_n, and one additional vertex v_{n+1} succeeding v_n. The vertices $v_1, \ldots, v_n, v_{n+1}$ have the corresponding distances (predefined in S_n) $l_1, \ldots, l_n, l_1$ to u. A is an assignment for each vertex whether it should be convex or reflex, as seen from u. As $l_1 = \max_i l_i$, v_1 and v_{n+1} are always convex (fact (3) in Observation 1). For the remaining vertices any convex/reflex assignment, which respects the facts (2) and (3) in Observation 1, can be considered. The edges of $L_{S_n}(t, A)$ have equal orthogonal distance t to u. Of course, not all possible combinations of t and an arbitrary embedding $E(S_n)$ allow such a polyline. But it is possible to construct $L_{S_n}(t, A)$ and $E(S_n)$ simultaneously for a fixed $t \in (0, \min_i l_i]$.

For a fixed assignment A and a fixed $t \in (0, \min_i l_i]$ we construct $L_{S_n}(t, A)$ (and $E(S_n)$) in the following way. Consider the circle C with center u and radius t. Start with v_1 at polar coordinate $(l_1, 0)$, with u as origin. For each v_i, $i = 2 \ldots (n+1)$, consider a tangent g_{i-1} to C (such that the vertices will be placed counter clockwise around the circle) through v_{i-1}. If v_{i-1} is convex, then there exist two points with distance l_i (l_1 for v_{n+1}) on g_{i-1}. If v_i is assigned to be reflex, then v_i is placed on the point closer to v_{i-1}, and if v_i is assigned to be convex, then v_i is placed on the other point. If v_{i-1} is reflex, then there exists only one applicable point for placing v_i on g_{i-1}. See Figure 2.

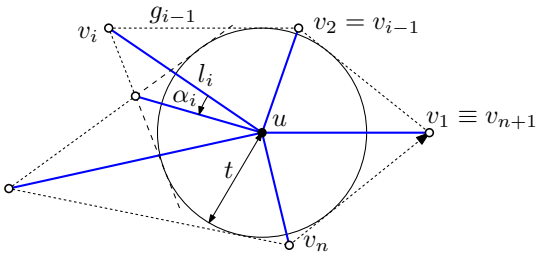


Figure 2: Construction of $L_{S_n}(t, A)$ (and $E(S_n)$) for a given S_n and a fixed distance t and assignment A.

The $L_{S_n}(t, A)$ constructed this way is unique (for fixed t and A), and may be not simple (e.g. when circling C many times), simple but not closed ($v_{n+1} \not\equiv v_1$), or simple and closed ($v_{n+1} \equiv v_1$). In the latter case, the construction reveals a witness pair (t, A) for the existence of some $E(S_n)$, a suitable polygon $P_{E(S_n)}$, and thus the feasibility of S_n.

It is easy to see, that for each suitable polygon $P_{E(S_n)}$, there exists a polyline $L_{S_n}(t, A)$ (just duplicate the vertex v_1). Hence, deciding feasibility of S_n is equivalent to finding an assignment A and a $t \in (0, \min_i l_i]$ such that $L_{S_n}(t, A)$ is closed and simple. For a polyline $L_{S_n}(t, A)$ and a corresponding embedding $E(S_n)$, we denote with α_i, $i = 1 \ldots n$, the counter clockwise angle at u, spanned by uv_i and uv_{i+1}. (Note that for a suitable polygon $P_{E(S_n)}$ α_i can be defined the same way, with $v_{n+1} \equiv v_1$.) It is easy to see that the sum of all α_i is 2π if and only if $L_{S_n}(t, A)$ is closed and simple.

Lemma 1 *Let $S_n \in \mathcal{G}$, distance $t \in (0, \min_i l_i]$ and assignment A be fixed, and let $L_{S_n}(t, A)$ be the resulting polyline. Then $\alpha_A(t) := \sum_{i=1}^{n} \alpha_i$ can be expressed as*

$$\alpha_A(t) = 2 \sum_{\substack{i=1 \\ v_i \text{ convex}}}^{n} \arccos \frac{t}{l_i} - 2 \sum_{\substack{i=1 \\ v_i \text{ reflex}}}^{n} \arccos \frac{t}{l_i} \,. \quad (1)$$

Proof. Omitted in this version. □

For the following result we use the first derivative of α_A:

$$\alpha'_A(t) = 2 \sum_{\substack{i=1 \\ v_i \text{ reflex}}}^{n} \frac{1}{\sqrt{l_i^2 - t^2}} - 2 \sum_{\substack{i=1 \\ v_i \text{ convex}}}^{n} \frac{1}{\sqrt{l_i^2 - t^2}}. \quad (2)$$

Lemma 2 *A suitable convex polygon for a star graph S_n exists if and only if $\sum_i \arccos \frac{\min_i l_i}{l_i} \leq \pi$. If a suitable convex polygon exists then it is unique.*

Proof. As all vertices are assumed to be convex, we obtain $\alpha_A(0) = n\pi > 2\pi$. Furthermore, we observe that $\alpha_A(t)$ is monotonically decreasing since $\alpha'_A(t) < 0$ for all $t \in (0, \min_i l_i]$. Hence, there is a $t \in (0, \min_i l_i]$ with $\alpha(t) = 2\pi$ if and only if $\alpha_A(\min_i l_i) \leq 2\pi$ which is $\sum_i \arccos \frac{\min_i l_i}{l_i} \leq \pi$. If this is the case the solution is unique as $\alpha(t)$ is monotonic. □

For $n = 3$, $\alpha_A(0) = 3\pi$ and $\alpha_A(\min_i l_i) < 2\pi$, and thus we immediately get the following corollary.

Corollary 3 *For every S_3 there exists a unique suitable convex polygon.*

Considering star graphs with $n = 5$, we show in the following lemma that they are not always feasible, and that suitable polygons (if they exist) are not always unique.

Lemma 4 *There exist infeasible star graphs, $S_n \in \mathcal{G}$. Further, there exist feasible star graphs for which multiple suitable polygons exist.*

Proof. To prove the first claim consider a star graph with $n = 5$, $l_1 = l_2 = l_3 = l_4 = 1$, and $l_5 = 0.25$. There exist only two possible assignments: either all vertices convex or all but v_5 convex. It is easy to check that for both assignments $\sum_i \alpha_i > 2\pi$, for every $t \in (0, \min_i l_i]$. To prove the second claim consider a star graph with $n = 5$, $l_1 = l_3 = 1$, $l_2 = 0.6$, $l_4 = 0.79$, and $l_5 = 0.75$. Assign all vertices convex, except for v_2. Then $\sum_i \alpha_i$ evaluates to 2π for $t \approx 0.537$ and $t \approx 0.598$. Hence, there exist (at least) two different suitable polygons for this star graph. □

In the following we discuss sufficient and necessary conditions for the feasibility of a star graph S_4. By Lemma 2 we know in which cases suitable convex polygons exist. The remaining cases are solved by the following lemma.

Lemma 5 *Consider an S_4 for which no suitable convex polygon exists. A suitable non-convex polygon exists if and only if $\frac{1}{\min_i l_i} < \sum_{j=1, l_j \neq \min_i l_i} \frac{1}{l_j}$.*

Proof. First of all, if a polyline has two or more reflex vertices assigned then $\alpha_A(t) < 2\pi$, as each positive summand in Equation (1) is bound by $\pi/2$. Hence, we only need to consider polylines with exactly one reflex vertex, which implies $\alpha_A(0) = 2\pi$.

For simplicity, we may reorder v_i and l_i such that $l_4 = \min_i l_i$. We show that for suitable non-convex polygons v_4 needs to be reflex. Assume to the contrary that some v_k, with $1 \leq k \leq 3$, is reflex. In this case we obtain that $\alpha'_A(t) < 0$ as $1/\sqrt{l_4^2 - t^2}$ dominates $1/\sqrt{l_k^2 - t^2}$ for all $t \in [0, l_4)$. But since $\alpha_A(0) = 2\pi$ we see that $\alpha_A(t) < 2\pi$ for all $t \in (0, \min_i l_i]$.

Observe that the assumption in the lemma, that no suitable convex polygon exists, is equivalent to $\alpha_A(l_4) > 2\pi$. Recall that $\alpha_A(0) = 2\pi$. Hence, if $\alpha'_A(0) < 0$ then there exists a $t \in (0, l_4)$ such that $\alpha_A(t) = 2\pi$, as α_A is continuously differentiable.

Finally, we show that if $\alpha'_A(0) \geq 0$ then $\alpha'_A(t) > 0$ for all $t \in (0, l_4)$. Hence, there is no $t \in (0, l_4]$ such that $\alpha_A(t) = 2\pi$. From Equation (2) we get that $\alpha'_A(t) > 0$ is equivalent to

$$\frac{1}{\sqrt{l_4^2 - t^2}} > \sum_{i=1}^{3} \frac{1}{\sqrt{l_i^2 - t^2}} \quad \Leftrightarrow \quad 1 > \sum_{i=1}^{3} \sqrt{1 - \frac{l_i^2 - l_4^2}{l_i^2 - t^2}}$$

The right side of this equivalence is true since

$$1 \geq \sum_{i=1}^{3} \sqrt{1 - \frac{l_i^2 - l_4^2}{l_i^2}} > \sum_{i=1}^{3} \sqrt{1 - \frac{l_i^2 - l_4^2}{l_i^2 - t^2}}, \qquad (3)$$

where the first inequality is given by $\alpha'_A(0) \geq 0$ and the second inequality holds for all $t \in (0, l_4)$.

To conclude, we have shown that if no suitable convex polygon exists for some S_4, then a suitable non-convex polygon exists for this S_4 if and only if $\alpha'(0) < 0$, which is equivalent to $\frac{1}{\min_i l_i} < \sum_{j=1, l_j \neq \min_i l_i} \frac{1}{l_j}$, as claimed in the lemma. □

3 Caterpillar graphs

The techniques developed in the previous section can be generalized to so-called caterpillar graphs. A *caterpillar graph* $G \in \mathcal{G}$ is a graph that becomes a path if all its leaves (and their incident edges) are removed. We call this path the *backbone* of G. Figure 1 shows a caterpillar graph whose backbone comprises three backbone edges.

In general, a caterpillar graph has m backbone vertices, consecutively denoted by $v_0^1, \ldots, v_0^m$. We denote the adjacent vertices of a backbone vertex v_0^i, with k_i incident edges, by $v_1^i, \ldots, v_{k_i}^i$, such that $v_{k_i}^i = v_0^{i+1}$ for $1 \leq i < m$. Furthermore, we denote by l_j^i the length of the edge $v_0^i v_j^i$, see Figure 3. Let us consider a polygon P whose straight skeleton $\mathcal{S}(P)$ forms a caterpillar graph.

Observation 2 *All edges of P whose straight-skeleton faces contain the same backbone vertex v_0^i have identical orthogonal distance to v_0^i.*

We denote this orthogonal distance by r_i. Hence, the supporting lines of the corresponding polygon edges are tangents to the circle of radius r_i centered at v_0^i, see Figure 3.

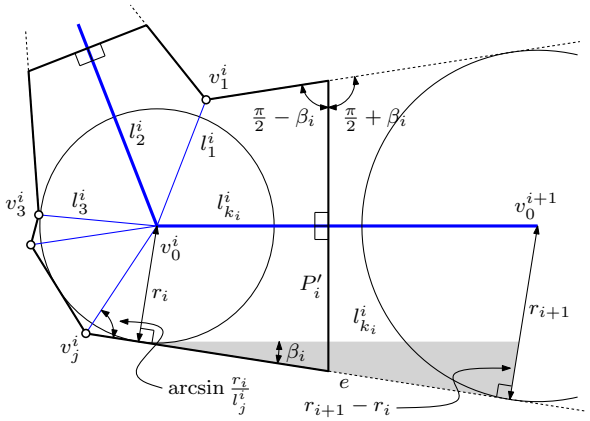


Figure 3: A section of a polygon P for which $\mathcal{S}(P)$ is a caterpillar graph.

Lemma 6 *The radii $r_2,\dots,r_m$ of a suitable polygon $P_{E(G)}$ for some given caterpillar graph G are determined by r_1 and the predefined edge lengths of G according to the following recursions, for $1 \le i < m$:*

$$r_{i+1} = r_i + l_{k_i}^i \sin\beta_i$$
$$\beta_i = \beta_{i-1} + (1 - {}^{k_i}/_2)\pi + \sum_{\substack{j=1 \\ v_j^i \neq v_0^{i-1}}}^{k_i-1} \begin{cases} \arcsin\frac{r_i}{l_j^i} & v_j^i \text{ is convex} \\ \pi - \arcsin\frac{r_i}{l_j^i} & v_j^i \text{ is reflex} \end{cases}$$

For $i = 1$ we define that $\beta_0 = 0$ and $v_j^1 \neq v_0^0$ being true for all $1 \le j < k_1$.

Proof. Denote with e one of the two edges of $P_{E(G)}$ whose faces of $\mathcal{S}(P_{E(G)})$ contain the edge $v_0^i v_0^{i+1}$. The supporting line of e is tangential to the circles at v_0^i and v_0^{i+1}. Considering the shaded right-angled triangle in Figure 3, we obtain $r_{i+1} - r_i = l_{k_i}^i \cdot \sin\beta_i$.

Consider the polygon P_i' (bold in Figure 3) which comprises the edges of $P_{E(G)}$ whose faces of $\mathcal{S}(P_{E(G)})$ contain v_0^i, trimmed by two additional edges orthogonal to $v_0^{i-1}v_0^i$ and $v_0^i v_0^{i+1}$, respectively. P_i' comprises k_i+2 vertices (k_1+1 for P_1') and hence, the sum of inner angles equals $k_i\pi$ ((k_1−1)π for P_1'). On the other hand, we can express this sum as follows (also for P_1'), which implies the second recursion:

$$k_i\pi = 2\pi + 2\beta_{i-1} - 2\beta_i + 2\sum_{\substack{j=1 \\ v_j^i \neq v_0^{i-1}}}^{k_i-1} \begin{cases} \arcsin\frac{r_i}{l_j^i} & v_j^i \text{ is convex} \\ \pi - \arcsin\frac{r_i}{l_j^i} & v_j^i \text{ is reflex} \end{cases}$$

□

Corollary 7 *The sum of the inner angles of $P_{E(G)}$ with convexity assignment A is a function*

$$\alpha_A(r_1) = 2\sum_{j=1}^{n} \begin{cases} \arcsin\frac{r_{v_j}}{l_j} & v_j \text{ is convex} \\ \pi - \arcsin\frac{r_{v_j}}{l_j} & v_j \text{ is reflex} \end{cases}, \quad (4)$$

where r_{v_j} denotes the radius of the circle at the backbone vertex that is adjacent to v_j and l_j denotes the length of the incident edge of G.

The previous corollary provides us with a tool in order to find suitable polygons $P_{E(G)}$ for caterpillar graphs G. We know that for any suitable polygon $P_{E(G)}$ the identity $\alpha_A(r_1) = (n-2)\pi$ must hold. Hence, we can determine all suitable polygons $P_{E(G)}$ as follows: for all 2^n possible assignments A we determine all r_1 such that $\alpha_A(r_1) = (n-2)\pi$.

For any such pair (A, r_1) we construct a polyline $v_1,\dots,v_n,v_{n+1}$ by a similar method as outlined for star graphs: shooting rays tangential to circles centered at the backbone vertices v_0^i. In order to switch over from v_0^i to v_0^{i+1}, we consider the previously constructed ray, which needs to be tangential to the two circles centered at both, v_0^i and v_0^{i+1}, respectively. As the length of the edge $v_0^i v_0^{i+1}$ is given, the center v_0^{i+1} of the next circle is uniquely determined, cf. Figure 3. If there is any non-backbone edge with length $l_j^i < r_i$ then there is no suitable polygon for that particular pair (A, r_1). For each candidate polyline we check whether it is closed, simple and forms a suitable polygon. Note that all suitable polygons can be constructed by the above method.

Lemma 8 *There is at most a finite number of suitable polygons $P_{E(G)}$ for a caterpillar graph G.*

Proof. As α_A is analytic, there are no accumulation points in the set $\{r_1 : \alpha_A(r_1) = (n-2)\pi\}$. Otherwise, α_A would be identical to $(n-2)\pi$. In other words, there is only a finite number of possible pairs (A, r_1) that correspond to a suitable polygon. □

References

[1] O. Aichholzer, D. Alberts, F. Aurenhammer, and B. Gärtner. *A novel type of skeleton for polygons.* Journal of Universal Computer Science, 1(12):752–761, 1995.

[2] H. Cheng, S.L. Devadoss, B. Li, A. Risteski. *Skeletal rigidity of phylogenetic trees.* preprint 2011.

[3] S.L. Devadoss, J. O'Rourke. *Discrete and Computational Geometry.* Princeton University Press, Princeton and Oxford, 2011.

[4] S. Huber. Computing Straight Skeletons and Motorcycle Graphs: Theory and Practice. PhD thesis, University of Salzburg, Austria, 2011.

[5] E. Müller. *Lehrbuch der darstellenden Geometrie für technische Hochschulen.* Band 2, Verlag B.G.Teubner, Leipzig & Berlin, 1916.

[6] G.A.V. Peschka. *Kotirte Ebenen und deren Anwendung.* Verlag Buschak & Irrgang, Brünn, 1877.

Upward planar embedding of a n-vertex oriented path into $O(n^2)$ points

Tamara Mchedlidze*

Abstract

We prove that every n-vertex oriented path admits an upward planar embedding into every general set of $n^2 - n$ points on the plane.

1 Introduction

A *planar straight-line embedding* of a graph G into a point set S is a mapping of each node of G to a distinct point of S and of each edge of G to the straight-line segment between the corresponding end-points so that no two edges cross each other. Planar straight-line embeddings of graphs have been studied since 1991 ([5]) and a large body of literature has been accumulated.

Planar straight-line embeddings have also been studied for directed graphs (*digraphs*, for short). The hierarchy expressed by relations of a digraph can be emphasized by drawing it so that its arcs are represented by curves monotonically increasing in a common direction. Thus, *upward point-set embeddings* were defined. An *upward straight-line point-set embedding* (*UPSE*, for short) of a digraph $G = (V, E)$ into a point-set S ($|S| = |V|$), is a mapping of each vertex of G to a distinct point of S and of each arc to a straight-line segment between its end-points such that no two arcs cross and each arc (u, v) has $y(u) < y(v)$. Upward point-set embeddings recently became an active field of research and various results have already been obtained ([1, 3, 4, 8]). Among them are the following results: (i) Every oriented path admits an UPSE into every convex point set [3], (ii) It is $\mathcal{NP}$-complete to decide whether an upward planar digraph admits an UPSE into a given general point set [4].

In this paper we concentrate on the family of oriented paths and on general point sets. An *oriented path* is a digraph the underlying undirected structure of which is a simple path. Despite the simplicity of the structure of oriented paths, the freedom of embedding that is provided by a general point set makes the problem very complicated. Thus, we only know that the following families of oriented paths always have an UPSE into any general point set:

(i) An oriented path with at most 5 switches[1] and at least two of its monotone paths having length two [1].

(ii) An oriented path P with 3 switches (trivial).

(iii) Every oriented path $P = (v_1, \ldots, v_n)$, so that if its vertex v_i is a sink, then its vertex v_{i+1} is a source [3].

Note that while the above positive results are available, it is still not known whether there exists an oriented path P and a point set S, with $|P| = |S|$, so that P does not admit an UPSE into S.

The variant of the problem when the point set is larger than the oriented path, was considered in [1]. The authors proved that every oriented path P with n vertices and k switches admits an UPSE into every general point set with $n2^{k-2}$ points. Note that this bound is exponential on the number of switches of the given path.

In this paper we reduce the upper bound of $n2^{k-2}$ points, by showing that every oriented path with n vertices admits an UPSE into every set of n^2-n points in general position.

2 Definitions

We say that a point set is *general position*, or is a *general point set*, if no three of its points lie on the same line and no two of its points have the same y-coordinate. The *convex hull* $H(S)$ of a point set S is the point set that can be obtained as a convex combination of the points of S. We say that a point set is in *convex position*, or is a *convex point set*, if none of its point lie in the convex hull of the others. Given a general point set S, we denote by $b(S)$ and by $t(S)$ the lowest and the highest point of S, respectively. In a convex point set S, the subset of points that lie to the left (resp. right) of the line through $b(S)$ and $t(S)$ including $b(S)$ and $t(S)$ is called the *left (resp. right) side of* S. A subset of points of a convex point set S is called *consecutive* if its points appear consecutive as we traverse the convex hull of S. Consider a point set S and its convex hull $H(S)$. Let $L_1 = S \setminus H(S)$, $\ldots$, $L_m = L_{m-1} \setminus H(L_{m-1})$, $L_{m+1} = L_m \setminus H(L_m)$. If m is the smallest integer such that $L_{m+1} = \emptyset$, we say that S *has* m *layers* $L_1, \ldots, L_m$. Two layers L_i and L_{i+1}, $\forall i \in \{1, \ldots, m\}$, are called *consecutive layers* of S. Let p and q be points of L_i and L_{i+1}, respectively. We say that p is *visible* from q (or vice versa) if the

*Faculty of Informatics, Karlsruhe Institute of Technology (KIT), Germany, mched@iti.uka.de

[1] A *switch* is a vertex of a digraph with either no incoming or no outgoing edge.

straight-line segment $\overline{p,q}$ does not cross the polygon defined by the points of L_{i+1}, otherwise we say that p is *invisible* from q (see Figure 1).

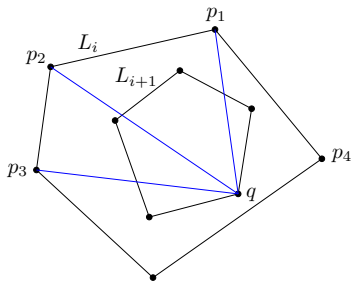


Figure 1: Points p_1, p_2, p_3 are invisible from q, while point p_4 is visible from q.

The graphs we study are directed and the directed edge from a vertex u to a vertex v is denoted by (u, v). A *monotone path* $(v_1, v_2, \ldots, v_k)$ is an oriented path containing edges (v_i, v_{i+1}), $1 \leq i \leq k-1$. A vertex of a digraph with in-degree (resp. out-degree) equal to zero is called a *source* (resp. *sink*). Thus, a vertex of a digraph which is either a source or a sink represents its switch.

If r is a point on the plane, by $y(r)$ we denote its y-coordinate. If v is a vertex of a graph, by $p(v)$ we denote the point to which vertex v has been mapped during the execution of an algorithm. A *free point* of a point set is a point to which no vertex of a graph has been mapped yet.

3 Main result

The following simple proposition represents the key idea for the proof of the main theorem.

Proposition 1 *Let S be a set of $n^2 - n$ points in general position. If S does not contain a convex subset of n points then S has at least n layers.*

The following lemma represents a tool for the proof of the main theorem.

Lemma 1 *Let S be a point set in general position that has m layers $L_1, \ldots, L_m$. Let p_i, q_i be two consecutive points of the right side of the layer L_i and p_{i+1}, q_{i+1} be two consecutive points of the right side of layer L_{i+1}, such that $y(q_i) < y(q_{i+1}) < y(p_{i+1}) < y(p_i)$. At least one of the following statements holds: (a) p_i is visible from q_{i+1}, (b) q_i is visible from p_{i+1}.*

Proof. For the illustration of the statement of the lemma see Figure 2. Assume for the sake of contradiction that both statements (a) and (b) are false, i.e. neither p_i is visible from q_{i+1} nor q_i is visible from p_{i+1}. Rotate line l clockwise through p_{i+1} starting

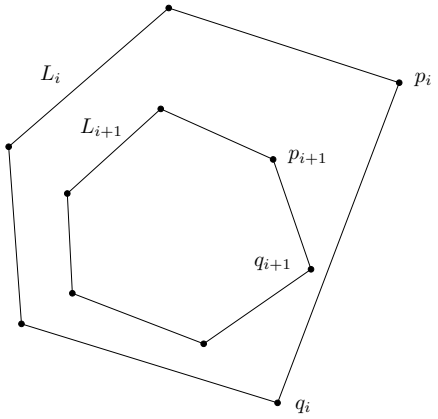


Figure 2: Point p_i is visible from q_{i+1}, but point q_i is invisible from p_{i+1}.

from the horizontal position, see Figure 3. Line l encounters point q_{i+1} before point q_i, since q_i is not visible from p_{i+1}. Similarly, rotate line f counterclockwise though q_{i+1}, starting from the horizontal position. Line f encounters p_i after p_{i+1}. This means that both point q_{i+1} and p_{i+1} are in convex position with points of L_i, a clear contradiction. □

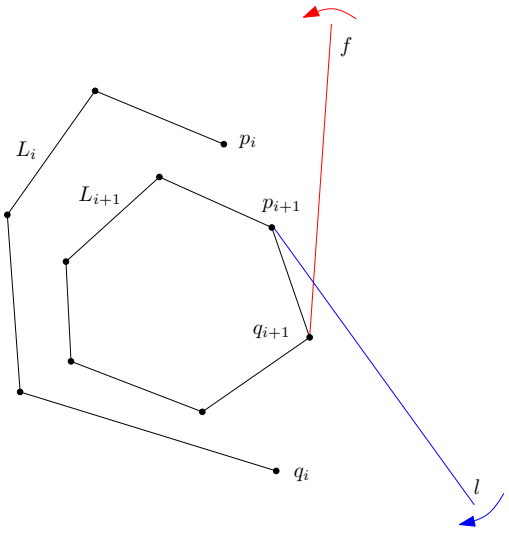


Figure 3: Illustration of the proof of Lemma 1. Points q_{i+1} and p_{i+1} are in convex position with points of L_i.

Theorem 2 *Let S be a set of $n^2 - n$ points in general position and let P be a n-vertex oriented path. P admits an UPSE into S.*

Proof. If S contains n points in convex position, say S', then by Theorem 7 of Binucci *et. al.* [3], P admits an UPSE into S' and thus into S. Assume that none of n points of S are in convex position. Then by Proposition 1, S contains at least n layers. Denote n of these layers by $L_1, L_2, \ldots, L_n$, so that L_1 is the

external layer and L_i, L_{i+1} are two consecutive layers, $i = 1, \ldots, n-1$. Let $V(P) = \{v_1, \ldots, v_n\}$ be the vertex set of P, where the vertices are present in the order they appear in the path P.

We prove the statement of the theorem by induction on the edges of P.

If the first edge of P is (v_1, v_2), we place vertex v_1 on point $t(L_n)$ and vertex v_2 on point $t(L_{n-1})$. Note that $t(L_{n-1})$ is always visible from $t(L_n)$. Case when edge (v_2, v_1) is present in the path P is symmetrical, i.e. we place vertex v_1 on point $b(L_n)$ and vertex v_2 on point $b(L_{n-1})$.

The main concept of the remaining proof is as follows: if vertex v_i is placed on a point of layer L_j, we try to place vertex v_{i+1} to a point of layer L_{j-1}. We only use a point of L_j for vertex v_{i+1}, if it is impossible to place v_{i+1} to L_{j-1} without creating a crossing with the polygon defined by L_j. We also note that only the points of the right sides of the layers are used for the embedding.

Assume that after i steps we have placed vertex v_i on the layer j, $j \leq n-i+1$ and assume also that the following three conditions hold:

($\mathcal{R}$ 1) For any $k < i$, if edge (v_{k-1}, v_k) is present and both vertices v_{k-1}, v_k are placed on the same layer L_f, then all of the points of L_{f-1} which are higher than point $p(v_{k-1})$ are invisible from $p(v_{k-1})$.

($\mathcal{R}$ 2) For any $k < i$, if edge (v_k, v_{k-1}) is present and both vertices v_{k-1}, v_k are placed on the same layer L_f, then all of the points of L_{f-1} which are lower than point $p(v_{k-1})$ are invisible from $p(v_{k-1})$.

($\mathcal{R}$ 3) The drawing of the path induced by the vertices $v_1, \ldots, v_i$ is upward and planar.

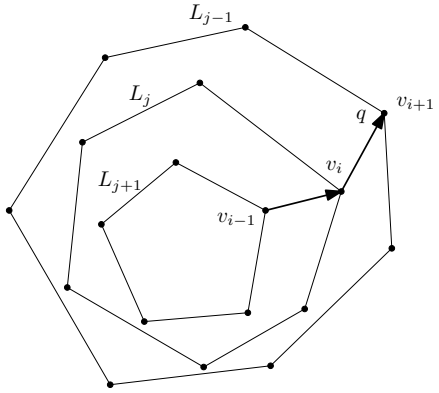


Figure 4: Proof of Theorem 2, Case 1.

Now we place vertex v_{i+1}. Assume that edge (v_i, v_{i+1}) is present, the case when edge (v_{i+1}, v_i) is present in the path P is symmetrical. Recall that vertex v_i was mapped to layer L_j and more specifically, to a point of the right side of layer L_j.

Case 1: *Vertex v_{i-1} was mapped to a point of layer L_{j+1}* (Figure 4 and Figure 5). Recall that vertex v_{i-1} was mapped to the right side of L_{j+1}. Let q be a point of the right side of L_{j-1} which is next higher than point $p(v_i)$. If q is visible from $p(v_i)$, we map there v_{i+1} (Figure 4). If q is invisible from $p(v_i)$, we place v_{i+1} into the point of L_j, that is consecutive higher point of $p(v_i)$ (Figure 5). We know that such a point exists, because otherwise $p(v_i)$ is $t(L_j)$ and then $t(L_{j-1})$ is visible from $t(L_j)$. We also know that this point is free because vertex v_{i-1} was mapped to a point of layer L_{j+1}. Note also that if q is invisible from $p(v_i)$, then all the points of L_{j-1} which are higher than $p(v_i)$ are also invisible from $p(v_i)$. Thus in both cases conditions $\mathcal{R}$ 1, $\mathcal{R}$ 2 hold and the drawing is

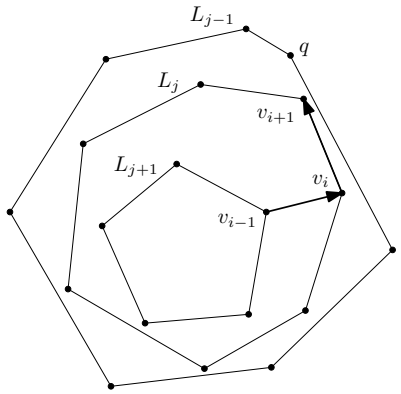


Figure 5: Proof of Theorem 2, Case 1.

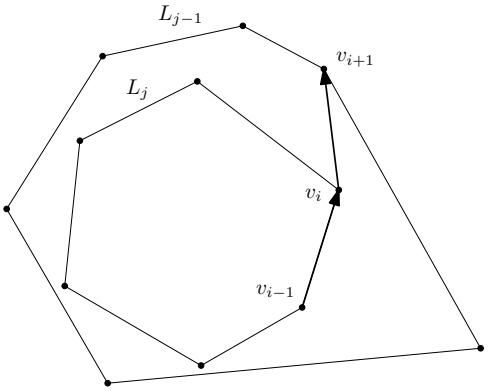


Figure 6: Proof of Theorem 2, Case 2.

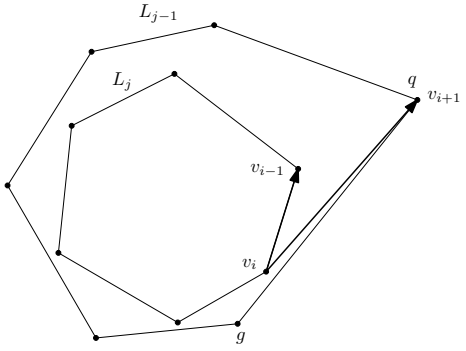


Figure 7: Proof of Theorem 2, Case 2.

upward and planar, since the newly introduced edge can not cross any previously drawn edge (condition $\mathcal{R}$ 3 holds).

Case 2: *Vertex v_{i-1} was mapped to a point of layer L_j* (Figure 6 and Figure 7).

Case 2.a: *Edge (v_{i-1}, v_i) is present*, see Figure 6. Then we proceed exactly as in Case 1.

Case 2.b: *Edge (v_i, v_{i-1}) is present*, see Figure 7. By induction hypothesis and rule $\mathcal{R}$ 2, the point g of L_{j-1} which is next lower than point $p(v_{i-1})$, is invisible from $p(v_{i-1})$. Let q be the consecutive higher than g point of L_{j-1}. Point q is higher than point $p(v_{i-1})$. By Lemma 1, point q is visible from $p(v_i)$. So, we map vertex v_{i+1} to point q. Conditions $\mathcal{R}$ 1 – $\mathcal{R}$ 3 hold. □

4 Conclusion

In this paper we showed that every n-vertex oriented path admits an upward planar embedding into every set of $n^2 - n$ points in general position. This improves the previously known bound of $n2^{k-2}$ points, which is exponential on the number of switches k of the path. Two possible research direction are: (1) to provide a counterexample, proving that not every oriented path admits an UPSE into every general point set of the same size and (2) to reduce the bound of n^2-n points. In the next paragraphs we formulate an equivalent statement of studied problem and present relevant literature which enforces our belief that the bound presented here can be improved.

Let P be an oriented path and S be a general point set, so that $|P| = |S|$. We construct a geometric tournament $T(S)$ from our point-set S as follows. Geometric tournament $T(S)$ contains $|S|$ vertices, which are represented by points of S and $n(n-1)/2$ edges, that connect every pair of its vertices. The direction of the edges is the one which corresponds to the upward orientation. An existence of an UPSE of P into S can be now formulated as the following question: "Does geometric tournament $T(S)$ contain a planar copy of oriented path P?".

The undirected and non-planar versions of this problem are well-studied problems in graph theory. Thus, Hayward [7] showed that every drawing of a complete graph K_n contains at least $c3.2684^n$ crossing-free Hamiltonian circuits, where c is a constant. A non-planar but directed counterpart of our problem is as follows: given a tournament T and an oriented path P, determine whether T contains a copy of P. This problem is known as M.Rosenfeld's conjecture, i.e. M.Rosenfeld conjectured that any n-vertex oriented tournament T contains a copy of any n-vertex oriented path P. This conjecture was finally positively resolved by Thomason [9] for the large values of n. Recently, Havet and Thomasse [6] presented three specific exceptions for $n = 3, 5, 7$, for which the Rosenfeld's conjecture does not hold and proved that it holds in the rest cases. See also [2] for a survey on this problem.

Acknowledgments

I thank Michael Kaufmann and Antonios Symvonis for the useful discussions during the work on this paper.

References

[1] Patrizio Angelini, Fabrizio Frati, Markus Geyer, Michael Kaufmann, Tamara Mchedlidze, and Antonios Symvonis. Upward geometric graph embeddings into point sets. In *18th International Symposium on Graph Drawing (GD '10)*, Lecture Notes in Computer Science, pages 25–37, 2010.

[2] Jørgen Bang-Jensen and Gregory Gutin. Paths, trees and cycles in tournaments. Technical report.

[3] Carla Binucci, Emilio Di Giacomo, Walter Didimo, Alejandro Estrella-Balderrama, Fabrizio Frati, Stephen G. Kobourov, and Giuseppe Liotta. Upward straight-line embeddings of directed graphs into point sets. *Computat. Geom. Th. Appl.*, 43:219–232, 2010.

[4] Markus Geyer, Michael Kaufmann, Tamara Mchedlidze, and Antonios Symvonis. Upward point-set embeddability. In *37th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'11)*, Lecture Notes in Computer Science, pages 272–283. Springer-Verlag, 2011.

[5] Peter Gritzmann, Bojan Mohar, Janos Pach, and Richard Pollack. Embedding a planar triangulation with vertices at specified points. *Amer. Math. Mont.*, 98:165–166, 1991.

[6] Frederic Havet and Stephan Thomasse. Oriented hamiltonian paths in tournaments: A proof of rosenfeld's conjecture. *Journal of Combinatorial Theory, Series B*, 78(2):243 – 273, 2000.

[7] Ryan Hayward. A lower bound for the optimal crossing-free hamiltonian cycle problem. *Discrete and Computational Geometry*, 2:327–343, 1987.

[8] Michael Kaufmann, Tamara Mchedlidze, and Antonios Symvonis. Upward point set embeddability for convex point sets is in P. In *19th International Symposium on Graph Drawing (GD '11)*, Lecture Notes in Computer Science, pages 403–414, 2011.

[9] Andrew Thomason. Paths and cycles in tournaments. *Transactions of the American Mathematical Society*, 296:167–180, 1986.

Aligned Matched Drawability of Some Graph Triples

Maryam Tahmasbi* Zahra Mazaheri†

Abstract

In this paper we extend the concept of matched drawability to aligned matched drawability of triples and quadruples of graphs and develop algorithms for constructing such drawings for quadruples and triples of wheels. We also introduce a new class of graphs called double-rim wheel and develop algorithms for constructing aligned matched drawing for triples consisting of two wheels and a tree or a double-rim wheel.

1 Introduction

The concept of matched drawing of a pair of planar graphs was first introduced by Di Giacomo et al. in [3]. Matched drawings are a relaxation of simultaneous geometric embedding with mapping that was introduced by Brass et al. [1].

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two planar graphs with $|V_1| = |V_2|$ and φ be a one-to-one mapping from V_1 to V_2. A matched drawing of G_1 and G_2 with respect to φ is a pair of straight-line planar drawings Γ_1 and Γ_2 of G_1 and G_2, respectively, such that any pair of matched vertices have the same unique y-coordinate in both drawings. Figure 1 shows a matched drawing of G_1 and G_2.

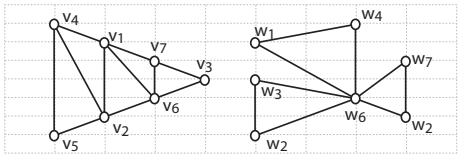


Figure 1: A matched drawing of G_1 and G_2 with one-to-one mapping $\varphi(v_i) = w_i$

If a pair of planar graphs have matched drawings with respect to any given one-to-one mapping, then they are *matched drawable*. Di Giacomo et al. [3] develop algorithms to construct a matched drawing for different classes of graphs and proved that there exist two pairs of planar graphs that are not matched drawable. They also constructed these pairs. Grilli et al. [4] presented a new approach for computing matched drawings of planar graphs. This approach consists of the three following phases: In the *labeling phase* vertices of the two graphs are labeled with "B" or "T" such that matched vertices have the same label. Such a labeling depends on the topology of G_2 and it specifies that vertices labeled as B ("bottom") will be drawn "below" those labeled as T ("top"). In the *leveling phase* a straight-line planar drawing of G_1 is constructed such that the vertices have disjoint non-negative integer y-coordinates and every B-labeled vertex is given an y-coordinate strictly smaller than any T-label vertex. In the *matching phase* a straight-line planar drawing of G_2 is computed such that the y-coordinates of the vertices are those defined in the leveling phase. Based on this approach they provided four algorithms for computing matched drawings of four different graph pairs. They also introduced matched drawings of triples of planar graphs. Let G_1, G_2 and G_3 be three planar graphs with the same number of vertices and for $i, j \in \{1, 2, 3\}$, φ_{ij} be a one to one mapping between vertices of G_i and G_j such that $\varphi_{13} = (\varphi_{12} \circ \varphi_{23})^{-1}$. The three graphs have a matched drawing with respect to the given mappings if there exist three straight-line planar drawings Γ_1, $\Gamma_2$and Γ_3 of G_1 and G_2 and G_3, respectively, such that each pair of drawings is a matched drawing of corresponding pair of graphs with respect to given mapping and the parallel lines of any pairs of drawings do not intersect with the drawing of the remaining graph. Figure 2 shows an example of a matched drawing of three graphs each having five vertices where $\varphi_{12}(v_i) = w_i$ and $\varphi_{23}(w_i) = z_i$ on three sets of lines. They introduced two triples of graphs that always admit a matched drawing [4].

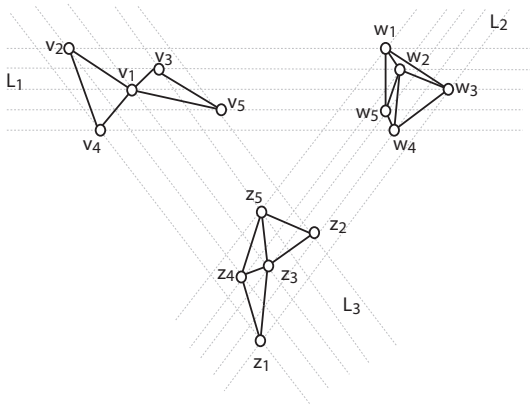


Figure 2: A matched drawing of three graphs.

*Department of Computer Science, Shahid Beheshti University, $m\_tahmasbi$@sbu.ac.ir

†Department of Computer Science, Shahid Beheshti University, zarha67@gmail.com

In this paper we introduce the concept of *aligned matched drawing*. Let G_1, G_2 and G_3 be three planar graphs with three one to one mappings φ_{12}, φ_{13} and φ_{23} defined as before. We can rename the vertices of G_2 and G_3 such that the three functions be identity. In the rest of the paper, for simplicity of notations, we suppose that $V(G_1) = V(G_2) = V(G_3)$ and all mappings are identity. An *aligned matched drawing* of $<G_1, G_2, G_3>$ is a triple $<\Gamma_1, \Gamma_2, \Gamma_3>$ such that Γ_i is a straight line drawing of G_i, $1 \leq i \leq 3$ and every vertex has the same y-coordinate in all drawings. In other words, every triple of matched vertices lie on one of n parallel lines and each line contains exactly three vertices. Figure 3 shows an aligned matched drawing. Aligned matched drawing is a relaxation of the matched drawings of graph triples introduced by Grilli et al. [4]. In algorithms presented for computing matched drawing of triples of graphs, there are restrictions that force both coordinates of one of the three graphs to be fixed, but in aligned matched drawing, we always restrict y-coordinate of vertices.

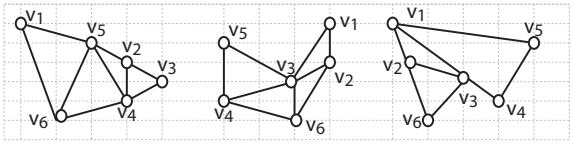


Figure 3: An aligned matched drawing.

A triple of planar graphs is *aligned matched drawable* if it admits an aligned matched drawing with respect to any given one-to-one mapping with mentioned property. We can extend the concept to four graphs in a natural way.

The rest of the paper is organized as follows: in section 2 we show that triples and quadruplets of wheels are aligned matched drawable. In section 3, we extend these results to triples consisting of two wheels and tree or a double-rim wheel. We present conclusions and further works in section 4.

2 Triples and quadruples of wheels

A *wheel* with n vertices, W_n, consists of a cycle $w_1 - w_2 - \ldots - w_{n-1}$ and a vertex w_0, called *center*, that is adjacent to all vertices of the cycle. In the rest of the paper we suppose that wheels are not unlabeled level planar [5]. In the case that they be unlabeled level planar these results follow the paper Di Giacomo et al. [3]. A *y-assignment* of vertices of a graph G with n vertices is a one to one function $\lambda : V(G) \rightarrow \{1, \ldots, n\}$ [3]. A straight-line drawing of a graph with respect to a given y-assignment is a straight line drawing where each vertex v has y-coordinate $\lambda(v)$. Grilli et al.proved that every pair of wheels is matched drawable [4]. They also showed that there always exists a straight-line planar drawing of W_n for any y-assignment λ, if the center of W_n has the smallest y-coordinate. With similar approach we prove the following result.

Theorem 1 *For a wheel W_n and an y-assignment λ, there exists a straight line drawing with respect to λ if $\lambda(w_0) \in \{0,1,n-2,n-1\}$.*

Proof. Let $L = \{l_i : y = i, i = 0, \ldots, n-1\}$ be a set of n lines. Suppose that $\lambda(w_0) = n-2$ and $\lambda(w_1) = n-1$. Now let $w_1, w_2, \ldots, w_{n-1}$ be the counterclockwise order of vertices on the cycle of W_n. Place w_0 on l_0 and w_1 one unit to the left of it on l_1. Now emanate a ray $\overrightarrow{w_0 w_i}$ and rotate it counterclockwise until is crosses $l_{\lambda(w_2)}$ in an integer coordinate. Place w_2 at this point and continue to w_{n-1}. Now you need to draw the edge $w_1 w_{n-1}$. In case that it is not possible to draw it in a planar way, shift w_{n-1} to right until the edge do not cross any other edge of the drawing. Figure 4-(a) shows the drawing. Similar approach can

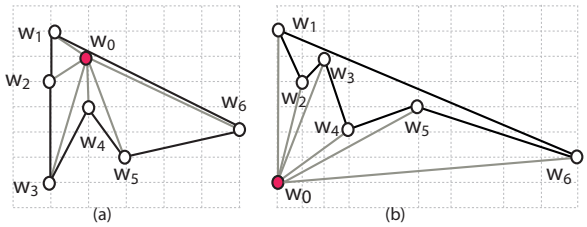


Figure 4: A straight-line planar drawing of W_7

be used when $\lambda(w_0) = 1$, starting with a vertex w_1 with $\lambda(w_1) = 0$. Now suppose that $\lambda(w_0) = 0$ and w_1 be a vertex with $\lambda(w_1) = n - 1$. We repeat the proof of [4]. Let $w_1, w_2, \ldots, w_{n-1}$ be the counterclockwise order of vertices on the cycle of W_n. We place w_0 on l_0 and w_1 one unit to the right of w_0 on l_{n-1}. Then for $i = 1$ we emanate a ray $\overrightarrow{w_0 w_i}$ rotate it clockwise until it crosses horizontal line $y = y(w_{i+1})$ in first an integer x-coordinate, and then increase i one unit. Repeat this until $i = n - 2$. Figure 4-(b) shows this case. Similar approach can be used for the case $\lambda(w_0) = n - 1$. □

In the rest of the paper, we call the algorithm in the proof of theorem 1, *Wheel Draw* algorithm.

Theorem 2 *The pair $<G, W_n>$ is matched drawable for every planar n-vertex graph G.*

Proof. Let w_0 be the center of the wheel. First we compute an embedding of G with w_0 on the external face. We can compute a planar straight-line drawing of G such that all vertices have different y-coordinates and w_0 has the greatest y-coordinate [2]. Then we draw W_n using Wheel Draw algorithm. □

Theorem 2 proves that a planar graph and a wheel are always matched drawable, on the other hand, it has been proven that two wheels are always matched drawable [4]. In the rest of the paper we prove aligned matched drawability of some classes of planar graphs in a triple consisting of two wheels.

Theorem 3 *The triple $<W_n, W'_n, W''_n>$ of wheels is aligned matched drawable.*

Proof. Using theorem 1 it is enough to construct an y-assignment that assigns values in the set $\{0, 1, n-2, n-1\}$ to the centers of the wheels. Then we can construct a straight line drawing of each wheel using Wheel Draw algorithm. □

Corollary 4 *A quadruples $<W_n, W'_n, W''_n, W'''_n>$ of four wheels is aligned matched drawable.*

Proof. With the same approach as theorem 3, it is enough to define an y-assignment that assigns values in the set $\{0, 1, n-2, n-1\}$ to the centers of the wheels. □

3 Aligned matched drawing of other triples

In this section, we describe an algorithm for computing an aligned matched drawing of triple $<wheel, wheel, tree>$. Then we introduce a planar graph called *double-rim wheel* and show that a triple $<wheel, wheel, double-rimwheel>$ is aligned matched drawable.

Theorem 5 *Let T_n be a tree with n vertices and W_n and W'_n be two wheels. Then $<W_n, W'_n, T_n>$ is aligned matched drawable.*

Proof. If W_n and W'_n have the same centers, w_0, we assign $\lambda(w_0) = 0$. Then we construct a planar straight line drawing of T_n where w_0 has y-coordinate 0 [2].

Otherwise suppose that the centers of W_n and W'_n are w_0 and w_1, respectively. For any y-assignment λ where $\lambda(w_0) = 0$ and $\lambda(w_1) = n-1$, W_n and W'_n are matched drawable. Now we compute the y-coordinate of all other vertices as follows: We remove an edge $e = uv$ from T_n to construct two trees T' (containing w_0 and u) and T'' (containing w_1 and v). We then use the BFS algorithm starting from w_0 (and w_1) to label vertices of T' (and T''). Then we construct straight line drawings of T' and T'' such that u has the greatest x-coordinate of vertices of T' and v has smallest x-coordinate of vertices of T''. Figure 5 shows a drawing of T_{13}. □

A double-rim wheel, R_{2n+1} is a planar graph consisting of a vertex v_0, called center, and two cycles $C_1 = v_1 - v_2 - \ldots - v_n$ and $C_2 = u_1 - u_2 - \ldots - u_n$ together with all edges u_iv_i and v_0v_i for $i = 1, \ldots n$.

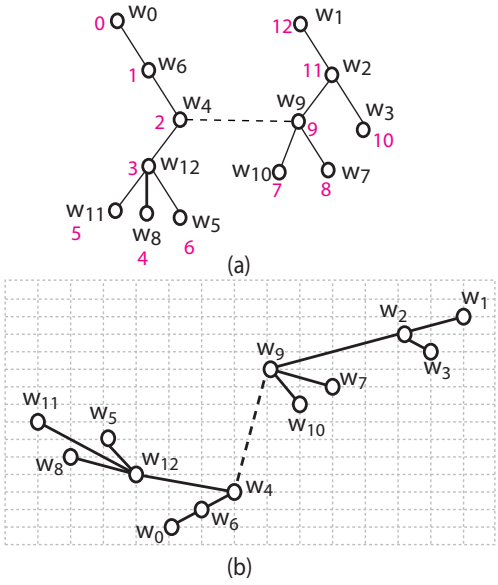


Figure 5: (a)T_{13}, subtrees T' and T'' and y-coordinate of vertices (b) A straight line drawing of T_{13} as in the proof of 5.

The cycle C_1 is called the *inner cycle* and C_2 is called the *outer cycle*. Figure 6-(a) shows a double-rim wheel. In the following theorem we show that a triple consisting of two wheels and a double-rim wheel is aligned matched drawable.

Theorem 6 *The triple $<W_{2n+1}, W'_{2n+1}, R_{2n+1}>$ is aligned matched drawable.*

Proof. First suppose that W_{2n+1}, W'_{2n+1} and R_{2n+1} have the same center, v_0. We start with drawing R_{2n+1}. We place v_0 at (1,1), an arbitrary vertex v_1 at (0,0), any vertex v_i at $(2i, n-i+2)$ and any vertex u_i at $(2i+1, i+n-1)$ except u_1 that is placed at $(2, 2n)$. Figure 6-(b) shows vertex placement. Since v_0 has y-coordinate 1, using Wheel Draw algorithm, we can construct straight line drawings for W_{2n+1} and W'_{2n+1}.

Now suppose that only the centers of the two wheels are the same. If it is on the inner cycle, let v_1 be the center, otherwise, let u_1 be the center. Then construct the drawing of R_{2n+1} as the former case. In the case that the center of one of the wheels are the same with the center of R_{2n+1}, let v_0, we construct the drawing of R_{2n+1} as the former case such that the center of the other wheel is v_1 or u_1.

In the remaining case, suppose that none of the centers are the same. We start with drawing R_{2n+1}. Recall that λ is the y-assignment of the vertices.

1. If the centers of the wheels lie on C_1, let v_1 and v_j be the centers of the wheels. Then we let $\lambda(u_k) = 2n - k$ for $1 \leq k \leq n$ and $\lambda(v_i) = n - i + 1$, for $i \neq 1, j$ and $\lambda(v_1) = 2n$ and $\lambda(v_j) = 1$. For each

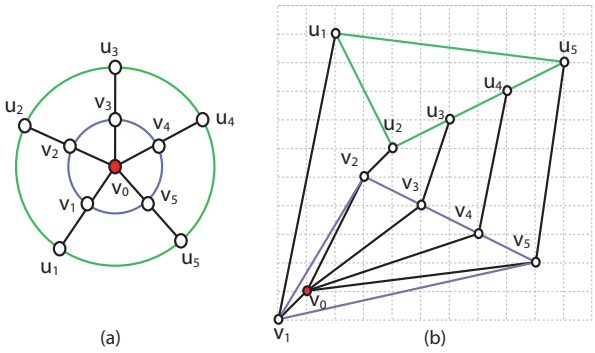


Figure 6: (a) A double-rim wheel, R_{11} (b) A straight-line drawing of R_{11}

v_i, $i = 2, 3, \ldots n$ we assign an arbitrary integer x-coordinate except 1 so that the slope of edges v_0v_i be strictly smaller than the slope of v_0v_{i-1}. We assign the same x-coordinate to u_i and v_i except for u_1 that receives x-coordinate 1. v_0 v_1 have the x-coordinate 0. We can shift v_n and u_n to the right in case any crossing happened between v_1v_n or u_1u_n and the rest of the edges. Figure 7 shows an example.

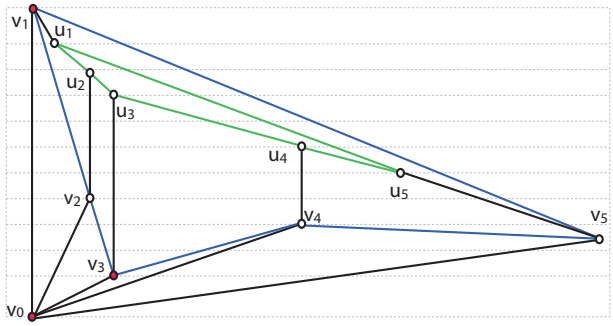


Figure 7: drawing of R_{11} when the centers of W_{11} and W'_{11} lie on C_1 and $j = 3$.

2. If the centers of the wheels lie on C_2, let u_1 and u_j be the centers of the wheels. We place v_0 at $(1, 1)$ any vertex v_i at $(2i, n-i+2)$ for $1 \leq i \leq n$ and any u_k at $(2k+1, k+n+1)$ for $k \neq 1, j$ and u_1 at $(2n, 1)$ and u_j at $(2j+1, 2n-1)$. And in order to avoid crossings, we shift u_n to right.

3. If the centers of the wheels lie on different cycles, we study two cases: if they are adjacent, we label the centers v_1 and u_1 and continue as the case where all the centers are the same. If they are not adjacent we call the centers v_1 and u_j that v_1 has y-coordinate 0 and u_j has y-coordinate $n-1$. Then we draw the double-rim wheel in the same way as case 2. Figure 8 shows an example.

□

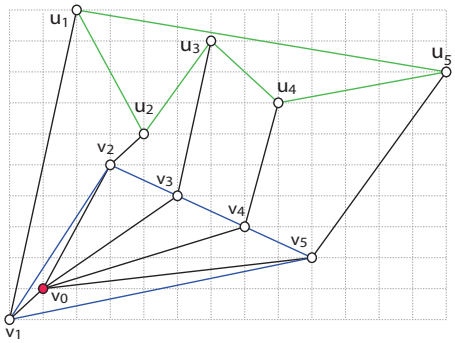


Figure 8: Drawing of R_{11} when the centers of W_{11} and W'_{11} lie on different cycles and $j = 3$.

4 Conclusion

In this paper we defined aligned matched drawing and aligned matched drawability of triples and quadruplets of graphs consisting of two or three wheels. The main tool in this paper is the algorithm Wheel Draw that finds a straight line drawing for W_n if the center has y-coordinate in the set $\{0, 1, n-2, n-1\}$.

A interesting problem in this field is to find triples $<G_1, G_2, G_3>$ such that every pair of them is matched drawable but they are not aligned matched drawable. An open problem is to find new classes of aligned matched drawable graphs.

References

[1] P. Brass, E. Cenek, C. A. Duncan, A. Efrat, C.Erten, D. P. Ismailescu, S. G. Kobourov, A. Lubiw, J.S.B. Mitchell. *On simultaneous planar graph embeddings.* Computational Geometry. Theory and Applications 36 (2), 2007 ,117-130.

[2] G. Di. Battista, P. Eades, R. Tamassia, I.G. Tollis. Graph Drawing, Algorithms for the visualization of graphs. Prentice Hull, 1999.

[3] E. Di Giacomo, W. Didimo, M. van Kreveld, G. Liotta, B. Speckmann. *Matched drawings of planar graphs.* Journal of Graph Algorithms and Applications 13 (3) ,2009, 423-445.

[4] L. Grilli, S. H. Hong, G. Liotta, H. Meijer, S. K. Wismathd. *Matched drawability of graph pairs and of graph triples.* Computational Geometry 43, 2010, 611-634.

[5] J.J. Fowler, S.G. Kobourov. *Characterization of unlabeled level planar graphs.* lecture notes in computer science 4875, 2007, 37-49.

Outerplanar graph drawings with few slopes

Kolja Knauer* Piotr Micek† Bartosz Walczak‡

Abstract

We prove that every outerplanar graph with maximum degree $\Delta \geqslant 4$ has a straight-line outerplanar drawing with at most $\Delta - 1$ distinct edge slopes. This bound is tight: for every $\Delta \geqslant 4$ there is an outerplanar graph of maximum degree Δ which requires at least $\Delta - 1$ distinct edge slopes for its outerplanar straight-line drawing.

1 Introduction

A *straight-line drawing* of a graph G is a mapping of the vertices of G to distinct points in the plane and of the edges of G to straight-line segments connecting the points representing their end-vertices and passing through no other points representing vertices. If it leads to no confusion, in notation and terminology, we make no distinction between a vertex and the corresponding point, and between an edge and the corresponding segment. The *slope* of an edge in a straight-line drawing is the family of all straight lines parallel to the segment representing this edge. The *slope number* of a graph G, introduced by Wade and Chu [8], is the smallest number s such that there is a straight-line drawing of G using s slopes.

Since at most two edges at each vertex can use the same slope, $\lceil \frac{\Delta}{2} \rceil$ is a lower bound for the slope number of a graph with maximum degree Δ. In general, graphs with maximum degree $\Delta \geqslant 5$ may have arbitrarily large slope number, see [1, 7]. If the maximum degree of a graph is at most 3 then the slope number is at most 4 as shown by Mukkamala and Szegedy [6], improving a result of Keszegh et al. [4]. The question whether the slope number of graphs with maximum degree 4 is bounded by a constant remains open.

The situation is different for *planar* straight-line drawings. It is well known that every planar graph admits a planar straight-line drawing [5]. The *planar slope number* of a planar graph G is the smallest number s such that there is a planar straight-line drawing of G using s slopes. In [3] Keszegh et al. show that the planar slope number is bounded by a function of maximum degree. Their bound is exponential and their proof is non-constructive. Jelínek et al. [2] give an upper bound for the planar slope number of planar graphs of treewidth at most 3, which is $O(\Delta^5)$.

In the present paper we consider drawings of outerplanar graphs. As outerplanar graphs have treewidth at most 2, they admit a planar drawings with $O(\Delta^5)$ slopes. A straight-line drawing of a graph G is *outerplanar* if it is planar and all vertices of G lie on the outer face. The *outerplanar slope number* of an outerplanar graph G is the smallest number s such that there is an outerplanar straight-line drawing of G using s slopes. We provide a tight bound for the outerplanar slope number in terms of the maximum degree.

Theorem 1 *The outerplanar slope number of every outerplanar graph with maximum degree $\Delta \geqslant 4$ is at most $\Delta - 1$.*

This result is sharp, as witnessed by a long cycle where each vertex is made adjacent to $\Delta - 2$ additional independent vertices. The tight bounds for the outerplanar slope number with respect to the maximum degree Δ are therefore: 1 for $\Delta = 1$, 3 for $\Delta \in \{2, 3\}$, and $\Delta - 1$ for $\Delta \geqslant 4$.

The proofs of Theorem 1 are somewhat different for $\Delta = 4$ and $\Delta \geqslant 5$. In the following we sketch the ideas of the proof for $\Delta \geqslant 5$.

2 Bubbles

Suppose we are given an outerplanar drawing of a connected graph G with maximum degree $\Delta \geqslant 5$. This drawing determines the cyclic ordering of edges around every vertex. We produce an outerplanar straight-line drawing of G with few edge slopes which preserves this ordering at every vertex. Our construction is inductive: it composes the entire drawing of G from drawings of subgraphs of G that we call bubbles.

We distinguish the *outer face* of G (the one that is unbounded in the given drawing of G and contains all vertices on the boundary) from the *inner faces*. The edges on the boundary of the former are *outer edges*, while all remaining ones are *inner edges*. A *snip* is a simple closed counterclockwise-oriented curve γ which

*Technical University Berlin, knauer@math.tu-berlin.de, Supported by DFG grant FE-340/8-1 as part of ESF project GraDR EUROGIGA.

†Jagiellonian University, piotr.micek@tcs.uj.edu.pl, Supported by Ministry of Science and Higher Education of Poland as part of ESF project GraDR EUROGIGA.

‡Jagiellonian University, walczak@tcs.uj.edu.pl, Supported by Ministry of Science and Higher Education of Poland as part of ESF project GraDR EUROGIGA.

- passes through some pair of vertices u and v of G (possibly being the same vertex) and through no other vertex of G,
- on the way from v to u goes entirely through the outer face of G,
- on the way from u to v (considered only if $u \neq v$) goes through inner faces of G possibly crossing some inner edges of G, each at most once.

Every snip γ defines a *bubble* H in G as the subgraph of G induced on the vertices lying on or inside γ. Note that H is a connected subgraph of G as γ crosses no outer edges. The oriented simple path P from u to v in H going counterclockwise along the boundary of the outer face of H is called the *root-path* of H. If $u = v$ then the root-path consists of that single vertex only. The *roots* of H are the vertices u and v together with all vertices of H incident to the edges crossed by γ. Note that vertices of H not being roots cannot have edges to $G - H$. Note also that the root-path and the roots of H do not depend on the particular snip γ used to define H. The order of roots along the root-path gives the *root-sequence* of H. A bubble with k roots is called a *k-bubble*. A special role in our proof is played by 1- and 2-bubbles.

Bubbles admit a natural decomposition, which is the base of our recursive drawing.

Lemma 2 *Let H be a bubble with root-path $P = v_1 \dots v_k$. Every component of $H - \{v_1, \dots, v_k\}$ is adjacent to either one vertex among $v_1, \dots, v_k$ or two consecutive vertices from $v_1, \dots, v_k$. Moreover, there is at most one component adjacent to v_i and v_{i+1} for all $1 \leqslant i < k$.*

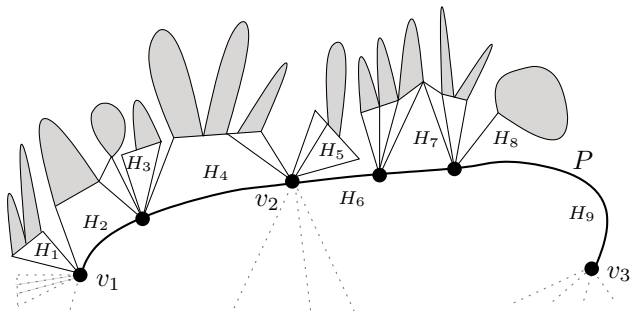


Figure 1: A 3-bubble H with root path P (drawn thick), root-sequence (v_1, v_2, v_3) (connected to the remaining graph by dotted edges), and splitting sequence into v- and e-bubbles $(H_1, \dots, H_9)$. For example, (H_2, H_3, H_4) is a 2-bubble.

Lemma 2 allows us to assign each component of $H - \{v_1, \dots, v_k\}$ to a vertex of P or an edge of P so that every edge is assigned at most one component. For a component C assigned to a vertex v_i, the graph induced on $C \cup \{v_i\}$ is called a *v-bubble*. If P consists of a single vertex with no component assigned to it, we consider that vertex alone to be a v-bubble. For a component C assigned to an edge $v_i v_{i+1}$, the graph induced on $C \cup \{v_i, v_{i+1}\}$ is called an *e-bubble*. If no component is assigned to an edge of P then we consider that edge alone an e-bubble. All v-bubbles of v_i in H are naturally ordered by their clockwise arrangement around v_i in the drawing. All this leads to a decomposition of the bubble H into a sequence $(H_1, \dots, H_b)$ of v- and e-bubbles such that the naturally ordered v-bubbles of v_1 precede the e-bubble of $v_1 v_2$, which precedes the naturally ordered v-bubbles of v_2, and so on. We call this sequence the *splitting sequence* of H and write $H = (H_1, \dots, H_b)$. Note that v- and e-bubbles are special kinds of 1- and 2-bubbles respectively. Every 1-bubble is a bouquet of v-bubbles. The splitting sequence of a 2-bubble may consist of several v- and e-bubbles. For an illustration see Figure 1.

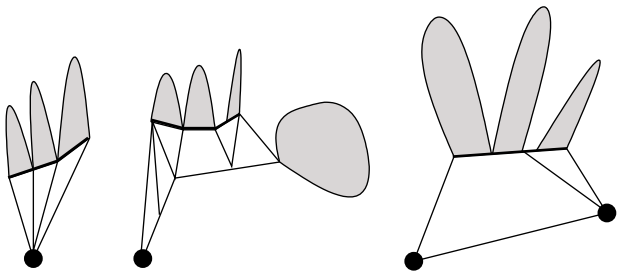


Figure 2: Three ways of obtaining smaller bubbles from v- and e-bubbles. The new root-path is drawn thick.

The general structure of the induction in our proof is covered by the following (see Figure 2):

Lemma 3

3.1. Let H be a v-bubble rooted at v and let $v_1, \dots, v_k$ be the neighbors of v in H in clockwise order. Then $H - v$ is a k-bubble with root-sequence $v_1, \dots, v_k$.

3.2. Let H be a v-bubble rooted at v. Let P be an induced path going from v counterclockwise along the outer face of H to a vertex w of H, such that the internal vertices of P are not cut-vertices of H. Let H' be the component of $H - P$ adjacent to both v and w. Then H' is a bubble with roots being the neighbors of P in H'.

3.3. Let H be an e-bubble with root-edge uv. Suppose that $u_1, \dots, u_k, v$ are the neighbors of u in H in clockwise order and $u, v_1, \dots, v_\ell$ are the neighbors of v in H in clockwise order. Then $H -$

$\{u, v\}$ is a bubble with root-sequence $u_1, \ldots, u_k$, $v_1, \ldots, v_\ell$, where u_k and v_1 may coincide.

3 Bounding regions

Depending on the maximum degree Δ of G we define the set S of $\Delta - 1$ slopes to consist of the horizontal slope and the slopes of vectors $\mathbf{f}_1, \ldots, \mathbf{f}_{\Delta-2}$ where

$$\mathbf{f}_i = (-\tfrac{1}{2} + \tfrac{i-1}{\Delta-3}, 1) \quad \text{for } i = 1, \ldots, \Delta - 2.$$

An important property of S is that it cuts the horizontal segment from $(-\frac{1}{2}, 1)$ to $(\frac{1}{2}, 1)$ into $\Delta-3$ segments of equal length $\frac{1}{\Delta-3}$. We construct an outerplanar straight-line drawing of G using only slopes from S and preserving the given cyclic ordering of edges at each vertex of G.

The essential tool in proving that our construction does not make bubbles overlap are bounding regions. Their role is to bound the regions occupied by bubbles. The bounding region for a bubble is parametrized by ℓ and r which depend on the degrees of the roots in the bubble. Let v be a point in the plane. For a vector x let $R(v; x) = \{v + \alpha x : \alpha \geqslant 0\}$. We define $LB(v; \ell)$ to be the set consisting of v and of all points p such that $p_y \geqslant v_y$. If $\ell > 0$ then we furthermore require that p lies

- to the right of $R(v; \mathbf{f}_{\Delta-2})$ if $\ell = \Delta - 1$,
- to the right of $R(v; \mathbf{f}_\ell + \frac{1}{\Delta-4}\mathbf{f}_1)$ if $2 \leqslant \ell \leqslant \Delta - 2$,
- on or to right of $R(v; \mathbf{f}_1)$ if $\ell = 1$.

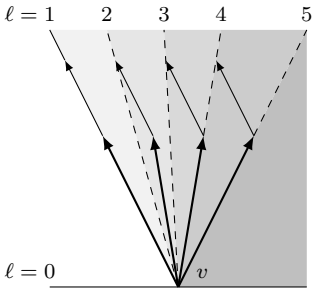


Figure 3: Boundaries of $LB(v; \ell)$ for $\Delta = 6$. Vectors $\mathbf{f}_i$ at v are indicated by thick arrows. Vectors $\frac{1}{2}\mathbf{f}_1$ at $v + \mathbf{f}_i$ are indicated by thin arrows. Note that $\mathbf{f}_3$ lies on the boundary of $LB(v; 4)$.

See Figure 3 for an illustration. Similarly, $RB(v; r)$ consists of v and of all points p such that $p_y \geqslant v_y$. If $r < \Delta - 1$ we furthermore require that p lies

- to the left of $R(v; \mathbf{f}_1)$ if $r = 0$,
- to the left of $R(v; \mathbf{f}_r + \frac{1}{\Delta-4}\mathbf{f}_{\Delta-2})$ if $1 \leqslant r \leqslant \Delta - 3$,
- on or to the left of $R(v; \mathbf{f}_{\Delta-2})$ if $r = \Delta - 2$.

Now, for points u, v in the plane such that $u_y = v_y$ and $u_x \leqslant v_x$ we define bounding regions as follows: if $0 \leqslant \ell, r \leqslant \Delta-1$ then $B(uv; \ell, r) = LB(u; \ell) \cap RB(v; r)$ and if additionally a height $h > 0$ is specified then $\bar{B}(uv; \ell, r; h) = B(uv; \ell, r) \cap \{p : p_y < u_y + h\}$.

We denote $B(vv; \ell, r)$ simply by $B(v; \ell, r)$ and $\bar{B}(vv; \ell, r; h)$ simply by $\bar{B}(v; \ell, r; h)$. We use $B(v; \ell, r)$ and $\bar{B}(v; \ell, r; h)$ to bound drawings of 1-bubbles H with root v such that $r - \ell + 1 = d_H(v)$. Note that every 1-bubble drawn inside $B(v; \ell, r)$ may be scaled to fit inside $\bar{B}(v; \ell, r; h)$ for any $h > 0$ without changing slopes. We use $\bar{B}(uv; \ell, r; h)$ with $u \neq v$ to bound drawings of 2-bubbles H whose root-path starts at u and ends at v, such that $\ell = \Delta - d_H(u)$ and $r = d_H(v) - 1$. Here H cannot be scaled as the positions of two of its vertices are fixed, so the value of h matters. We also use $\bar{B}(uv; 1, \Delta - 2; h)$ with $u \neq v$ to bound drawings of bubbles with any number of roots and with root-path starting at u and ending at v.

4 The drawing

The following lemma does the main job in the proof of Theorem 1 for $\Delta \geqslant 5$.

Lemma 4 *Suppose $\Delta \geqslant 5$.*

4.1. Let H be a 1-bubble with root v. Suppose that the position of v is fixed. Let ℓ and r be such that $0 \leqslant \ell, r \leqslant \Delta - 1$ and $r - \ell + 1 = d_H(v)$. Then there is a straight-line drawing of H inside $B(v; \ell, r)$.

4.2. Let H be a 2-bubble with first root u and last root v. Suppose that the positions of u and v are fixed on a horizontal line so that u lies to the left of v. Let $\ell = \Delta - d_H(u)$ and $r = d_H(v) - 1$. Then there is a straight-line drawing of H inside $\bar{B}(uv; \ell, r; \frac{\Delta-3}{\Delta-4}|uv|)$ such that the root-path of H is drawn as the segment uv.

4.3. Let H be a k-bubble with roots $v_1, \ldots, v_k$ in this order along the root-path. If $k = 1$ then suppose $d_H(v_1) \leqslant \Delta - 2$, otherwise suppose $d_H(v_1), d_H(v_k) \leqslant \Delta - 1$. Suppose that for some $\lambda > 0$ the positions of $v_1, \ldots, v_k$ are fixed in this order on a horizontal line so that $|v_1v_2| = \ldots = |v_{k-1}v_k| = \lambda$. Then there is a straight-line drawing of H inside $\bar{B}(v_1v_k; 1, \Delta-2; \frac{\Delta-3}{\Delta-4}\lambda)$ such that the root-path of H is drawn as the segment v_1v_k.

The drawings claimed above use only slopes from S and preserve the order of edges around each vertex w of H under the assumption that if there are edges connecting w to $G - H$ then they are drawn in the correct order outside the considered bounding region.

Proof. The proof of all three statements goes by one induction on the size of the bubble H. In the following

we only show the induction step for 4.2 in a special case where the root-path of H consists of the single edge uv. This already uses a big part of the ideas for the full proof.

In the considered case the splitting sequence of H consists of some v-bubbles $X_1, \ldots, X_p$ rooted at u, followed by an e-bubble Y, followed by some v-bubbles $Z_1, \ldots, Z_q$ rooted at v. We start by drawing Y. Define $\ell' = \Delta - d_Y(u)$ and $r' = d_Y(v) - 1$. Suppose that Y is not the single edge uv as otherwise there is nothing more from Y to draw. Let $u_{\ell'}, \ldots, u_{\Delta-2}, v$ be the neighbors of u in Y in the clockwise order. Let $u, v_1, \ldots, v_{r'}$ be the neighbors of v in Y in the clockwise order. It follows from 3.3 that $Y - \{u, v\}$ is a bubble with roots $u_{\ell'}, \ldots, u_{\Delta-2}, v_1, \ldots, v_{r'}$, where $u_{\Delta-2}$ and v_1 may coincide. Define

$$\alpha = \begin{cases} |uv| & \text{if } u_{\Delta-2} = v_1, \\ \frac{\Delta-3}{\Delta-2}|uv| & \text{if } u_{\Delta-2} \neq v_1. \end{cases}$$

Put each vertex u_i at point $u + \alpha \mathbf{f}_i$ and each vertex v_i at point $v + \alpha \mathbf{f}_i$. Note that if $u_{\Delta-2}$ and v_1 coincide then they are correctly put to the same point. The points u_i and v_i split the horizontal segment $u_{\ell'}v_{r'}$ into segments of length $\frac{\alpha}{\Delta-3}$. Induction hypothesis 4.3 applied to the bubble $Y - \{u, v\}$ yields its drawing inside $\bar{B}(u_{\ell'}v_{r'}; 1, \Delta - 2; \frac{\alpha}{\Delta-4})$ having all additional properties stated in 4.3. It follows easily from the definition of bounding regions that

$$\bar{B}(u_{\ell'}v_{r'}; 1, \Delta - 2; \tfrac{\alpha}{\Delta-4}) \subset \bar{B}(uv; \ell, r; \tfrac{\Delta-3}{\Delta-4}|uv|).$$

This already completes the proof if $H = Y$. Now, suppose there are some v-bubbles $X_1, \ldots, X_p$ starting the splitting sequence of H. By induction hypothesis 4.1 the 1-bubble $X = (X_1, \ldots, X_p)$ can be drawn properly inside $B(u; \ell, \ell' - 1)$. We scale this drawing so that it fits inside $\bar{B}(u; \ell, \ell' - 1; \alpha)$. The latter bounding region is contained in $\bar{B}(uv; \ell, r; \frac{\Delta-3}{\Delta-4}|uv|)$. Moreover, it lies entirely below the horizontal line going through the points u_i and v_i, and entirely to the left of the edge $uu_{\ell'}$. Therefore, it does not overlap with the drawing of Y. The 1-bubble $Z = (Z_1, \ldots, Z_q)$ is drawn similarly on the other side. The resulting drawing of H clearly fulfills all the requirements. If Y is the single edge uv then we draw X and Z as above choosing $\alpha = |uv|$, which makes their bounding regions disjoint. □

To prove Theorem 1 for $\Delta \geqslant 5$, fix the position of any vertex v of G and apply 4.1 to the graph G considered as a 1-bubble with root v.

5 Further comments

The set of slopes used to prove Theorem 1 is rather special. A natural question arises: Depending on Δ, what is the smallest number s, such that any set S of s slopes allows for an outerplanar drawing using only slopes from S? We can only provide the following upper bound.

Theorem 5 *Every set S of $2\Delta - 4$ slopes may be used to draw any outerplanar graph G with maximum degree Δ using only slopes from S.*

Another problem we would like to mention is the following: Are there a function f and a polynomial p such that every outerplanar graph with maximum degree Δ and n vertices admits an outerplanar straight-line drawing on integer coordinates inside a $p(n)\times p(n)$ grid while using at most $f(\Delta)$ slopes?

References

[1] J. Barát, J. Matoušek, and D. Wood, *Bounded-degree graphs have arbitrarily large geometric thickness*, Electron. J. Combin. **13** (2006), #R3, 14 pp.

[2] V. Jelínek, E. Jelínková, J. Kratochvíl, B. Lidický, M. Tesař, and T. Vyskočil, *The planar slope number of planar partial 3-trees of bounded degree*, Graph drawing, Lecture Notes in Comput. Sci., vol. 5849, 2010, pp. 304–315.

[3] B. Keszegh, J. Pach, and D. Pálvölgyi, *Drawing planar graphs of bounded degree with few slopes*, Graph drawing, Lecture Notes in Comput. Sci., vol. 6502, 2011, pp. 293–304.

[4] B. Keszegh, J. Pach, D. Pálvölgyi, and G. Tóth, *Drawing cubic graphs with at most five slopes*, Graph drawing, Lecture Notes in Comput. Sci., vol. 4372, 2007, pp. 114–125.

[5] P. Koebe, *Kontaktprobleme der konformen Abbildung*, Berichte Verhande. Sächs. Akad. Wiss. Leipzig, Math.-Phys. Klasse **88** (1936), 141–164.

[6] P. Mukkamala and M. Szegedy, *Geometric representation of cubic graphs with four directions*, Comput. Geom. **42** (2009), 842–851.

[7] J. Pach and D. Pálvölgyi, *Bounded-degree graphs can have arbitrarily large slope numbers*, Electron. J. Combin. **13** (2006), #N1, 4 pp.

[8] G. Wade and J. Chu, *Drawability of complete graphs using a minimal slope set*, The Computer Journal **37** (1994), 139–142.

The graphs that can be drawn orthogonally with one bend per edge

Stefan Felsner
Institut für Mathematik,
Technische Universität Berlin, Germany.

Michael Kaufmann
Wilhelm-Schickard-Institut für Informatik,
Universität Tübingen, Germany.

Pavel Valtr
Department of Applied Mathematics and Institute for Theoretical Comp. Sci. (ITI),
Charles University, Praha, Czech Republic.

Abstract

We provide a precise description of the class of graphs that admit a one-bend drawing, i.e., an orthogonal drawing with at most one bend per edge. The main tools for the proof are Eulerian orientations of graphs and discrete harmonic functions.

1 Introduction

The term *d-dimensional orthogonal drawing* traditionally denotes a drawing of a graph in which vertices are placed at distinct points of the d-dimensional integer lattice and edges are represented by chains of axis-parallel segments. Orthogonal drawings and variations are classical topics in graph drawing. The discrete nature of the model makes orthogonal drawings accessible for tools from combinatorial optimization. Orthogonal drawings are also related to various applications ranging from circuit layout to information visualization.

Planar graphs with $\Delta \leq 4$ admit crossing-free 2-dimensional orthogonal drawings. Tamassia's seminal paper [17] is about the minimization of bends of such a drawing. For non-planar graphs, authors mainly worked on area minimization allowing a constant number of bends per edge [12]. Generalizations have been made in various directions, e.g. for higher degree graphs by representing vertices as boxes [1, 2, 7], incremental drawings [11, 6] and simple faces [8, 14, 13]. On 3-dimensional orthogonal drawings, there is a less extensive literature, most prominent are [4, 3].

Here we are interested in 2-dimensional orthogonal drawings of graphs with maximal degree $\Delta \leq 4$ and vertices represented as points. We only deal with 2-dimensional orthogonal drawings. For simplicity we refer to them as *orthogonal drawings*.

Every graph with $\Delta \leq 4$ has an orthogonal drawing. A *good* drawing, however, should be compact and readable. Therefore drawing algorithms are usually compared with respect to the drawing area and the number of bends. Optimizing either of these two parameters is NP-hard. This was shown for the area in [9, 5] and for the bend number in [16]. Several constructions of orthogonal drawings have been proposed e.g. by Schäffter [15] or by Eades, et al. [4]. From work of Biedl and Kant [1] and Papakostas and Tollis [12] it follows that graphs with n vertices and $\Delta \leq 4$ admit orthogonal drawings with area $0.76n^2$, at most two bends per edge and a total of at most $2n + 2$ bends.

We investigate which graphs admit an orthogonal drawing with at most one bend per edge, for the sake of brevity we call such drawings *one-bend drawings*. The paper by Biedl and Kaufmann [2] provides a general framework for such one-bend drawings but it focuses on the more general case of higher degrees where the vertices are represented by boxes. In [18], their algorithm has been analyzed more closely and extended by several heuristics. Here we focus on the case of vertices represented as points, which restricts the graphs to be of maximal degree 4. Theorem 4 gives a full characterization of maximal graphs admitting a one-bend drawing. The construction of the embedding is based on discrete harmonic functions, a tool that has not been used before in this area. Theorem 6 extends the characterization so that it includes non-maximal graphs.

Let $G = (V, E)$ be a graph with maximum degree $\Delta = 4$. Suppose that there is an orthogonal drawing of G such that every edge has at most one bend. If v is a vertex and edge (v, w) is leaving v towards the north, i.e., constant x-coordinate and increasing y-coordinate, then w is further to the north than v, i.e., $v_y < w_y$. Corresponding statements hold for edges leaving a vertex towards the other directions.

Let $S \subset V$ be a set of vertices. Consider the subgraph $G[S] = (V, E[S])$ of G induced by this set and its induced one-bend drawing. From the above we see that in $G[S]$ the northernmost vertex of S has no edge towards the north. With the respective statements for the south, east and west extreme vertices we obtain that the sum of degrees of vertices in $G[S]$ can be at most $4|S| - 4$. This yields the following necessary condition for the existence of one-bend drawings:

Partially supported by DFG grant FE-340/7-2 and ESF EuroGIGA project GraDR.

Proposition 1 *If $G = (V, E)$ has a one-bend drawing and $S \subseteq V$, then $|E[S]| \leq 2|S| - 2$.*

The assertion of Theorem 6 will be that the degree condition $\Delta \leq 4$ together with the condition on edge densities given in Proposition 1 yields a sufficient set of conditions for the existence of a one-bend drawing.

2 The four-regular case

Let $G = (V, E)$ be a connected four-regular graph. We even allow multiple edges but Proposition 1 implies that in interesting cases the multiplicity of edges is at most two.

From Proposition 1 we know that there is no one-bend drawing for G. However if we specify any vertex v_∞ let $G' = G[V \setminus v_\infty]$, then G' may have a one-bend drawing. Such a drawing of G' can also be interpreted as a one-bend drawing of G with v_∞ placed at ∞. We refer to such a drawing of G as an *extended one-bend drawing*. Figure 1 shows an example.

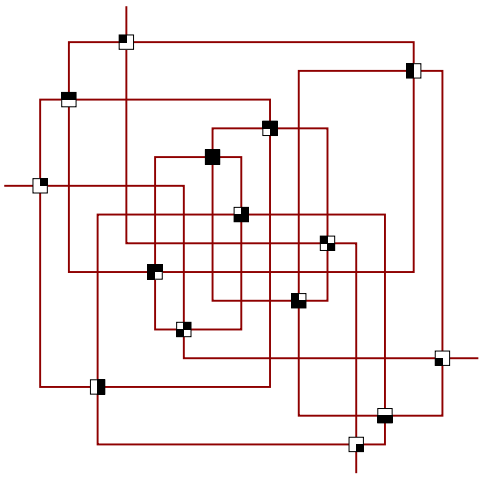


Figure 1: An extended one-bend representation of the four dimensional cube.

We can use an extended one-bend drawing of G that has the extra property that every edge has exactly one bend to define an orientation D_G of G according to the rule:

- edge vw is oriented as $v \to w$ iff in the drawing the edge is leaving v horizontally, i.e., to the east or to the west.

Every edge has a unique orientation in D_G because by assumption every edge has exactly one bend in the drawing. Moreover, the same orientation can be defined in terms of vertical segments as: edge vw is oriented as $v \to w$ iff in the drawing the edge is leaving w vertically. Together the rules imply a unique orientation for the edges incident to v_∞.

The orientation D_G has the property that $\mathsf{in\text{-}deg}(v) = 2$ and $\mathsf{out\text{-}deg}(v) = 2$ for every vertex v of G, hence it is an Eulerian orientation.

Our construction of extended one-bend drawings starts with an Eulerian orientation O_G of a four-regular graph G with a special vertex v_∞. We then aim at constructing a one-bend drawing such that the orientation D_G defined on G according to the above rule equals the Eulerian orientation O_G.

Given an Eulerian orientation O_G we identify

- v_L and v_R, these are the two vertices with edges $v_L \to v_\infty$ and $v_R \to v_\infty$, the indices L and R are assigned arbitrarily. We call v_L and v_R the *horizontal poles*.
- For $v \in V \setminus \{v_L, v_R, v_\infty\}$ we let B_v be the set of vertices w such that in O_G there is an edge $v \to w$. Clearly $|B_v| = 2$ and in a one-bend drawing corresponding to O_G vertex v has to be horizontally between the elements of B_v, i.e., if $B_v = \{w', w''\}$ the either $w'_x < v_x < w''_x$ or $w'_x > v_x > w''_x$. This is the *horizontal betweenness condition* for v.

The Eulerian orientation also provides two vertical poles v_T and v_B and a vertical betweenness condition for all $v \in V \setminus \{v_T, v_B, v_\infty\}$. We now focus on solving the horizontal betweenness problem.

Although the adequate representation of a solution of a betweenness problem is a permutation of the vertices we model the problem as a continuous one. The advantage is that in the continuous formulation we can use ideas from spring-embedding and solve the problem with the aid of discrete harmonic functions (cf. [10]). Consider the following system of linear equations in the variables x_v:

(H) $x_v = \frac{1}{2}(x_{w'} + x_{w''})$ whenever $B_v = \{w', w''\}$ and $x_{v_L} = 0$, and $x_{v_R} = 1$.

Lemma 2 *The system (H) has a unique solution.*

Proof. The system has as many equations as it has variables. We claim that the corresponding homogeneous system only has the trivial solution. From the claim it follows that there exists a unique solution for any right hand side.

Suppose that (z_v) is a non-trivial solution of the homogeneous system. Consider a variable of maximal absolute value $|z_u| \neq 0$ and let $A = \{v \in V : |z_v| = |z_u|\}$. Since G is connected there is some edge connecting A to $V \setminus A \supseteq \{v_L\}$. Since O_G is Eulerian there is an edge $v \to w'$ with $v \in A$ and $w' \in V \setminus A$. If $v \to w''$ is the other out-edge of v, then $|z_v| > \frac{1}{2}(|z_{w'}| + |z_{w''}|) \geq |\frac{1}{2}(z_{w'} + z_{w''})|$, a contradiction. □

The solution of system (H) does not necessarily give a solution of the betweenness problem. Indeed a solution of the system may clump a set of vertices at a single point. Such a clumping can be accidental (resolvable by a perturbation $x_v = \frac{1}{2}((1+\varepsilon)x_{w'} + (1-\varepsilon)x_{w''})$ of the equations) or the clumping can be essential

(this happens e.g. if v_L is a cut vertex and one component is fixed at 0). To exclude essential clumpings we need a little more than mere connectivity.

Let $S \subset V$ be a set of vertices. A *pole* of S is a vertex $v \in S$ such that in O_G there is an edge $v \to w$ with $w \notin S$. Note that v_L, v_R are the poles of $V' = V \setminus \{v_\infty\}$.

Proposition 3 *The betweenness problem has a solution $\iff$ every subset $S \subset V'$ with $|S| > 1$ has at least two poles.*

Proof. Suppose a permutation π of V is a solution to the betweenness problem. It follows immediately from the definitions that the leftmost and the rightmost vertex of S with respect to π are poles of S.

Now suppose that every subset $S \subset V'$ with $|S| > 1$ has at least two poles. We consider solutions of perturbed systems (H_ϵ) were the equations of the vertices perturbed by independent parameters ε. Consider a solution z_v of a perturbed problem where the number of pairs of vertices sharing a position is minimized. Suppose that this solution has a clump A at a, i.e., $A = \{v : z_v = a\}$ and $|A| \geq 2$. Since A has at least two poles and at least one of v_L, v_R is not in A there is an edge $v \to w$ leaving A. If for all edges $v \to w$ leaving A we have $z_v > z_w$ we get a contradiction because if $v \to w''$ is the other out-edge of v, then $\frac{1}{2}((1+\varepsilon)z_w + (1-\varepsilon)z_{w''}) > a = z_v$. Basically the same argument shows that if $v \in A$ has an edge $v \to w$ with $z_v > z_w$, then the other out-edge $v \to w''$ has $z_v < z_{w''}$. Increasing the parameter ε of the equation of v by a small $\delta > 0$ will move v slightly to the left. This resolves the clump A. By choosing δ small enough we can make sure that the effect of moving v on other vertices will not form new clumps. The contradiction shows that there is a system (H_ϵ) such that the solution has no clumps. □

Theorem 4 *A four-regular graph $G = (V, E)$ with a designated vertex $v_\infty \in V$ admits an extended one-bend drawing iff $|E[S]| \leq 2|S| - 2$ for all $S \subset V$.*

Proof. The necessity of the density condition was shown in Proposition 1.

To prove sufficiency we choose an Eulerian orientation O_G of G such that with respect to O_G every $S \subset V'$ with $|S| > 1$ has at least two poles. In fact we want that the two poles condition also holds for the reverse orientation $\overline{O_G}$. In Proposition 5 below we show that such an Eulerian orientation O_G of G exists.

Proposition 3 implies that the horizontal betweenness problem associated with O_G has a solution. Let $\pi_x : V' \to \{1, .., |V'|\}$ be the ordering of the vertices obtained by sorting them according to the x_v values.

The two poles condition for $\overline{O_G}$ allows us to use Proposition 3 again and infer the existence of a solution y_v of the vertical betweenness associated with O_G. Let π_y be the ordering of the vertices obtained by sorting them according to the y_v values.

Let $n = |V'|$. We now can safely place the vertices of G' on integral points of the $n \times n$ grid $v \to (\pi_x(v), \pi_y(v))$ and draw the edges with one bend. The horizontal and vertical betweenness conditions guarantee that there is no conflict of direction. □

Proposition 5 *A four-regular multi-graph G with $|E[S]| \leq 2|S| - 2$ for all $S \subset V$ has an Eulerian orientation O_G such that with respect to O_G and $\overline{O_G}$ every $S \subset V$ with $|S| > 1$ has at least two poles.*

Proof. Note that the condition implies that G is 4-edge connected. Hence, if O is an Eulerian orientation and $S \subset V$, then there are at least two edges leaving S. If there are two such edges with different tail-vertices, then there are two poles in S. Conversely, if S with $|S| > 1$ has only one pole in O, then there are only four edges in the cut $E[S, \overline{S}]$ and the two edges in O pointing from S to $\overline{S}$ have the same tail $v \in S$. Note that in this case the other two edges incident to v belong to $E[S]$.

We call a vertex v *dangerous* if v has four different neighbors w_1, w_2, w_3, w_4 and there exists some $S \subset V$ with $|E[S, \overline{S}]| = 4$, $v, w_1, w_2 \in S$, and $w_3, w_2 \in \overline{S}$. In the first step of the construction of an Eulerian orientation with the desired properties we take dangerous vertices and duplicate them, i.e., if v is dangerous with neighbor set w_1, w_2, w_3, w_4 as above, then we replace v and its four edges by vertices v' and v'' and edges $(w_1, v'), (w_2, v'), (w_3, v''), (w_4, v'')$ and a double-edge connecting v' and v'', see Figure 2.

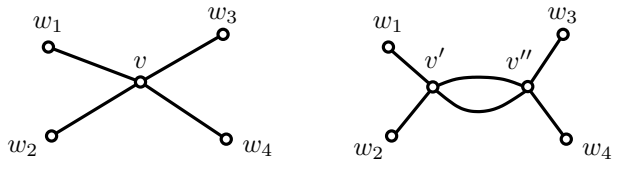


Figure 2: The replacement of a dangerous vertex.

The replacement of a dangerous vertex reduces the number of vertices with four different neighbors the step ends with a four-regular graph G_1^+ that has no dangerous vertex. Remove all double edges from G_1^+, this yields G_2^+. Since all vertices of G_2^+ are of degree 4 or 2 or 0 there is an Eulerian orientation O^+ of G_2^+. Merging pairs of vertices v', v'' that were created by the replacement of a dangerous vertex we recover the original graph G. Let O_G be the orientation of G inherited from O^+. The orientation O_G is Eulerian. Moreover, it has the property that every $S \subset V$ with $|S| > 1$ has at least two poles with respect to O_G as well as with respect to $\overline{O_G}$. For the proof of this property note that whether $u \neq v$ is dangerous is not effected by the replacement of v by v' and v''. Also if v

is dangerous because of two sets S and T then either $(N(v) \cap S) = (N(v) \cap T)$ or $(N(v) \cap S) \cap (N(v) \cap T) = \emptyset$. □

3 The general case

Theorem 6 *If $G = (V, E)$ is a graph with $\Delta \leq 4$ and with $|E[S]| \leq 2|S| - 2$ for all $S \subseteq V$ then there is an extended one-bend drawing of G with respect to any designated vertex $v_\infty \in V$.*

Proof. We aim at using Theorem 4. To this end we define a 4-regular graph G^+ that has G as a subgraph. Let X be a set of $4|V| - 2|E|$ new vertices, i.e., X is disjoint from V. Let E_C be the set of edges of a bipartite graph with color classes X and V such that every vertex in X has degree 1 and a vertex $v \in V$ has degree $4 - \mathsf{deg}_G(v)$ in E_C. Finally, let E_X be the edge set of a 3-connected 3-regular graph on the vertex set X. Since $|X|$ is even and $|X| \geq 4$ we can e.g. take the dual of a plane triangulation as (X, E_X). Let $G^+ = (V \cup X, E \cup E_C \cup E_X)$. This graph is clearly 4-regular. It remains to verify the density condition of Theorem 4. For $S \subset V$ this is part of the assumption. If $\emptyset \neq S \cap X \neq X$, then there are at least three edges in $E[S, \overline{S}] \cap E_X$ but since G^+ is 4-regular the size of the cut is even, hence at least four. □

The proof does not cover the case $|E| = 2|V| - 1$. However, the density condition is also sufficient for this case. This can be shown by reduction to the situation of Theorem 4.

4 Additional Problems

(i) Improvements in the area requirement and in the total number of bends can be achieved through an obvious compaction that can be applied in a post processing phase. Is it possible to guide the algorithm so that the gain of this compaction can be controlled?

(ii) The technique of this paper clearly yields a polynomial algorithm for one-bend drawings. It might be worth investigating whether the technique can be used for a linear or near linear time algorithm.

(iii) 2-Bends Problem (c.f. Wood [19]): Does every simple graph with maximum degree 6 have a 3-D orthogonal graph drawing with at most 2 bends per edge?

(iv) Extend the new technique using Eulerian orientation to graphs with higher degree and box representation for the vertices. Improve the unbalanced orientations used in Biedl/Kaufmann and prove better bounds for vertices with bounded aspect ratio.

Acknowledgment

The problem was presented by Ignaz Rutter at the Homonolo Workshop in Nova Louka, December 2011. We thank Ignaz and the organizers of the workshop.

References

[1] T. C. Biedl and G. Kant, *A better heuristic for orthogonal graph drawings*, Comput. Geom., 9 (1998), pp. 159–180.

[2] T. C. Biedl and M. Kaufmann, *Area-efficient static and incremental graph drawings*, in ESA, LNCS 1284, 1997, pp. 37–52.

[3] T. C. Biedl, T. Thiele, and D. R. Wood, *Three-dimensional orthogonal graph drawing with optimal volume*, Algorithmica, 44 (2006), pp. 233–255.

[4] P. Eades, A. Symvonis, and S. Whitesides, *Three-dimensional orthogonal graph drawing algorithms*, Disc. Appl. Math., 103 (2000), pp. 55–87.

[5] M. Formann and F. Wagner, *The vlsi layout in various embedding models*, in WG, LNCS 484, 1991, pp. 130–139.

[6] U. Fössmeier, *Interactive orthogonal graph drawing: Algorithms and bounds*, in GD, LNCS 1353, 1997, pp. 111–123.

[7] U. Fössmeier, G. Kant, and M. Kaufmann, *2-visibility drawings of planar graphs*, in GD, LNCS 1190, 1996, pp. 155–168.

[8] G. Kant and X. He, *Two algorithms for finding rectangular duals of planar graphs*, in WG, LNCS 790, 1993, pp. 396–410.

[9] M. Kramer and J. van Leeuwen, *The complexity of wire-routing and finding minimum area layouts for arbitrary vlsi circuits*, Adv. in Comp. Res., 2 (1984), pp. 129–146.

[10] L. Lovász, *Geometric representations of graphs.* `www.cs.elte.hu/~lovasz/geomrep.pdf`, 2009.

[11] A. Papakostas, J. M. Six, and I. G. Tollis, *Experimental and theoretical results in interactive orthogonal graph drawing*, in GD, LNCS 1190, 1996, pp. 371–386.

[12] A. Papakostas and I. G. Tollis, *A pairing technique for area-efficient orthogonal drawings*, in Graph Drawing, LNCS 1190, 1996, pp. 355–370.

[13] M. S. Rahman, N. Egi, and T. Nishizeki, *No-bend orthogonal drawings of subdivisions of planar triconnected cubic graphs*, IEICE Transactions, 88-D (2005), pp. 23–30.

[14] M. S. Rahman, T. Nishizeki, and S. Ghosh, *Rectangular drawings of planar graphs*, J. Algorithms, 50 (2004), pp. 62–78.

[15] M. W. Schäffter, *Drawing graphs on rectangular grids*, Disc. Appl. Math., 63 (1995), pp. 75–89.

[16] J. A. Storer, *On minimal-node-cost planar embeddings*, Networks, 14 (1984), pp. 181–212.

[17] R. Tamassia, *On embedding a graph in the grid with the minimum number of bends*, SIAM J. Comput., 16 (1987), pp. 421–444.

[18] R. Wiese and M. Kaufmann, *Adding constraints to an algorithm for orthogonal graph drawing*, in Graph Drawing, LNCS 1547, 1998, pp. 462–463.

[19] D. R. Wood, *Orthogonal graph drawing page*, `www.ms.unimelb.edu.au/~woodd/ortho.html`.

Non-Crossing Connectors in the Plane

Jan Kratochvíl* Torsten Ueckerdt*

Abstract

We consider the non-crossing connectors problem, which is stated as follows: Given n simply connected regions $R_1, \ldots, R_n$ in the plane and finite point sets $P_i \subset R_i$ for $i = 1, \ldots, n$, are there non-crossing connectors γ_i for (R_i, P_i), i.e., arc-connected sets γ_i with $P_i \subset \gamma_i \subset R_i$ for every $i = 1, \ldots, n$ such that $\gamma_i \cap \gamma_j = \emptyset$ for all $i \neq j$?

We prove that non-crossing connectors do always exist if the regions form a collection of pseudo-disks, i.e., the boundaries of every pair of regions intersect at most twice. We provide a simple polynomial-time algorithm if the regions are axis-aligned rectangles. Finally we prove that the general problem is NP-complete, even if the regions are convex, the boundaries of every pair of regions intersect at most four times and P_i consists of only two points on the boundary of R_i for $i = 1, \ldots, n$.

1 Introduction

Connecting points in the plane by straight lines in a non-crossing way is a natural problem in computational geometry. For example, a bunch of research has been done for connecting n red and n blue points in the plane by a matching, a non-crossing path that alternates between red and blue vertices, or two non-crossing spanning trees, one on each color. We refer to the survey article of Kaneko and Kano [10]. Recent investigations consider the problem of finding a non-crossing matching between n ordered pairs of points sets, e.g., between a single point and a vertical line [1], or between two vertical line segments [21].

In this paper, we investigate what happens if we allow general curves instead of just straight line segments. Moreover, we want to connect not only n pairs of points, but n finite sets of points. Of course, we can always find such non-crossing curves, unless two point sets intersect. But if we impose for every point set a region that the corresponding curve must be contained in, then determining whether or not such non-crossing curves exist is a non-trivial problem, which is formally stated as follows.

Given: Collection $R_1, \ldots, R_n$ of simply connected subsets of the plane, called *regions*, and a finite point set $P_i \subset R_i$, for $i = 1, \ldots, n$ with $P_i \cap P_j = \emptyset$ for $i \neq j$.

Question: Is there a collection $\gamma_1, \ldots, \gamma_n$ of curves, called *connectors* such that $P_i \subset \gamma_i \subset R_i$ for $i = 1, \ldots, n$ and $\gamma_i \cap \gamma_j = \emptyset$ for $i \neq j$?

Related work. A lot of research has been done for connecting points in the plane with straight line segments in a non-crossing way. For instance, a point set P is *universal* for a class $\mathcal{G}$ of planar graphs if the vertices of every graph $G \in \mathcal{G}$ can be embedded onto the points P such that straight edges do not cross. It has been shown [17, 3] that every set of n points in general position is universal for the class of n-vertex outer-planar graphs. The smallest point set that is universal for the class $\mathcal{G}_n$ of all n-vertex planar graphs consists of at least $1.235n$ [4, 14] and at most $\mathcal{O}(n^2)$ [9, 19] points. Deciding whether a given graph embeds on a given point set, is known to be NP-complete [2].

In many variants edges are allowed to be more flexible than straight line segments. Kaufmann and Wiese [11] show that every n-element point set is universal for $\mathcal{G}_n$ if every edge is a polyline with at most 2 bends. Moreover, some n-element point sets are universal for $\mathcal{G}_n$ if edges bend at most once [7]. If the bending points have to be embedded onto P as well, then universal sets of size $\mathcal{O}(n^2/\log n)$ for 1 bend, $\mathcal{O}(n \log n)$ for 2 bends, and $\mathcal{O}(n)$ for 3 bends are known [6]. A variant with so-called ortho-geodesic edges was studied especially for trees of maximum degree 3 and 4 [5].

Variants where every vertex v has an associated subset P_v of P of its possible positions has been studied for much simpler graphs like matchings and cycles, however for straight edges only. For cycles, deciding whether a set of non-crossing edges exists is known to be NP-complete, even if every P_v is a vertical line segment or every P_v is a disk [16]. For matchings, NP-completeness has been shown if $|P_v| \leq 3$ [1], or P_v is a vertical line segment of unit length [21]. The latter result still holds if every edge $\{u, v\}$ may be a monotone curve within the convex hull of $P_u \cup P_v$ [20].

Our results. We mainly consider the computational complexity of the non-crossing connectors problem. In Section 2 we prove that the problem lies in NP, which is non-trivial. Moreover we show that non-crossing connectors do always exist if the given regions

*Department of Applied Mathematics, Charles University in Prague, honza@kam.mff.cuni.cz, torsten@kam.mff.cuni.cz, Research was supported by GraDR EUROGIGA project No. GIG/11/E023.

form a collection of pseudo-disks, i.e., the boundaries of every pair of regions intersect at most twice. In Section 3 we show that the problem is polynomial if the regions are axis-aligned rectangles, while the general problem is NP-complete, even if the regions are convex, the boundaries of every pair of regions intersect at most four times, and $|P_i| = 2$ for every $i = 1, \ldots, n$.

2 NP-Membership and Pseudo-Disks

The boundary of every region R_i is a simple closed curve, denoted by ∂R_i. We assume here and for the rest of the paper that $\partial R_i \cap \partial R_j$ is a finite point set. We may think of $\bigcup_{i=1}^{n} \partial R_i$ as an embedded planar graph $G = (V, E)$ with vertex set $V = \{p \in \mathbb{R}^2 \mid p \in \partial R_i \cap \partial R_j, i \neq j\}$ and edge set $E = \{e \subset \mathbb{R}^2 \mid e \text{ is a connected component of } \bigcup \partial R_i \setminus V\}$. A point $p \in \partial R_i \cap \partial R_j$ is either a *crossing point* or a *touching point*, depending on whether the cyclic order of edges in ∂R_i and ∂R_j around p is alternating or not. We say that two regions R_i, R_j are *k-intersecting* for $k \geq 0$ if $|\partial R_i \cap \partial R_j| \leq k$ and all these points are crossing points, i.e., w.l.o.g. k is even. A set $R_1, \ldots, R_n$ of regions is *k-intersecting* if this is the case for any two of them. For example, $R_1, \ldots, R_n$ are 0-intersecting if and only if[1] they form a nesting family, i.e., $R_i \cap R_j \in \{\emptyset, R_i, R_j\}$ for $i \neq j$.

Regions $R_1, \ldots, R_n$ are a called a *collection of pseudo-disks* if and only if they are 2-intersecting. Usually two pseudo-disks may have one touching point. However, this can be locally modified into two crossing points without affecting the existence of non-crossing connectors. A collection of axis-aligned rectangles is always 4-intersecting, but not necessarily 2-intersecting. We close this section by showing that the non-crossing connectors problem belongs to NP.

Proposition 1 *Non-crossing connectors is in NP.*

Proofsketch. We reduce our problem to Weak Realizability of Abstract Topological Graphs. Abstract topological graphs (AT-graphs, for short) have been introduced in [13] as triples (V, E, R) where (V, E) is a graph and R is a set of pairs of edges of E. The AT-graph (V, E, R) is weakly realizable if (V, E) has a drawing (not necessarily non-crossing) in the plane such that $ef \in R$ whenever the edges e, f cross in the drawing. Weak realizability of AT-graphs is NP-complete. The NP-hardness was shown in [12], and the NP-membership was shown in [18].

We assume the input of the problem be described as the plane graph G (the boundaries of the regions) described above. First, we augment G to a planar triangulation G' and then, for every i, add edges that connect the points of P_i by a path. Call the resulting graph $\widetilde{G}$. Now define an AT-graph with underlying graph $\widetilde{G}$ by allowing the connecting edges of points P_i to cross anything but other connecting edges and the edges resulting from the boundary of R_i. No other edges are allowed to cross, in particular, no edges of G'' may cross each other. It is straightforward to see that a weak realization of this AT-graph is a collection of non-crossing connectors, and vice versa. Thus the NP-membership follows by the result of Schaefer et al. [18] and the fact that our construction of $\widetilde{G}$ is polynomial. □

Theorem 2 *If $R_1, \ldots, R_n$ is a collection of pseudo-disks, then non-crossing connectors exist for any point sets $P_i \subset R_i$ $(i = 1, \ldots, n)$.*

Proofsketch. The proof is constructive. Let the regions $R_1, \ldots, R_n$ be labeled such that for every $i = 2, \ldots, n$ the set $R_i \setminus (R_1 \cup \cdots \cup R_{i-1})$ contains some point p_i (not necessarily in P_i). We start by defining a connector γ_1 for (R_1, P_1) arbitrarily. For $i = 2, \ldots, n$ we defined connectors $\gamma_1, \ldots, \gamma_{i-1}$, which partition R_i into connected components in some way. Let C_0 be the component containing p_i. We reroute some of the existing connectors until P_i is completely contained in C_0, and then define γ_i within C_0 arbitrarily.

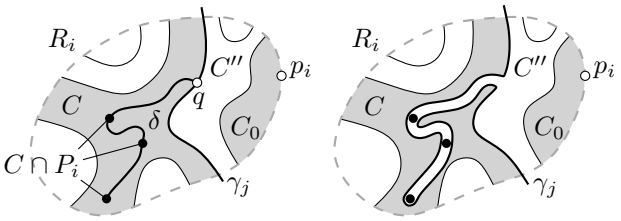


Figure 1: Rerouting connector γ_j around δ.

The adjacency graph between the components of R_i is a tree T, which we consider to be rooted at C_0. Let $C \neq C_0$ be a component such that $C \cap P_i \neq \emptyset$ but $C' \cap P_i = \emptyset$ for every descendant C' of C in T. Let γ_j be the connector that forms the border between C and its father C'' in T, i.e., $j < i$. Let q be a point on $\gamma_j \cap R_i$ and δ be any curve with $(C \cap P_i) \cup \{q\} \subset \delta \subset C \cup \{q\}$. See Figure 1 for an example. Next, γ_j is opened at q and rerouted within a very small tube around δ. That this operation is feasible, which means that δ lies within R_j, follows from the fact that R_i and R_j are pseudo-disks and $p_i \notin R_j$.

The rerouting does only affect the subtree of T under C''. Moreover, C'' now contains all points in $P_i \cap C$. After finitely many steps we have $P_i \subset C_0$ and thus can define the connector γ_i for (R_i, P_i). □

3 Computational Complexity

In this section we consider the computational complexity of the non-crossing connectors problem. First

[1] To be precise, we always require for the "if"-part that $\partial R_i \cap \partial R_j$ is a finite set of crossing points for $i \neq j$.

we show that the problem is polynomial if the regions $R_1, \ldots, R_n$ are axis-aligned rectangles. We say that two rectangles R_i, R_j form a *cross* if they are 4-intersecting, and form a *filled cross* if additionally both connected components of $R_i \setminus R_j$ contain a point from P_i, and both connected components of $R_j \setminus R_i$ contain a point from R_j. Obviously, non-crossing connectors do not exist if some pair of rectangles is a filled cross. It turns out that the absence of filled crosses is a necessary and sufficient condition for the existence of non-crossing connectors.

Theorem 3 *A set of axis-aligned rectangles admits a set of non-crossing connectors if and only if it does not contain a filled cross.*

Proofsketch. For the "if"-part consider axis-aligned rectangles $R_1, \ldots, R_n$ such that no two of them form a filled cross. If there is no cross at all, then the rectangles are pseudo-disks and non-crossing connectors exist by Theorem 2. So let R_iR_j be a cross where $R_i \cap R_j$ is inclusion-minimal among all crosses. W.l.o.g. let C be a connected component of $R_j \setminus R_i$ that contains no point from P_j. We chop off a slight superset of C from R_j and replace R_j by the truncated axis-aligned rectangle. See Figure 2 for an example.

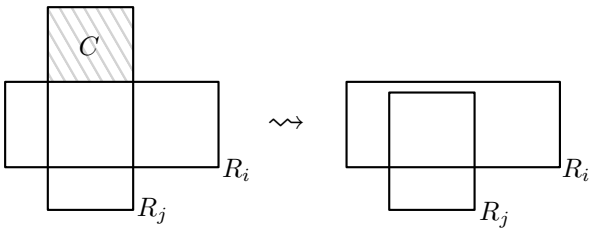


Figure 2: Chopping off C from rectangle R_j.

From the inclusion-minimality of $R_i \cap R_j$ follows that every rectangle R_k crosses the new rectangle R_j only if it crosses the former R_j, too. Hence no (filled) cross is created by the chopping operation. Since the pair R_iR_j is no longer a cross, the number of crosses has decreased. Repeating the procedure at most $\binom{n}{2}$ times results in a collection of pseudo-disks, which by Theorem 2 admit non-crossing connectors. □

Corollary 4 *It can be tested in $\mathcal{O}(n^2)$ whether or not a set of n axis-aligned rectangles admits a set of non-crossing connectors.*

Finally, we prove NP-completeness of the non-crossing connectors problem. By Proposition 1 the problem is in NP. We prove NP-hardness, even if the regions and their point sets are very restricted.

Theorem 5 *The non-crossing connectors problem is NP-complete, even if the regions are 4-intersecting convex polygons and for every $i = 1, \ldots, n$ the set P_i consists of two points on the boundary of R_i.*

The proof of Theorem 5 is rather technical and pretty long. We sketch here the ideas with non-convex, but still 4-intersecting, regions. We do a polynomial reduction from planar 3-SAT, i.e., we are given a formula ψ in conjunctive normal form where each clause has at most 3 literals, that is positive or negated variables. Moreover, the *formula graph* G_ψ, namely the bipartite graph whose vertices are the clauses and variables of ψ, and whose edges are given by $\{x, c\}$ with variable x contained in clause c, is planar. It is known [15] that planar 3-SAT is NP-complete, even if every variable appears in at most 3 clauses, i.e., G_ψ has maximum degree 3 [8].

For every clause c we define five regions together with two points per region as illustrated in Figure 3. The three red regions are associated with the literals contained in c. Each red region contains a small part outside the black region. Given non-crossing connectors for these five regions, we say a literal x *satisfies the clause* c if the corresponding connector is completely contained in the black region. The clause gadget is defined so that at least one literal satisfies the clause, and any literal may do so. See Figure 3 for an illustration. If c consists of only two literals, then the red point of the third red region that is contained in the blue region is put outside the black region, which implies that this artificial variable may not satisfy c.

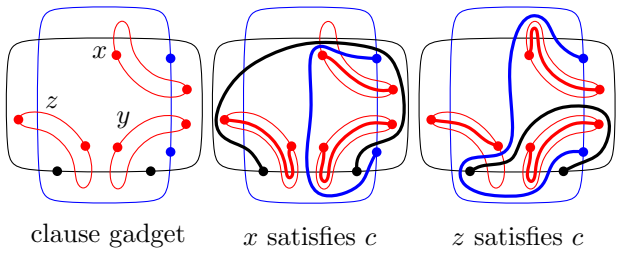


Figure 3: Clause gadget for clause c consisting of literals x, y, z.

W.l.o.g. every variable x appears in three clauses, positive in c_1 and negated in c_2, c_3, or vice versa. Now x is associated with a red region in the clause gadgets for c_1, c_2 and c_3, each containing a part outside the corresponding black region. We bring together these three parts (together with the corresponding black and blue regions) and overlap them as shown in Figure 4. It follows that if non-crossing connectors exist, then x satisfies only c_1, or c_2 and c_3 but not c_1.

4 Conclusion

In this paper we investigated the computational complexity of the non-crossing connectors problem. We proved that it is polynomial if the regions are pseudo-disks or axis-aligned rectangles. It might be worthwhile to derive from our proofs polynomial-time al-

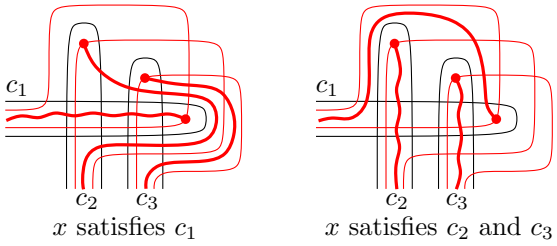


Figure 4: Variable gadget for variable x contained in clauses c_1, c_2, c_3.

gorithms to actually compute non-crossing connectors. Moreover, we have shown that the non-crossing connectors problem is NP-complete for 4-intersecting convex regions, even if every P_i consists of 2 elements. However, the complexity in case each R_i is the convex hull of the corresponding P_i remains open. It is as well interesting to consider other sets of regions, like ellipsoids or isosceles triangles with horizontal bases.

Instead of fixing the position of the points in P_i one may consider a set of possible positions for every such point. From previous results [1] it follows that this variant is NP-complete if every point has at most 3 possible positions, but this proof again does not work if $R_i = \text{conv}(P_i)$ for $i = 1, \ldots, n$. What if we want to connect pairs of vertical straight segments by non-crossing curves, each within the convex hull of the corresponding segments?

Furthermore, we could allow $P_i \cap P_j \neq \emptyset$ for $i \neq j$ and allow connectors γ_i, γ_j to intersect in $P_i \cap P_j$. If $|P_i| = 2$ for every i, this corresponds to drawing a given planar graph with fixed vertex positions and curved edges, each lying within a prescribed region. In this variant non-crossing connectors sometimes do not exist even when regions are pseudo-disks.

Acknowledgments

We would like to thank Maria Saumell, Stefan Felsner and Irina Mustata for fruitful discussions.

References

[1] G. Aloupis, J. Cardinal, S. Collette, E. Demaine, M. Demaine, M. Dulieu, R. Fabila-Monroy, V. Hart, F. Hurtado, S. Langerman, M. Saumell, C. Seara, and P. Taslakian. Non-crossing matchings of points with geometric objects. *Computational Geometry: Theory and Applications*. to appear.

[2] S. Cabello. Planar embeddability of the vertices of a graph using a fixed point set is NP-hard. *Journal of Graph Algorithms and Applications*, 10(2):353–363, 2006.

[3] N. Castañeda and J. Urrutia. Straight line embeddings of planar graphs on point sets. In *8th Canadian Conference on Computational Geometry (CCCG)*, pages 312–318, 1996.

[4] M. Chrobak and H. Karloff. A lower bound on the size of universal sets for planar graphs. *SIGACT News*, 20, 1989.

[5] E. Di Giacomo, F. Frati, R. Fulek, L. Grilli, and M. Krug. Orthogeodesic point-set embedding of trees. In *19th International Symposium on Graph Drawing (GD 2011)*, 2011.

[6] V. Dujmović, W. Evans, S. Lazard, W. Lenhart, G. Liotta, D. Rappaport, and S. Wismath. On point-sets that support planar graphs. In *19th International Symposium on Graph Drawing (GD 2011)*, 2011.

[7] H. Everett, S. Lazard, G. Liotta, and S. Wismath. Universal sets of n points for one-bend drawings of planar graphs with n vertices. *Discrete & Computational Geometry*, 43:272–288, 2010.

[8] M. R. Fellows, J. Kratochvíl, M. Middendorf, and F. Pfeiffer. The complexity of induced minors and related problems. *Algorithmica*, 13:266–282, 1995.

[9] H. Fraysseix, J. Pach, and R. Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10:41–51, 1990.

[10] A. Kaneko and M. Kano. Discrete geometry on red and blue points in the plane – a survey. In *Discrete and Computational Geometry, The Goodman-Pollack Festschrift*, pages 551–570, 2003.

[11] M. Kaufmann and R. Wiese. Embedding vertices at points: Few bends suffice for planar graphs. *Journal of Graph Algorithms and Applications*, 6(1):115–129, 2002.

[12] J. Kratochvíl. String graphs II. Recognizing string graphs is NP-hard. *Journal of Combinatorial Theory, Series B*, 52(1):67–78, 1991.

[13] J. Kratochvíl, A. Lubiw, and J. Nešetřil. Noncrossing subgraphs in topological layouts. *SIAM Journal on Discrete Mathematics*, 4(2):223–244, 1991.

[14] M. Kurowski. A 1.235 lower bound on the number of points needed to draw all n-vertex planar graphs. *Information Processing Letters*, 92(2):95–98, 2004.

[15] D. Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982.

[16] M. Löffler. Existence of simple tours of imprecise points. In *23rd European Workshop on Computational Geometry (EuroCG)*, 2007.

[17] J. Pach, P. Gritzmann, B. Mohar, and R. Pollack. Embedding a planar triangulation with vertices at specified points. *American Mathematical Monthly*, 98:165–166, 1991.

[18] M. Schaefer, E. Sedgwick, and D. Štefankovič. Recognizing string graphs in NP. *Journal of Computer and System Sciences*, 67(2):365–380, 2003.

[19] W. Schnyder. Embedding planar graphs on the grid. In *1st ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 138–148, 1990.

[20] B. Speckmann. personal communication, 2011.

[21] K. Verbeek. *Non-crossing paths with fixed endpoints*. Master's Thesis, Eindhoven, 2008.

Simplified Medial-Axis Approximation with Guarantees*

Christian Scheffer
Faculty of Computer Science,
Technische Universität Dortmund,
44227 Dortmund, Germany
christian.scheffer@cs.tu-dortmund.de

Jan Vahrenhold
Department of Computer Science,
University of Münster, Einsteinstr. 62,
48149 Münster, Germany
jan.vahrenhold@uni-muenster.de

Abstract

We present an algorithm that approximates the medial axis of a manifold in $\mathbb{R}^3$ given by a sufficiently dense point sample. The resulting, non-discrete approximation converges to the medial axis as the sampling density approaches infinity and we establish a homotopy between the canonical parameterizations of both structures. While there is no subquadratic upper bound on the output complexity of previous algorithms for non-discrete medial axis approximation, the output of our algorithm is guaranteed to be of linear size. The algorithm is (arguably) simpler than previous approaches and practically efficient.

1 Introduction

The medial axis of a geometric shape Γ in Euclidean space is the closure of the set of all points for which there exists more than one closest point on the shape. Using medial axes to represent geometric shapes has a variety of applications as surveyed by, e.g., Dey and Zhao [5]. In this paper, we consider the case that Γ is a manifold in $\mathbb{R}^3$ that is given by a sufficiently dense, discrete point sample S. It has been shown by Amenta *et al.* [4] that the medial axis can be approximated by a well-defined, discrete subset of the Voronoi diagram of S, and Dey and Zhao [5] proved that it is also possible to derive a non-discrete approximation of the medial axis from the Voronoi diagram. As in previous approaches [4], we avoid points at infinity by assuming that Γ is contained inside an open, bounded region Q.

Definition 1 ([4, p. 132]) *A ball $B = B_{c,\rho} \subset Q$ with center c and radius ρ is empty (with respect to Γ) if the interior of B contains no point of Γ. A* medial ball *is a maximal empty ball. The set of centers of the medial balls, the so-called* medial points *form the* medial axis M_Γ *of Γ.*

Since Q is bounded, the medial axis as defined in Definition 1 is a compact set.

*Part of this work has been supported by Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876 "Providing Information by Resource-Constrained Analysis" (http://sfb876.tu-dortmund.de), project A2.

Since the medial axis captures the shape of the manifold Γ, the *local feature size*, $lfs(x)$, of a point $x \in \Gamma$, defined as the distance from a point on the manifold to the closest point on the medial axis [2, 8] locally captures the curvature and folding of Γ. To be able to guarantee that a point sample S of a manifold Γ is dense enough to capture Γ, Amenta *et al.* [2] introduced the notion of an *ε-sample,* i.e., a point sample S such that for every point $x \in \Gamma$ there is a sample point $s \in S$ with $|xs| \leqslant \varepsilon \cdot lfs(x)$. The analysis usually assumes that S is dense enough, that is that ε is less than some small constant, e.g., $\varepsilon \leqslant 0.06$ [3]. For technical reasons, we require $\varepsilon \leqslant \frac{1}{22} < 0.06$.

2 Related Work

The two results most relevant for our work are the construction of the PowerShape in the context of the PowerCrust algorithm developed by Amenta *et al.* [4] and the Voronoi-based algorithm by Dey and Zhao [5]. Both algorithms define a subset of the object to be approximated which is parameterized (directly or indirectly) by the unknown sampling parameter ε and show that these objects converge to the medial axis as the sampling density approaches infinity

Definition 2 *Let x be an arbitrary point on the manifold Γ. The* inner medial radius *of x is defined to be the radius ρ of the* inner medial ball *$B_{c,\rho}$, i.e., the medial ball that touches Γ in x and at least one other point such that c lies in the interior of Γ. Then, the* inner medial axis *M_Γ^+ is the (closure of the) set of the centers of all inner medial balls (for all $x \in \Gamma$). Similarly, define the* outer medial radius, *the* outer medial ball, *and the* outer medial axis *M_Γ^-.*

Amenta *et al.* [4] define a parameterized subset of the object to be approximated:

Definition 3 ([4, p. 132]) *For $\gamma \geqslant 0$, the* inner γ-medial axis *γ-$M_\Gamma^+ \subseteq M_\Gamma^+$ is defined to be the set of all inner medial points c such that there are at least two points $u_1, u_2 \in \Gamma$ on the boundary of the medial ball centered at c that form a* medial angle *$\angle(u_1, c, u_2) > 2\gamma$. Similarly, define the* outer γ-medial axis *γ-$M_\Gamma^- \subseteq M_\Gamma^-$.*

In their PowerCrust approach, Amenta *et al.* [4] first approximate the medial axis by so-called *poles*.

Definition 4 ([1, 4]) *The* poles *of some $s \in S$ are the two vertices of $Vor(s)$ farthest from s, one on either side of Γ. A pole p is called an* outer *pole if it lies outside Γ, and an* inner *pole otherwise.*

The main lemma of Amenta *et al.* [4] (which is also of crucial importance to Dey and Zhao's algorithm) that guarantees the quality of the approximation of the medial axis using a subset of the set of poles states that the distance between a point on the γ-medial axis and a well-defined pole, that is no point at infinity, converges to zero as ε approaches zero:

Lemma 1 ([4, Lemma 34]) *Let $B_{m,\rho}$ be a medial ball such that m belongs to the inner (outer) γ-medial axis, for some fixed γ, with $\gamma \in \Omega(\varepsilon^{1/3})$. Let $s \in S$ be the nearest sample to m. Then the distance from m to the inner (outer) pole p of s is in $\mathcal{O}\left(\rho\varepsilon^{2/3}\right)$.*

The authors of [4] establish the pointwise convergence of the poles to the medial axis. They prove that they can construct the PowerShape, the *dual shape* of the PowerCrust, which has the set of poles as its vertices. There is no (explicit) proof, however, that each non-vertex point on the PowerShape converges to a point on the medial axis.

Dey and Zhao [5] observe that approximating the medial axis by the set of poles alone does not yield a non-discrete approximation of the medial axis as needed in many applications. Such a non-discrete approximation is only possible on the basis of the power diagram, i.e., it requires computing a second (weighted) Voronoi diagram. The authors point out that the resulting object tends to be "noisy".

To alleviate these shortcomings, Dey and Zhao propose to directly derive an approximation of the medial axis from the Voronoi diagram (needed for the pole computation in any case). In a nutshell, the algorithm filters Delaunay edges using an "angle" or "ratio" filter and approximates the medial axis by the Voronoi faces dual to edges that result from the filter step and the set of all Voronoi edges and vertices incident to these faces. While the authors prove a convergence result, they state that proving homotopy equivalence would be subject of "future research" [5, p. 199].

Combinatorial Complexity of the Approximations As mentioned above, the PowerShape is defined as the dual shape of the PowerCrust; thus, in standard computational geometry terminology, the PowerShape is the graph-theoretic dual of the complement of the PowerCrust relative to the power diagram [4, p. 136] (or, a subset of the weighted Delaunay tetrahedrization). Amenta *et al.* [4, Observation 5] observe that the PowerCrust, defined as a set of faces in the power diagram of the set of polar balls, is the (possibly non-regular) boundary of a three-dimensional solid. As the power diagram and thus the weighted Delaunay tetrahedrization can be of quadratic size in the number of points in the input sample, the size of the PowerShape depends on the size of the PowerCrust and can be quadratic as well. We can, however, prove the following lemma:

Lemma 2 *The approximation of the medial axis constructed from the Voronoi diagram by Dey and Zhao's algorithm has a quadratic worst-case complexity.*

3 Outline of the Algorithm

From the above discussion we conclude that three issues need to be addressed simultaneously: (1) Construct an object that has linear complexity, (2) construct an object that converges to M_Γ as $\varepsilon \to 0$, and (3) establish a homotopy between the canonical parameterizations of M_Γ and the constructed object. It turns out that we can parameterize a surprisingly simple construction, originally proposed by Hisada *et al.* [7] in the context of feature recognition, to solve this task. In a nutshell, we start from a linear-size object that is homeomorphic to the (unknown) manifold Γ and continuously map this object to an object which is used as an approximation of M_Γ. This mapping is parameterized by ε, and based upon this definition, convergence and homotopy are established.

Algorithm 1 Approximate M_Γ from an ε-sample S.

1: Initialize $\widetilde{M}_\Gamma^+$ and $\widetilde{M}_\Gamma^-$ to be the empty set each.
2: Compute the Voronoi diagram $Vor(S)$.
3: Let P^+ be the set of all inner poles and let P^- be the set of all outer poles for points in S.
4: Compute a non-trivial pointwise lower bound $\Phi(s)$ for $\varepsilon \cdot lfs(s)$ for each $s \in S$ (see [6, Lem. 8.3]).
5: **for all** $s \in S$ **do**
6: $\quad p_s^{\varepsilon,-} := p_s^- - \frac{\Phi(s)}{|p_s^- s|} \cdot \overrightarrow{p_s^- s}$, $p_s^{\varepsilon,+} := p_s^+ - \frac{\Phi(s)}{|p_s^+ s|} \cdot \overrightarrow{p_s^+ s}$
$\quad\quad \triangleright$ Move poles slightly towards samples.
7: $\quad P^{\varepsilon,-} := P^{\varepsilon,-} \cup \{p_s^{\varepsilon,-}\}$, $\quad P^{\varepsilon,+} := P^{\varepsilon,+} \cup \{p_s^{\varepsilon,+}\}$
8: $(N, E_N, F_N) :=$CoCone$(S,\ Vor(S))$.
$\quad\quad \triangleright$ Surface reconstruction with CoCone.
9: **for all** $\triangle(s_1, s_2, s_3) \in F_N$ **do**
$\quad\quad \triangleright$ Iterate over all faces of N.
10: $\quad \widetilde{M}_\Gamma^+ := \widetilde{M}_\Gamma^+ \cup \{\triangle(p_{s_1}^{\varepsilon,+}, p_{s_2}^{\varepsilon,+}, p_{s_3}^{\varepsilon,+})\}$
$\quad\quad \triangleright$ Add triangle for approximated inner poles.
11: $\quad$ **if** $p_{s_1}^-$, $p_{s_2}^-$, and $p_{s_3}^-$ exist **then**
$\quad\quad\quad \triangleright$ Ignore poles at infinity.
12: $\quad\quad \widetilde{M}_\Gamma^- := \widetilde{M}_\Gamma^- \cup \{\triangle(p_{s_1}^{\varepsilon,-}, p_{s_2}^{\varepsilon,-}, p_{s_3}^{\varepsilon,-})\}$
$\quad\quad \triangleright$ Add triangle for approximated outer poles.
13: **return** $\widetilde{M}_\Gamma := \widetilde{M}_\Gamma^+ \cup \widetilde{M}_\Gamma^-$

By construction, the set $\widetilde{M}_\Gamma$ of triangles returned by Algorithm 1 is of linear size.

4 Convergence Guarantee

Analogously to Dey and Zhao, we can prove that a subset of the set $\widetilde{M}_\Gamma$ of triangles returned by Algorithm 1 converges to M_Γ as $\varepsilon \to 0$ and that "the difference between this subset and the output is small" [5, p. 193]. Due to space constraints, we give only a summary of our proofs. We build upon the observation that the CoCone algorithm constructs a surface that is homeomorphic to Γ and converges to Γ as $\varepsilon \to 0$:

Theorem 3 ([3, Theorem 1]) *Let S be an ε-sample for a smooth surface Γ, with $\varepsilon \leqslant 0.06$. The CoCone-algorithm computes a piecewise linear manifold N homeomorphic to Γ, such that any point on N is at most $\frac{1.15\cdot\varepsilon}{1-\varepsilon} \cdot lfs(x)$ from some point $x \in \Gamma$.*

Since Amenta *et al.* [4] only prove that the set of poles converges to the medial axis, it remains to be proved that for each point x on a triangle in $\widetilde{M}_\Gamma$ there is a point m on the medial axis M_Γ such that $|xm|$ converges to zero. The following technical lemma states that if there are two sample points "close enough" to each other, the distances to their respective medial points are approximately the same. Furthermore, as the sampling density approaches infinity, the distance between these medial points converges to zero.

Lemma 4 *Let s_1 and s_2 be two points from $\arcsin\left(\varepsilon^{1/3}\right)$-$\Gamma$ such that $|s_1 s_2| \leqslant 3 \cdot \varepsilon \cdot \min\{lfs(s_1), lfs(s_2)\}$. Let m_1 and m_2 be the two corresponding medial points on the same side of Γ. Then $|s_i m_i| \leqslant (1 + 17 \cdot \varepsilon^{1/3}) \cdot |m_j s_j|$ and $|m_1 m_2| \leqslant 16 \cdot \varepsilon^{1/3} \cdot |m_j s_j|$, for $i, j = 1, 2$.*

Using this lemma, we then prove that the triangles constructed by Algorithm 1 fulfill the "small triangle property": if the circumradius of a triangle $\triangle(s_1, s_2, s_3)$ (for properly chosen sample points $s_1, s_2, s_3 \in S$) converges to zero, so does the circumradius of the triangle $\triangle(p^{\varepsilon,\star}_{s_1}, p^{\varepsilon,\star}_{s_2}, p^{\varepsilon,\star}_{s_3})$ for $\star \in \{+, -\}$. This property is crucial for proving the convergence of the set $\widetilde{M}_\Gamma$ of triangles to M_Γ.

Definition 5 ([4, p. 149]) *For any $x \in \Gamma$, let m_x be its corresponding inner (outer) medial point. Let γ be the largest angle with apex m_x formed by x and a second touching point of the inner (outer) medial ball $B_{\rho,x}$. Then x belongs to the γ'-surface γ'-Γ of Γ if $\gamma \geqslant \gamma'$. $\gamma_1 \leqslant \gamma_2$ implies γ_2-$\Gamma \subseteq \gamma_1$-$\Gamma$.*

Let $\widetilde{M}^\gamma_\Gamma$ be the set of all triangles computed by Algorithm 1 whose corresponding sample points lie in γ-Γ and let $\widetilde{\mathcal{M}}^\gamma_\Gamma$ be the underlying subspace. Since all objects considered are compact and since the triangles constructed by our algorithm fulfill the "small triangle property" we can prove that no point in any of the triangles in $\widetilde{M}^{\arcsin\left(\varepsilon^{1/3}\right)}_\Gamma$ is too far away from the medial axis M_Γ.

Lemma 5 *For each point $x \in \widetilde{\mathcal{M}}^{\arcsin\left(\varepsilon^{1/3}\right)}_\Gamma$, there exists some medial point $m \in M_\Gamma$ such that $|xm| \leqslant \mathcal{O}\left(\varepsilon^{1/3}\right) \cdot \rho$ where ρ is the medial radius of m.*

We conclude that the Hausdorff distance of $\widetilde{M}^{\arcsin\left(\varepsilon^{1/3}\right)}_\Gamma$ and M_Γ converges to zero:

Theorem 6 $\lim_{\varepsilon\to 0}\left(\widetilde{\mathcal{M}}^{\arcsin\left(\varepsilon^{1/3}\right)}_\Gamma\right) = M_\Gamma$.

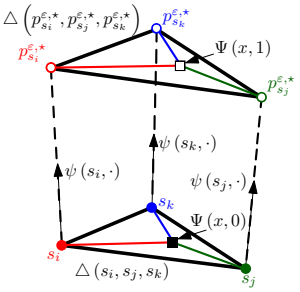
5 Topological Guarantee

In this section, we establish a homotopy between a parameterization of our approximation $\widetilde{\mathcal{M}}_\Gamma$ (the union of all triangles in $\widetilde{M}_\Gamma$) and a parameterization of the medial axis M_Γ. More precisely, since the outer medial axis may be disconnected, we establish a homotopy between parameterizations of $\widetilde{\mathcal{M}}^+_\Gamma$ and M^+_Γ; the proof for the isolated components of $\widetilde{\mathcal{M}}^-_\Gamma$ and corresponding components of M^-_Γ is similar. Due to space constraints, we only sketch the proof idea.

The parameterizations f and $\tilde{f}$ that we show to be homotopic are constructed as follows: we define $f : \Gamma \to M^+_\Gamma$ to be the function that maps each point $x \in \Gamma$ to its unique medial point on M^+_Γ; this function is easily seen to be continuous.

For the construction of $\tilde{f}$, we first observe that the output N of the CoCone-algorithm from which we construct $\widetilde{M}_\Gamma$ is homeomorphic to Γ – see Theorem 3. The function $\tilde{f}$ maps a point $x \in \Gamma$ to a unique point in $\widetilde{\mathcal{M}}^+_\Gamma$. For this, $\tilde{f}$ first uses the homeomorphism induced by Theorem 3 to map x to a unique point $n_x \in N$ and then maps n_x to a unique point $\tilde{n}_x$ in $\widetilde{\mathcal{M}}^+_\Gamma$ as described next.

The approximation of the medial axis is built on the set $\{p^{\varepsilon,+}_s, p^{\varepsilon,-}_s\}_{s\in S}$ of *approximated* poles where each approximated pole (if it exists) lies in the interior of $Vor(s)$. To establish a bijection between the sample points and the inner approximated poles and thus also between the set $\widetilde{M}^+_\Gamma$ of triangles in the approximation and the set F_N of triangles in the output of CoCone, we can prove that for each sample point there exists exactly one approximated inner pole point.



We can naturally associate each vertex of $\triangle(s_i, s_j, s_k)$ with a unique vertex of $\triangle(p^{\varepsilon,+}_{s_i}, p^{\varepsilon,+}_{s_j}, p^{\varepsilon,+}_{s_k})$ by the interpolating mapping $\psi : \{s_i, s_j, s_k\} \times [0,1] \to \mathbb{R}^3, (x, r) \mapsto x + r \cdot \overrightarrow{x p^{\varepsilon,+}_x}$. Obviously, $\psi(x, 0) = x$ and $\psi(x, 1) = p^{\varepsilon,+}_x$. Since every point $x \in \triangle(s_i, s_j, s_k)$ can be represented uniquely by a convex linear combination of s_i, s_j, and s_k (and since ψ is continuous and has a continuous inverse), we can extend the domain of ψ to $\triangle(s_i, s_j, s_k)$. This mapping between

two triangles can be extended to the sets F_N and $\widetilde{M}_\Gamma^+$ of triangles and thus to a function $\Psi : N \to \widetilde{\mathcal{M}}_\Gamma^+$.

Lemma 7 *The function $\Psi : N \to \widetilde{\mathcal{M}}_\Gamma^+$ is continuous.*

Finally, we concatenate Ψ and the homeomorphism between Γ and N (see Theorem 3); this yields the desired function $\tilde{f} : \Gamma \to \widetilde{\mathcal{M}}_\Gamma^+$ which, as the concatenation of two continuous functions, is continuous.

Based upon this construction we define a homeomorphism $H : [0,1] \times \Gamma \to \mathbb{R}^3$ via:

$$(r,x) \mapsto \begin{cases} x + 2 \cdot (0.5 - r) \cdot \overrightarrow{x\tilde{f}(x)} & r \in [0, 0.5[\\ x + 2 \cdot r \cdot \overrightarrow{xf(x)} & r \in [0.5, 1] \end{cases}$$

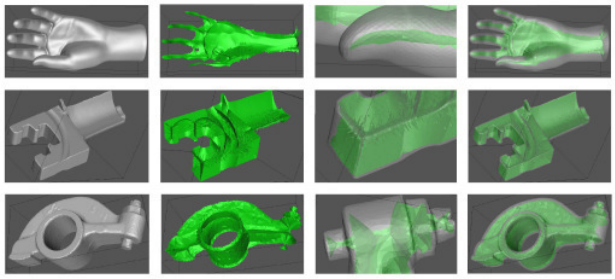
Lemma 8 *H is a homotopy between the parameterization $\tilde{f}$ of $\widetilde{\mathcal{M}}_\Gamma^+$ as constructed by Algorithm 1 and the parameterization f of the medial axis M_Γ^+.*

Theorem 9 *For a manifold Γ in $\mathbb{R}^3$ given by an ε-sample S, one can construct an object of size $\Theta(|S|)$ that converges to M_Γ as $\varepsilon \to 0$ and establish a homotopy between parameterizations of both structures.*

We point out that our algorithm consists of one invocation of the CoCone-algorithm and a linear-time postprocessing of this algorithm's output to approximate the tangent planes, to compute the approximations of the poles, and to construct the triangles.

6 Experimental Evaluation

To assess the quality of our approximation algorithm, we ran an implementation of our approach on a variety of realistic point samples.



The above figure shows that the algorithm constructs an approximation of the medial axis but unfortunately introduces a non-trivial amount of noise in some parts of the models. This noise appears to be of the same type as in some untuned versions of the algorithm of Dey and Zhao [5, p. 184/186].

7 Conclusions

We have presented an algorithm for approximating the medial axis of a manifold Γ in $\mathbb{R}^3$ based upon an ε-sample taken from Γ. Our algorithm can be seen as a postprocessing step of the CoCone-algorithm and does not add any asymptotic complexity to this algorithm's running time. Based upon the properties of the CoCone-algorithm, we can prove topological and convergence guarantees for the output of our algorithm. To the best of our knowledge, this algorithm is the first algorithm that guarantees these properties while at the same time producing an approximation of linear size. Furthermore, our algorithm is rather simple and – given an implementation of CoCone – thus easy to implement.

Acknowledgements The authors thank an anonymous reviewer for pointing out mistakes in an earlier version of this paper and for bringing reference [7] to our attention as well as Tamal K. Dey for providing, on behalf of all contributors, access to the implementation of the CoCone-algorithm and the algorithm of Dey and Zhao. The data sets used in the evaluation have been provided by AIM@SHAPE Shape Repository (`http://shapes.aimatshape.net`): Blade (courtesy of IMATI and CNR/Marco Attene and Corrado Pizzi) Hand (courtesy of INRIA), Rocker-Arm (courtesy of INRIA).

References

[1] N. Amenta and M. Bern. Surface reconstruction by Voronoi filtering. *Discrete & Computational Geometry*, 22(4):481–504, Dec. 1999.

[2] N. Amenta, M. Bern, and D. Eppstein. The crust and the β-skeleton: Combinatorial curve reconstruction. *Graphical Models and Image Processing*, 60(2):125–135, 1998.

[3] N. Amenta, S. Choi, T. K. Dey, and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. *International Journal of Computational Geometry and Applications*, 12(1–2):125–141, February & April 2002.

[4] N. Amenta, S. Choi, and R. K. Kolluri. The power crust, union of balls, and the medial axis transform. *Computational Geometry: Theory and Applications*, 19(2–3):127–153, July 2001.

[5] T. K. Dey and W. Zhao. Approximating the medial axis from the Voronoi diagram with a convergence guarantee. *Algorithmica*, 38(1):179–200, Oct. 2003.

[6] S. Funke and E. A. Ramos. Smooth-surface reconstruction in near-linear time. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 781–790. 2002.

[7] M. Hisada, A. G. Belyaev, and T. L. Kunii. A skeleton-based approach for detection of perceptually salient features on polygonal surfaces. *Computer Graphics Forum*, 21(4):689–700, 2002.

[8] J. Ruppert. A new and simple algorithm for quality two-dimensional mesh generation. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 83–92. 1993.

Approximating Tverberg Points in Linear Time for Any Fixed Dimension

Wolfgang Mulzer* Daniel Werner†‡

Abstract

Let P be a d-dimensional n-point set. A *Tverberg*-partition of P is a partition of P into r sets $P_1, \ldots, P_r$ such that the convex hulls $\operatorname{conv}(P_1), \ldots, \operatorname{conv}(P_r)$ have non-empty intersection. A point in $\bigcap_{i=1}^{r} \operatorname{conv}(P_i)$ is called a *Tverberg point* of depth r for P. A classic result by Tverberg implies that there always exists a Tverberg partition of size $\lceil n/(d+1) \rceil$, but it is not known how to find such a partition in polynomial time. Therefore, approximate solutions are of interest.

We describe a deterministic algorithm that finds a Tverberg partition of size $n/4(d+1)^3$ in time $d^{O(\log d)}n$. This means that for every fixed dimension we can compute an approximate Tverberg point (and hence also an approximate *centerpoint*) in *linear* time. Our algorithm is obtained by combining a novel lifting approach with a recent result by Miller and Sheehy [8].

1 Introduction

Let $P \subseteq \mathbb{R}^d$ be a d-dimensional point set with n points. In many applications (such as statistical analysis or mesh generation) we would like to have a way to generalize the one-dimensional notion of a median to the high-dimensional point set P. A very natural means to accomplish this are *centerpoints*: a point $c \in \mathbb{R}^d$ is a centerpoint for P if every halfspace that contains c meets P in at least $n/(d+1)$ points. A classic result in discrete geometry, the centerpoint theorem, shows that there exists a centerpoint for every point set [5, 9].

However, if we actually would like to compute a centerpoint for a given point set, the situation becomes more involved. Helly's theorem implies that the set of all centerpoints is given by the intersection of $O(n^d)$ halfspaces [6], so we can find a centerpoint in $O(n^d)$ time through linear programming. Chan [1] shows how to improve this running time to $O(n^{d-1})$ steps in expectation. He actually solves the harder problem of finding a point with maximum *Tukey depth*. The Tukey depth of a point c' is defined as the minimum number of points in P that are met by any halfspace containing c'. If the dimension is not fixed, Teng shows that it is co-NP-hard to check whether a given point is a centerpoint [10].

Since a running time of $O(n^{d-1})$ is not feasible for large d, it makes sense to look for faster approximate solutions. A classic approach uses ε-approximations [2]: in order to obtain a point of Tukey depth $n(1/(d+1) - \varepsilon)$ take a random sample $A \subseteq P$ of size $O((d/\varepsilon^2)\log(d/\varepsilon))$ and compute a centerpoint for A, using the linear-programming method. This gives the desired approximation with constant probability, and the resulting running time after the sampling step is constant. What more could we possibly wish for? For one, the algorithm is Monte-Carlo: with a certain probability, the reported point fails to be a centerpoint, and we know of no fast algorithm to check its validity. This problem can be solved by constructing the ε-approximation deterministically [2], at the expense of a more complicated algorithm. Nonetheless, in either case the resulting running time, though constant, still grows exponentially with d, an undesirable feature for large dimensions.

This situation motivated Clarkson et al. [3] to look for more efficient randomized algorithms for approximate centerpoints. They give a simple probabilistic algorithm that computes a point of Tukey depth $O(n/(d+1)^2)$ in time $O(d^2(d\log n + \log(1/\delta))^{\log(d+2)})$, where δ is the error probability. They also describe a more sophisticated algorithm that finds such a point in time polynomial in n, d, and $\log(1/\delta)$. Both algorithms are based on a repeated algorithmic application of Radon's theorem (see below). Unfortunately, there remains a probability of δ that the result is not correct, and we do not know how to detect a failure efficiently.

More than ten years later, Miller and Sheehy [8] launched a new attack at the problem. Their goal is to develop a deterministic algorithm for approximating centerpoints whose running time is subexponential in the dimension. The resulting algorithm is deterministic and runs in time $n^{O(\log d)}$. At the same time, it is the first algorithm that also finds an approximate *Tverberg partition* of P. The running time is subexponential in d, but it is still the case that n depends exponentially on $\log d$.

*Institut für Informatik, Freie Universität Berlin, Germany mulzer@inf.fu-berlin.de

†Institut für Informatik, Freie Universität Berlin, Germany dwerner@mi.fu-berlin.de

‡This research was funded by Deutsche Forschungsgemeinschaft within the Research Training Group (Graduiertenkolleg) "Methods for Discrete Structures"

In this paper, we show that the running time for finding approximate Tverberg partitions (and hence approximate centerpoints) can be improved. In particular, we show how to find a Tverberg partition for P that contains $n/4(d+1)^3$ sets in deterministic time $d^{O(\log d)}n$. This is linear in n for any fixed dimension, and the dependence on d is only quasipolynomial.

1.1 Some discrete geometry

We begin by recalling some basic facts and definitions from discrete geometry [7].

Theorem 1 (Radon's theorem) *For every set $P \subseteq \mathbb{R}^d$ with $d+2$ points there exists a partition (P_1, P_2) of P such that $conv(P_1) \cap conv(P_2) \neq \emptyset$.*

As mentioned above, Tverberg [11] generalized this theorem for larger point sets.

Theorem 2 (Tverberg's theorem) *Every set $P \subseteq \mathbb{R}^d$ with $n = (r-1)(d+1)+1$ points can be partitioned into r sets $P_1, \ldots, P_r$ such that $\bigcap_{i=1}^{r} conv(P_i) \neq \emptyset$.*

Let P be a set of n points in $\mathbb{R}^d$. We say that $x \in \mathbb{R}^d$ has *Tverberg depth* r (with respect to P) if there is a partition of P into sets $P_1, \ldots, P_r$ such that $x \in \bigcap_{i=1}^{r} \mathrm{conv}(P_i)$. Tverberg's theorem thus states that, for any set P in $\mathbb{R}^d$, there is a point of Tverberg depth at least $\lfloor (n-1)/(d+1)+1 \rfloor = \lceil n/(d+1) \rceil$. Note that every point with Tverberg depth r also has Tukey depth r. Thus, from now on we will use the term *depth* as a shorthand for Tverberg depth. As remarked above, Tverberg's theorem immediately implies the famous centerpoint theorem:

Theorem 3 (Centerpoint theorem) *For any set P of n points in $\mathbb{R}^d$ there is a point c such that all half-spaces containing c contain at least $\lceil n/(d+1) \rceil$ points from P.*

By Carathéodory's theorem, in order to describe a Tverberg partition of depth r, we only need $r \cdot (d+1)$ points from P. This observation is also used by Miller and Sheehy [8]. They also observe that replacing $d+2$ by $d+1$ points can be done in $O(d^3)$ time by using Gaussian elimination. We denote the process of replacing larger sets by sets of size $d+1$ as *pruning*. A partition for which each set consists of $d+1$ points is called a *pruned partition*.

1.2 Our contribution

In Section 2, we present a simple lifting argument which leads to an easy Tverberg approximation algorithm.

Theorem 4 *Let P be a set of n points in $\mathbb{R}^d$ in general position. One can compute a Tverberg point of depth $n/2^d$ for P and the corresponding partition in time $O(n)$.*

While this does not yet give a good approximation ratio (though constant for any fixed d), it is a very natural approach to the problem: it computes a higher dimensional Tverberg point via successive median partitions — just as a Tverberg point is a higher dimensional generalization of the 1-dimensional median.

By collecting several low-depth points and applying the brute-force algorithm on small point sets, we get an even higher depth in linear time for any *fixed* dimension:

Theorem 5 *Let P be a set of n points in $\mathbb{R}^d$. Then one can compute a Tverberg point of depth $n/2(d+1)^2$ and a corresponding partition in time $f(2^d) + d^{O(1)}n$, where $f(m) = O(m!)$ is the time for computing a Tverberg point of depth $m/(d+1)$ for m points brute force.*

Finally, by combining our approach with that of Miller and Sheehy, we improve our algorithm to have a running time which is quasipolynomial in d:

Theorem 6 *Let P be a set of n points in $\mathbb{R}^d$. Then one can compute a Tverberg point of depth $n/4(d+1)^3$ and a corresponding partition in time $d^{O(\log d)} \cdot n$.*

2 A simple fixed parameter algorithm

A natural way to compute a Tverberg point for $P \subseteq \mathbb{R}^d$ is to first project P to some lower-dimensional space, then to recursively compute a good Tverberg point for this projection, and to use this point to find a solution in the higher-dimensional space. Surprisingly, we are not aware of any argument along these lines having appeared in the literature so far.

In what follows, we will describe how to *lift* a lower-dimensional Tverberg point into some higher dimension. Unfortunately, this process will come at the cost of a decreased depth for the lifted Tverberg point. For clarity of presentation, we first explain the lifting lemma in its simplest form. In Section 3, we then state the lemma in its full generality.

Lemma 7 *Let P be a set of n points in $\mathbb{R}^d$, and let h be a hyperplane in $\mathbb{R}^d$. Let $c' \in h$ be a Tverberg point of depth r for the projection of P onto h, and suppose we know a corresponding Tverberg partition. Then we can find a Tverberg point $c \in \mathbb{R}^d$ of depth $\lceil r/2 \rceil$ for P and a corresponding Tverberg partition in time $O(n)$.*

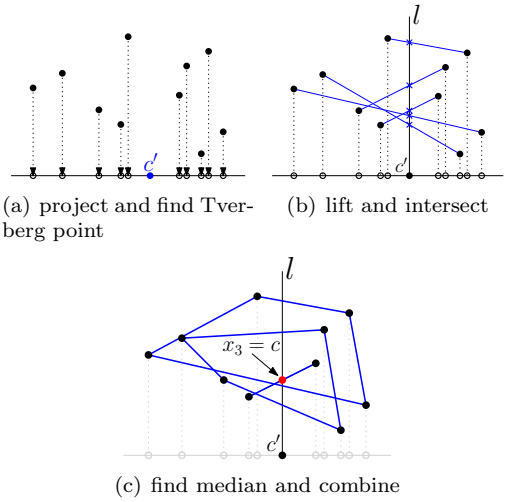
Proof. For every point $p \in P$, let $\mathrm{pr}(p)$ denote the projection of p onto h. Let $P_1, \ldots, P_r \subseteq P$ such that $\mathrm{pr}(P_1), \ldots, \mathrm{pr}(P_r)$ is a Tverberg partition for $\mathrm{pr}(P)$ with Tverberg point c'. Let ℓ be the line orthogonal to h that passes through c'.

Since our assumption implies $c' \in \mathrm{conv}(\mathrm{pr}(P_i))$ for $i = 1, \ldots, r$, it follows that ℓ intersects each $\mathrm{conv}(P_i)$ at some point x_i. Let $\widehat{Q}_i = \{x_{i_1}, x_{i_2}\}$, $i = 1, \ldots, \lceil r/2 \rceil$, be a Tverberg partition of $x_1, \ldots, x_r$. (If r is odd, one of the sets contains only one point, the median.) Since the points x_i lie on the line ℓ, such a Tverberg partition exists and can be computed in time $O(r)$ by finding the median c, i.e., the element of rank $\lceil r/2 \rceil$, according to the order along ℓ [4].

We claim that c is a Tverberg point for P of depth $\lceil r/2 \rceil$. Indeed, we have

$$c \in \mathrm{conv}(\widehat{Q}_i) = \mathrm{conv}(\{x_{i_1}, x_{i_2}\}) \subset \mathrm{conv}(P_{i_1} \cup P_{i_2}),$$

for $1 \le i \le \lceil r/2 \rceil$. Thus, if we set $Q_i := P_{i_1} \cup P_{i_2}$, then $Q_1, \ldots, Q_{\lceil r/2 \rceil}$ is a Tverberg partition for c. The total time to find c and the Q_i is $O(n)$, as claimed. See Figure 1 for a two-dimensional illustration of the lifting argument. □



(a) project and find Tverberg point (b) lift and intersect

(c) find median and combine

Figure 1: Illustrating the lifting lemma in the plane.

Proof. [of Theorem 4] We apply Lemma 7 recursively on the dimension, pruning the partitions after each lifting. This leads to an $O(n)$ running time. □

2.1 Improving the approximation factor

In order to improve the approximation factor, we will use an easy lemma to improve the Tverberg depth by collecting and combining several lower-depth points.

Lemma 8 *Let $P \subseteq \mathbb{R}^d$, $|P| = n$. If we can find a Tverberg point of depth n/ρ and the corresponding pruned partition in time $f(n,d)$, then we can find $\rho/(d+1)$ points of depth $n/2\rho$ in time $O\left(\frac{\rho}{(d+1)}\left(p(n,d) + f(n,d)\right)\right)$, where $p(n,d)$ is the time for the pruning phase.*

By Theorem 4 we can find a point of depth $n/2^d$ and a corresponding pruned partition in time $O(n)$. Thus, by applying Lemma 8 with $\rho = 2^d$, we can also find $2^d/(d+1)$ points of depth $n/2^{d+1}$ in linear time.

In order to make use of Lemma 8, we will need a lemma that states that sometimes by *combining* Tverberg points we can multiply their depth. This generalizes a similar lemma by Miller and Sheehy [8, Lemma 4.1].

Lemma 9 *Let P be a set of n points in $\mathbb{R}^d$, and let $P = \biguplus_{i=1}^{k} P_i$ be a partition of P. Furthermore, suppose that for each P_i we have a Tverberg point $c_i \in \mathbb{R}^d$ of depth r, together with a corresponding pruned Tverberg partition $\mathcal{P}_i$. Let $C := \{c_i \mid 1 \le i \le k\}$ and c be a point of depth r' for C, with corresponding pruned Tverberg partition $\mathcal{C}$. Then c is a point of depth $r \cdot r'$ for P. Furthermore, we can find a corresponding pruned Tverberg partition in time $d^{O(1)}n$.*

Theorem 5 then follows by combining Theorem 4 with Lemmas 8 and 9.

3 An improved algorithm

Finally we show how to improve our approach through a more sophisticated recursion and obtain an algorithm with running time $d^{O(\log d)} \cdot n$ for the price of losing a depth factor of $1/2(d+1)$. First, however, we describe the more general version of Lemma 7 we promised above.

Let $P \subseteq \mathbb{R}^d$. Recall that a k-dimensional *flat* $F \subseteq \mathbb{R}^d$ (often abbreviated as *k-flat*) is defined as a k-dimensional affine subspace of $\mathbb{R}^d$. A k-dimensional flat $F \subseteq \mathbb{R}^d$ is called a *Tverberg k-flat* of depth r for P, if there is a partition of P into sets $P_1, \ldots, P_r$ such that $\mathrm{conv}(P_i) \cap F \neq \emptyset$ for all $i = 1, \ldots, r$.

Lemma 10 *Let P be a set of n points in $\mathbb{R}^d$, and let $h \subseteq \mathbb{R}^d$ be a k-flat. Suppose we have a Tverberg point c of depth r for $\mathrm{pr}(P)$, as well as a corresponding Tverberg partition. Let $h_c^{\perp}$ be the $(d-k)$ flat orthogonal to h that passes through c. Then $h_c^{\perp}$ is a Tverberg $(d-k)$-flat for P of depth r, with the same Tverberg partition.*

First of all, this shows how a good algorithm for any fixed dimension improves the general case:

Corollary 11 *Let $\delta \ge 1$ be a fixed integer. Suppose we have an algorithm $\mathcal{A}$ with the following property: for every point set $Q \subseteq \mathbb{R}^\delta$, algorithm $\mathcal{A}$ constructs a Tverberg point of depth $|Q|/\rho$ for Q as well as a corresponding pruned Tverberg partition in time $f(|Q|)$.*

Then, for any n-point set $P \subseteq \mathbb{R}^d$ and for any $d \geq \delta$, we can find a Tverberg point of depth $n/\rho^{d/\delta}$ and a corresponding pruned partition in time $\lceil d/\delta \rceil f(n) + d^{O(1)}n$.

Proof. Induction on $k := \lceil d/\delta \rceil$ □

The idea of the new algorithm then is as follows: using Corollary 11, we reduce solving a d-dimensional instance to solving two instances of dimension $d/2$. This can be done recursively. Unfortunately, applying Corollary 11 reduces the depth of the partition. To fix this, we apply Lemmas 8, 9 and the Miller-Sheehy algorithm to increase the depth again. Details follow.

Proof. [of Theorem 6] We prove the theorem by induction on d. For $d = 1$ the claim is immediate: in this case the problem reduces to median computation.

Thus, suppose that $d > 1$. By induction, for any $(d/2)$-dimensional point set $Q \subseteq \mathbb{R}^{d/2}$ there exists an algorithm that returns a Tverberg point of depth $|Q|/4(d/2+1)^3$ and a corresponding pruned Tverberg partition in time $d^{\alpha \log(d/2)}n$, for some sufficiently large constant $\alpha > 0$.

Thus, by Corollary 11 (with $\delta = d/2$), there exists an algorithm that finds a Tverberg point for P of depth $n/16(d/2+1)^6$ and a corresponding Tverberg partition in time $2d^{\alpha \log(d/2)} + d^{O(1)}n$.

Now a more general version of Lemma 8 can be used to show that we can find $16(d/2+1)^6/(d+1)$ points of depth $n/32(d/2+1)^6$ and corresponding (disjoint) pruned partitions in time $d^{\alpha \log(d/2)+O(1)} \cdot n$.

Let C be the set of these Tverberg points. Applying the Miller-Sheehy algorithm, we can find a Tverberg point for C of depth $|C|/2(d+1)^2$ and a corresponding pruned Tverberg partition in time $|C|^{O(\log d)}$. Now, Lemma 9 shows that in additional $d^{O(1)}n$ time, we obtain a Tverberg point and a corresponding Tverberg partition for P of size

$$\frac{n}{2 \cdot 16(d/2+1)^6} \cdot \frac{16(d/2+1)^6}{2(d+1)^2 \cdot (d+1)} = \frac{n}{4(d+1)^3},$$

as desired.

It remains to analyze the running time. Adding up the various terms, we end up with a time bound of

$$T(n,d) = d^{\alpha \log(d/2)+O(1)}n + |C|^{O(\log d)} + d^{O(1)}n.$$

Since $|C| = d^{O(1)}$, for α large enough $T(n,d) \leq d^{\alpha \log d}n = d^{O(\log d)}n$, as claimed.

□

4 Conclusion and Outlook

We have presented a very simple algorithm for finding an approximate Tverberg point, which runs in linear time for any fixed dimension. Using more sophisticated methods and combing our approach with known results, we managed to improve the running time to $d^{O(\log d)} \cdot n$, while getting within a factor of $1/4(d+1)^2$ of the guaranteed optimum.

Unfortunately, the resulting running time is still quasi-polynomial in d, and we still do not know whether there exists a polynomial algorithm (in n and d) for finding an approximate Tverberg point. However, we are hopeful that our techniques constitute a further step in this direction and that such an algorithm will eventually be discovered.

Acknowledgments. We would like to thank Nabil Mustafa for suggesting the problem to us. We also thank him and Don Sheehy for helpful discussions and the anonymous reviewers for their helpful comments.

References

[1] T. M. Chan. An optimal randomized algorithm for maximum Tukey depth. In *Proc. 15th Annu. ACM-SIAM Sympos. Discrete Algorithms (SODA)*, pages 430–436, 2004.

[2] B. Chazelle. *The discrepancy method: randomness and complexity*. Cambridge University Press, Cambridge, 2000.

[3] K. L. Clarkson, D. Eppstein, G. L. Miller, C. Sturtivant, and S.-H. Teng. Approximating center points with iterated Radon points. *Internat. J. Comput. Geom. Appl.*, 6(3):357–377, 1996.

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT Press, Cambridge, MA, third edition, 2009.

[5] L. Danzer, B. Grünbaum, and V. Klee. Helly's theorem and its relatives. In *Proc. Sympos. Pure Math., Vol. VII*, pages 101–180. Amer. Math. Soc., Providence, R.I., 1963.

[6] H. Edelsbrunner. *Algorithms in combinatorial geometry*. Springer-Verlag, Berlin, 1987.

[7] J. Matoušek. *Lectures on Discrete Geometry*. Springer, 2002.

[8] G. L. Miller and D. R. Sheehy. Approximate centerpoints with proofs. *Comput. Geom. Theory Appl.*, 43(8):647–654, 2010.

[9] R. Rado. A theorem on general measure. *J. London Math. Soc.*, 21:291–300, 1946.

[10] S.-H. Teng. *Points, spheres, and separators: a unified geometric approach to graph partitioning*. PhD thesis, School of Computer Science, Carnegie Mellon University, 1992.

[11] H. Tverberg. A generalization of Radon's theorem. *J. London Math. Soc.*, 41:123–128, 1966.

All-maximum and all-minimum problems under some measures

Asish Mukhopadhyay *† Satish Ch. Panigrahi *

1 Introduction

In computational geometry, an oft-recurring problem is to identify subsets of a point set having desirable properties. The following type of proximity problems belong to this genre: for each point p_i in a planar point-set $P = \{p_1, p_2, p_3, \ldots, p_n\}$, find a pair $\{p_j, p_k\}$, $i \neq j \neq k$, such that a measure $\mathcal{M}$ defined on the points $\{p_i, p_j, p_k\}$ is maximized or minimized. The cases where $\mathcal{M}$ is the distance from p_i to the segment or line defined by $\{p_j, p_k\}$ have been extensively studied (see [6],[7],[12]). In this paper, we study a number of other measures in two and higher dimensions.

We summarize our $2d$ results in the following table. Extensions to higher dimensions are discussed in the relevant sections.

Measure	Maximum	Minimum
Sum	$O(n \log n)$	$O(n \log n)$
Product	$O(n \log n)$	$O(n \log n)$
Difference	$O(n \log n)$	$O(n^2 \log n)$
Line-distance	$O(n^2)$	$O(n^2)$
$\triangle$ Area	$O(nh)$	$O(n^2)$
$\triangle$ Perimeter	$O(nh)$	$O(n(n \log n + max_i(\phi_i)^2))$
Circumradius	$O(n^2 \log n)$	$O(n^2 \log n)$

2 Sum and Product Measures

The *sum* $\mathcal{S}(p_i, p_j, p_k)$ and *product* $\mathcal{P}(p_i, p_j, p_k)$ measures are respectively $|\overline{p_i p_j}| + |\overline{p_i p_k}|$ and $|\overline{p_i p_j}| * |\overline{p_i p_k}|$.

Claim 1 *For a point $p_i \in P$, $\mathcal{S}(p_i, p_j, p_k)$ and $\mathcal{P}(p_i, p_j, p_k)$ is maximum (minimum) when $p_j, p_k \in P - \{p_i\}$, realize the farthest (nearest) and second farthest (second nearest) distance from the point p_i.*

An $O(n^2)$ algorithm for the all-minimum as well as all-maximum in both measures is immediate. For an $O(n \log n)$ algorithm we construct the third order nearest and farthest Voronoi diagrams on P and $O(n \log n)$ point location structures on both.

3 Difference Measure

The *difference* measure is $\mathcal{D}(p_i, p_j, p_k) = ||\overline{p_i p_j}| - |\overline{p_i p_k}||$.

Claim 2 *For a point $p_i \in P$, the pair $\{p_j, p_k\} \in P - \{p_i\}$ realize the maximum $\mathcal{D}(p_i, p_j, p_k)$ iff p_j and p_k are respectively the nearest and farthest point from p_i or vice versa.*

An $O(n^2)$ algorithm for the all-maximum problem is immediate. For an $O(n \log n)$ algorithm we construct the nearest-point Voronoi diagram for the nearest point in $P - \{p_i\}$ for each p_i; for the farthest point of each p_i, we construct the farthest-point Voronoi diagram as well as a point location structure on top of this. This is because we need to locate p_i in the farthest-point Voronoi region of a point p_j it is farthest from. The $O(n^2)$ algorithm is easily extended to d-dimensions to run in $O(dn^2)$ time.

For a fixed p_i, we can determine the minimum $\mathcal{D}\{p_i, p_j, p_k\}$ in $O(n \log n)$ time by solving the closest-pair problem on the line. Thus the all-minimum problem can be solved in $O(n^2 \log n)$ time. This extends to an $O(dn^2 \log n)$ algorithm in d-dimensions. The all-minimum problem can be solved in expected $O(n^2)$ time, by solving the closest-pair problem in $O(n)$ expected time by a randomized algorithm, assuming floor and square-root operations can be done in constant time [11]. However, an $O(n^2)$ deterministic algorithm remains an interesting open question.

4 Line Distance Measure

The line-distance measure is $\mathcal{LD}(p_i, p_j, p_k) = d(p_i, \overline{p_j p_k})$, where $d(p, l)$ denotes the minimum distance from a point p to a line l. For a fixed p_i, Mount at al [6] gave an optimal $O(n \log n)$ time and $O(n)$ space algorithm to find the minimum value of $\mathcal{LD}(p_i, p_j, p_k)$. We give an $O(n^2)$ algorithm for the all-closest problem.

Let p_i occupy cell C in the arrangement of lines defined by all pairs of points in $P - \{p_i\}$. One of the bounding lines of this cell is closest to p_i.

For if a line closest to p_i is disjoint from C, the normal to this line from p_i crosses the boundary of C at a point p, say (see Fig. 1(a)). The line to which p belongs is closer, a contradiction. The search for the closest line is based on the following claim.

Claim 3 *In the dual plane, the points corresponding to the lines that bound the cell of p_i is part of the zone of the dual of p_i.*

*School of Computer Science, University of Windsor, {asishm, panigra}@uwindsor.ca

†Research supported by an NSERC grant

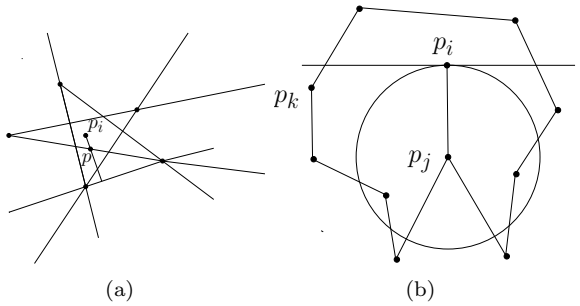


Figure 1: (a) *A closest line to* p_i (b) *A farthest line from* p_j*, incident on* p_i

As the zone is of linear size, the closest line to p_i can be found in linear time and thus in $O(n^2)$ time for all the points in P. Following Duffy et al [7], the problem can be shown to be n^2-hard by reduction from the problem of determining if 3 of n points in the plane are collinear. Also, the ideas can be extended to an $O(n^d)$ algorithm in d-dimensions.

For the *all-farthest* line distance problem. we follow an idea due to Duffy et al [7]. For an anchor point p_i, we find a line incident on it that is farthest from each p_j in $P - \{p_i\}$. This is updated vis-a-vis p_j as we consider the angular order about the remaining $n-2$ anchor points. Assume we know the sorted order of the points in the set $P - \{p_i\}$ around p_i.

Consider a circle $\mathcal{C}$ of radius $|p_ip_j|$ and center p_j. The tangent to $\mathcal{C}$ at p_i is a farthest line if another point p_k is incident on it; otherwise, we determine the smallest angle through which we must pivot the tangent around p_i to find a point p_k incident on it (see Fig. 1(b)). This is the farthest line from p_j that is incident on p_i. Since we might have to pivot clockwise or anti-clockwise, we do a clockwise sweep, followed by an anti-clockwise sweep. By careful bookkeeping, as we pivot around p_i we find the farthest line from each of the remaining points that is incident on p_i in $O(n)$ time.

We repeat the above steps by considering the angular order about the remaining $n-1$ points. Since we can determine the circular order of $P - \{p_i\}$ around each p_i in $O(n^2)$ time [13], [8], the time-complexity of the all-farthest problem in the line-distance metric is in $O(n^2)$.

It would be interesting to know (i) if the above algorithm is optimal in the algebraic decision tree model and (ii) if the algorithm extends to d-dimensions with time complexity in $O(n^d)$. For $d = 3$, we can get an $O(n^3 \log^2 n)$ algorithm using ideas from [3].

5 Triangle Area Measure

Let $\mathcal{A}(p_i, p_j, p_k)$ measure the area of the triangle formed by the points p_i, p_j and p_k. Below, we describe $O(n^2)$ time algorithms for both the all-maximum and all-minimum versions in this measure.

5.1 Maximum Area Triangle

The following claim gives a structural characterization of a maximum area triangle, anchored at p_i.

Claim 4 *$\mathcal{A}(p_i, p_j, p_k)$ is maximum when the points p_j and p_k lie on the boundary of the convex hull of P and p_j (resp. p_k) is farthest from the supporting line of $\overline{p_ip_k}$ (resp. $\overline{p_ip_j}$).*

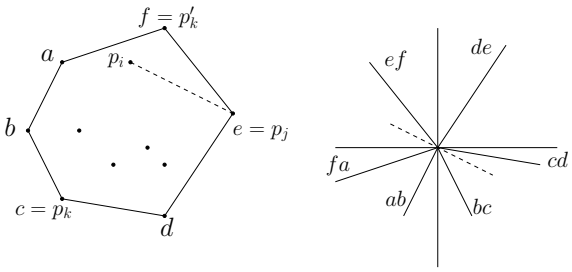


Figure 2: Convex hull and its corresponding ray diagram

An efficient algorithm for the all-maximum problem follows from Claim 4. In a preprocessing step, we construct the convex hull of P and then its ray diagram (see Fig. 2). For each p_i we linearly scan the hull boundary for p_j and p_k. If p_k is the farthest antipodal vertex corresponding to $\overline{p_ip_j}$, we tag p_k with the index j; when the scan reaches p_k, we determine its farthest antipodal vertex and use the tag attached to p_k to check if it is the same as p_j. If so, this is another candidate pair. We determine the area of $\triangle p_ip_jp_k$ and update the current maximum area triangle if necessary. Doing this for each p_i, gives an $O(n(h + \log h))$ algorithm for the all-maximum problem.

An $O(n^2)$ algorithm can also be got by dualization. For fixed p_i and p_j, $\triangle p_ip_jp_k$ has maximum area when p_k is farthest from the supporting line of p_ip_j. Thus we find the farthest p_k for each of $n-1$ different line segments $\overline{p_ip_j}$ for the maximum value of $\mathcal{A}(p_i, p_j, p_k)$. The dual version helps us solve our problem if for each intersection point, which corresponds to a point pair $\{p_i, p_j\}$ in the dual plane, we determine the line that is vertically farthest from this intersection point. This line will be part of the lower or upper envelope in the arrangement. The upper and lower envelopes can be obtained in $O(n \log n)$ from the convex hull of the points in the primal plane.

A topological sweep of the line-arrangement in the dual plane discovers the intersection points on each line in a left to right order. Thus we can determine the intersection of a vertical line through each intersection point on this line with the upper and lower

envelopes in a left to right order. We maintain an intersection history for each. This requires $O(n)$ time for each line and hence $O(n^2)$ time for all the lines. Once we have the farthest point for each line, we can determine the maximum area triangle for an anchored point by selecting from among the intersection points on the dual of p_i the one whose farthest line is the farthest of all. Since topological sweep can be done in $O(n^2)$ time, the time complexity of this alternate algorithm is in $O(n^2)$.

This algorithm generalizes to 3-dimensions, using a topological sweep algorithm to compute an arrangement of planes in 3-dimensions [9] to find a maximum volume tetrahedron for each p_i in $O(n^3)$ time.

5.2 Minimum Area Triangle

First this claim.

Claim 5 *For an anchor point p_i and a fixed p_j, if $\mathcal{A}(p_i, p_j, p_k)$ is minimum then p_k is vertically closest to the supporting line, ℓ, of p_i and p_j.*

Since dualization preserves vertical distances we can take advantage of the above characterization; in the dual plane a fixed p_i corresponds to a fixed line ${p_i}^*$, and for each p_j, the pair $\{p_i, p_j\}$ corresponds to an intersection on the line ${p_i}^*$. Thus for each intersection point we find a line ${p_k}^*$ that is vertically closest to it.

We simulate Chazelle et al's [5] construction of a line-arrangement with some additional book-keeping and some reasonable data structure (e.g doubly connected edge list) to represent the planar graph corresponding to the arrangement. To ensure that the addition of each new line to the current arrangement takes linear time, we assume that the arrangement lies inside a large bounding box. Since a line intersects the boundary of this bounding box twice, we can easily determine the entry face of the ith line ℓ_i of the arrangement. The arrangement can be updated by walking along the lower parts of zone(ℓ_i) (i.e. zone of ℓ_i) as shown in Fig. 3. As the combinatorial complexity of the faces of the arrangement that intersect ℓ_i is at most $8i$ [5], the walk along zone(ℓ_i) takes $O(i)$ time. While updating the data structure, we maintain the vertically closest line from each vertex.

Since computing the arrangement takes $O(n^2)$ time and $O(n^2)$ space, the same complexities hold for the all-minimum area triangle problem. This problem is n^2-hard since we can use this to settle if 3 of n points in the plane are collinear. The algorithm generalizes to d-dimensions to run in $O(n^d)$ time as the size of a zone is $O(n^{d-1})$ [10].

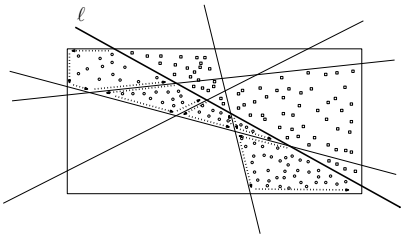


Figure 3: Walking the lower part of zone(ℓ)

6 Triangle Perimeter Measure

The triangle perimeter measure $\mathcal{P}(p_i, p_j, p_k)$ is defined to be $|\overline{p_ip_j}| + |\overline{p_jp_k}| + |\overline{p_ip_k}|$.

6.1 Maximum perimeter

We have the following characterization.

Claim 6 *For a fixed p_i, if $\mathcal{P}(p_i, p_j, p_k)$ is maximum then p_j and p_k are vertices on the convex hull, $CH(P)$, of P.*

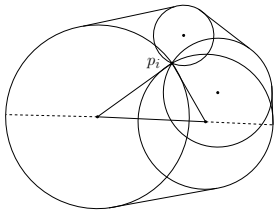


Figure 4: The diameter of the convex figure is equal to the maximum perimeter triangle rooted at p_i

For all the p_i's on the hull boundary, we find the maximum perimeter in $O(h)$ time by using the monotone matrix method of [1]. For an internal point p_i, we use an ingenious reduction due to Boyce et al [4] that reduces the problem to that of computing the diameter of a convex region bounded by circular arcs and segments that are common external tangents to two circles (see Fig. 4). This takes $O(h)$ time for a fixed p_i and thus $O((n-h)h)$ time for all the $n-h$ points inside the convex hull. Thus we have an $O(nh)$ algorithm for the all-maximum problem.

6.2 Minimum perimeter

We first discuss the problem of computing a minimum perimeter triangle anchored at a point p_i. Assign to each $p_j \in P - \{p_i\}$ a weight $w_j = |p_ip_j|$ where $|p_ip_j|$ is the Euclidean distance between p_i and p_j.

We have the following characterization for an anchored minimum perimeter triangle.

Claim 7 *If $\mathcal{P}_i$ is the perimeter of any triangle $\triangle p_ip_jp_k$, anchored at p_i, then neither p_j nor p_k can be at a distance farther than $\frac{\mathcal{P}_i}{2}$ from p_i.*

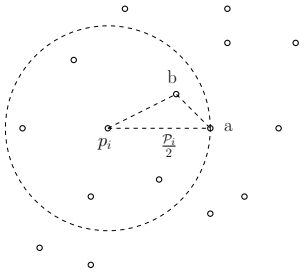


Figure 5: The perimeter of $\triangle p_i ab > \mathcal{P}_i$

Sort all the points by their assigned weights w_j. Let $\mathcal{P}_i$ be the perimeter of the triangle formed by p_i and two points p_j and p_k closest to it. By Claim 7, all the candidate vertices of a minimum perimeter triangle anchored at p_i lie on or inside the circle of radius $\frac{\mathcal{P}_i}{2}$ centered at p_i (see Fig. 5). Let $S(p_i,\ \mathcal{P}_i/2)$ be this set of points. If there exists a $p'_j \in S(p_i,\ \mathcal{P}_i/2)$ such that perimeter of $\triangle p_i p'_j p_k$ or $\triangle p_i p_j p'_j$ is less than $\mathcal{P}_i$ then set it as minimum perimeter of the triangle found so far. This process is repeated to get a minimum perimeter triangle anchored at p_i. If Δ_i be the smallest separation of a pair of adjacent distances, an upper bound on the number of points inside a circle of radius $\mathcal{P}_i/2$ is $\mathcal{P}_i/2\Delta_i$. Thus the running time of the above algorithm for all points is $O(n(n \log n + max_i(\mathcal{P}_i/2\Delta_i)^2))$.

7 Circumcircle radius measure

The circumcircle radius measure $\mathcal{R}(p_i, p_j, p_k)$ of $\{p_i, p_j, p_k\}$ is defined to be the radius of a circle that circumscribes the $\triangle p_i p_j p_k$. The combinatorial complexities of the farthest and nearest 2-point site Voronoi diagrams in this distance measure were left open by Barequet et al [2].

In our restricted scenario, by treating each fixed p_i as a center of inversion and inverting all other points in $P - \{p_i\}$ with respect to it, the problem reduces to finding a nearest and farthest line from p_i, spanned by pairs of points in the inverted set. Daescu et al [6] has given an $O(n \log n)$ solution for the nearest and farthest line problem for a fixed p_i. Using these algorithms, we can solve the all-maximum and all-minimum radius circle problems in $O(n^2 \log n)$ time.

In [3] Bespamyatnikh and Segal considered the problem of selecting a hyperplane spanned by d of n points in d-dimensional space the rank of whose distance from the origin is k. They showed that the 3-dimensional version of this problem is almost 3-SUM hard and proposed an $O(n^2 \log^2 n)$ algorithm. Thus using each p_i as the origin of coordinates we can determine the farthest and nearest planes spanned by 3 points in $P - \{p_i\}$ in the same time and hence solve the all-farthest and all-nearest problems in this measure in $O(n^3 \log^2 n)$ time. Thus by using inversion we can solve the all-minimum and all-maximum versions in the same time.

8 Conclusion

We have made an exhaustive study of a restricted kind of proximity problems, indicating possible open problems. A particular mention ought to be made of the minimum perimeter problem.

9 Acknowledgements

We thank one of the reviewers for a painstaking review. It was very helpful.

References

[1] A. Aggarwal and J. Park. Notes on searching in multidimensional monotone arrays. In *SFCS '88: Proceedings of the 29th Annual Symposium on Foundations of Computer Science*, pages 497–512, Washington, DC, USA, 1988. IEEE Computer Society.

[2] G. Barequet, M. T. Dickerson, and R. L. S. Drysdale. 2-point site Voronoi diagrams. *Discrete Appl. Math.*, 122(1-3):37–54, 2002.

[3] S. Bespamyatnikh and M. Segal. Selecting distances in arrangement of hyperplanes spanned by points. *Journal of Discrete Algorithms*, 2:333–345, 2004.

[4] J. E. Boyce, D. P. Dobkin, R. L. S. Drysdale, and L. J. Guibas. Finding extremal polygons. *SIAM J. Computing*, 14(1):134–147, 1985.

[5] B. Chazelle, L. J. Guibas, and D. T. Lee. The power of geometric duality. *BIT*, 25:76–90, 1985.

[6] O. Daescu, J. Luo, and D. M. Mount. Proximity problems on line segments spanned by points. *Computational Geometry: Theory and Applications*, 33:115–129, 2006.

[7] K. Duffy, C. McAloney, H. Meijer, and D. Rappaport. Closest segments. In *Proc. of CCCG 2005*, pages 229–231, 2005.

[8] H. Edelsbrunner and L. Guibas. Topologically sweeping an arrangement. *Journal of Computer and System Sciences*, 38(1):165–194, 1989.

[9] H. Edelsbrunner, J. O'Rourke, and R. Seidel. Constructing arrangements of lines and hyperplanes with applications. *SIAM J. Comput.*, 15:341–363, 1986.

[10] H. Edelsbrunner, M. Sharir, and R. Seidel. On the zone therem for hyperplane arrangements. *SIAM J. Comput.*, 22(2):418–429, 1993.

[11] S. Khuller and Y. Matias. A simple randomized sieve algorithm for the closest-pair problem. *Information and Computation*, 118:34–37, 1995.

[12] A. Mukhopadhyay, S. Chatterjee, and B. Lafreniere. On the all-farthest-segments problem for a planar set of points. *Inf. Process. Lett.*, 100:120–123, 2006.

[13] M. Overmars and E. Welzl. New methods for computing visibility graphs. In *Proceedings of the fourth Annual Symposium on Computational Geometry*, pages 164–171, 1988.

Homotopic $\mathcal{C}$-oriented Routing

Kevin Verbeek*

Abstract

We study the problem of finding non-crossing $\mathcal{C}$-oriented paths that use as few links as possible and are homotopic to a set of input paths in an environment with $\mathcal{C}$-oriented obstacles. We introduce a special type of $\mathcal{C}$-oriented paths—*smooth paths*—and present a 2-approximation algorithm that runs in $O(n^2(n + \log \kappa) + k_{in} \log n)$ time, where n is the total number of paths and obstacle vertices, k_{in} is the total number of links in the input, and $\kappa = |\mathcal{C}|$. The algorithm also computes an $O(\kappa)$-approximation for general $\mathcal{C}$-oriented paths. As a related result we show that, given a set of (possibly crossing) $\mathcal{C}$-oriented paths with a total of L links, non-crossing $\mathcal{C}$-oriented paths homotopic to the input paths can require a total of $\Omega(L \log \kappa)$ links.

1 Introduction

Motivation. Schematic maps are an important tool in the field of cartography; they are used to visualize networks (like metro or road networks) in a highly simplified form, omitting details that are not relevant to the information the map is intended to convey. Edges of a schematic map are often drawn using few orientations (commonly rectilinear or octilinear) and with as few links as possible. Some schematic maps (like metro maps) can deviate significantly from the geographic context, retaining only the relative positions and topology of the network. Other schematic maps have to obey certain geometric restrictions imposed by the geographic context. For example, if a schematic road network is used as a thematic overlay for a geographic map, the cities must have the same positions in both maps. Fig. 1 illustrates what we want to achieve: given a set of embedded vertices (cities), edges (roads), and obstacles (landmarks), redraw the edges with fixed orientations and few links while retaining the vertex positions and the topology of the network to preserve recognizability.

Our input consists of n_p non-crossing polygonal paths and a set of polygonal obstacles with a total of n_v vertices. We are also given a set $\mathcal{C}$ of orientations. A path or obstacle is $\mathcal{C}$-oriented if every line segment, or *link*, is parallel to some orientation $c \in \mathcal{C}$. Our goal is to compute non-crossing $\mathcal{C}$-oriented paths that are homotopic to the input paths using as few links as possible. Two paths π and π' with the same endpoints are homotopic (notation $\pi \sim_h \pi'$) if π can be "continuously deformed" into π' without passing over any of the obstacles (see e.g. [4] for a formal definition). We assume that the obstacles are $\mathcal{C}$-oriented and that the endpoints of the paths are considered as obstacles as well when considering homotopy. We refer to this problem as the *$\mathcal{C}$-oriented routing problem.*

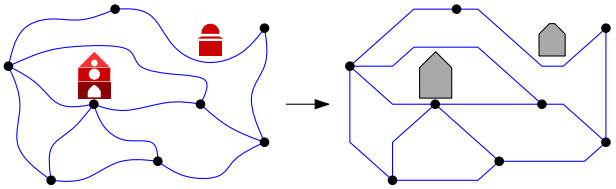


Figure 1: Input and output.

Related Work. Cabello *et al.* [2] give an algorithm that schematizes a network using 2 or 3 links per path, if possible. Nöllenburg and Wolff [11] use a method based on mixed-integer programming to generate metro maps using one edge per path. Both methods do not include obstacles and are restricted to a small number of links per path. Neyer [10] and Merrick and Gudmundsson [9] give algorithms for simplifying paths with fixed orientations. The simplified path can deviate at most a distance ϵ from a given input path. Both methods consider only one path at a time and cannot give guarantees for multiple paths.

There are several papers [1, 4] that find shortest paths homotopic to a given collection of input paths. However, while a set of shortest paths homotopic to a set of non-crossing input paths is necessarily non-crossing, the same does not hold for minimum-link paths. Our problem is also related to *wire routing* in VLSI design [3, 5, 7, 8]. None of these papers strives to minimize the number of links. Gupta and Wenger [6] present an approximation algorithm (with an approximation factor larger than 120) for finding non-crossing minimum-link paths inside a simple polygon, where all endpoints lie on the boundary of the polygon. Their paths are not $\mathcal{C}$-oriented. In a previous paper [12] we presented a 2-approximation algorithm for the rectilinear version of our problem.

Results. In this paper we generalize our previous result [12] to $\mathcal{C}$-oriented paths. To this end we introduce a special type of $\mathcal{C}$-oriented paths—

*Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands, k.a.b.verbeek@tue.nl. Kevin Verbeek is supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 639.022.707.

smooth paths—which are a generalization of rectilinear paths that retain several important properties of rectilinear paths. For the problem that asks for smooth paths—the *smooth routing problem*—we present a 2-approximation algorithm which runs in $O(n^2(n + \log \kappa) + k_{in} \log n)$ time, where $n = n_p + n_v$, k_{in} is the total number of links in the input, and $\kappa = |\mathcal{C}| \geq 2$. For the general $\mathcal{C}$-oriented routing problem, this algorithm computes an $O(\kappa)$-approximation.

We also study a related problem that occurs as a subproblem of our approach. Given a set of $\mathcal{C}$-oriented paths (with crossings) with a total of L links, how many links are necessary to untangle the paths, that is, how many links must a set of non-crossing $\mathcal{C}$-oriented paths have that are homotopic to the input paths? For smooth paths, $2L$ links are always enough and this bound is tight. For $\mathcal{C}$-oriented paths we show that $O(L\kappa)$ links are always enough and $\Omega(L \log \kappa)$ links can be necessary.

2 Preliminaries

We represent a $\mathcal{C}$-oriented path by a sequence of links $\langle \ell_1, \ldots, \ell_k \rangle$ that are connected together at *bends*. We assume that $\mathcal{C}$ is given as a set of vectors $\{c_1, \ldots, c_\kappa\}$ ordered clockwise, where c_1 points in the positive y-direction and the other vectors lie in the right half-plane. We define $c(\ell)$ as the orientation of a link ℓ. So a path π is $\mathcal{C}$-oriented iff $c(\ell) \in \mathcal{C}$ for every $\ell \in \pi$.

We define a $\mathcal{C}$-oriented path to be *smooth* if at every bend we do not "skip an orientation". To define this formally, we split up every orientation into two directions to obtain a set $\vec{\mathcal{C}} = \{\vec{c}_1, \ldots, \vec{c}_{2\kappa}\}$, with $\vec{c}_i = c_i$ and $\vec{c}_{i+\kappa} = -c_i$ for $1 \leq i \leq \kappa$. If we direct a $\mathcal{C}$-oriented path from one endpoint to the other, every link ℓ of a $\mathcal{C}$-oriented path has a direction $\vec{c}(\ell) \in \vec{\mathcal{C}}$. We can distinguish two types of bends: clockwise (CW) bends make a right turn, and counterclockwise (CCW) bends make a left turn. A bend between two links ℓ_i and ℓ_{i+1} is smooth if $\vec{c}(\ell_i)$ and $\vec{c}(\ell_{i+1})$ are adjacent directions. A smooth $\mathcal{C}$-oriented path, or *smooth path* in short, has only smooth bends. Note that we do not give lower bounds on the length of a link; for every $\mathcal{C}$-oriented path π, there is a geometrically equivalent smooth path π', but π' can have more links than π.

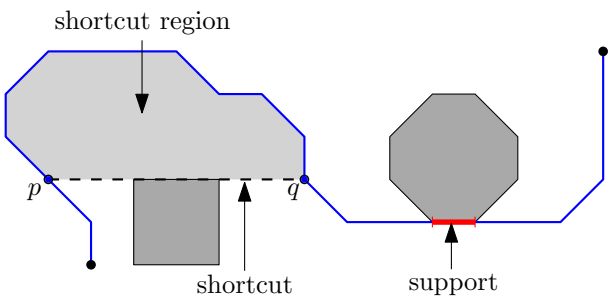


Figure 2: Supports and shortcuts.

Three consecutive links form a CW (CCW) *U-turn* if both bends between the links are CW (CCW). A CW (CCW) U-turn is *tight* if an obstacle is touching the right (left) side of the middle link. The overlapping part between the middle link and the obstacle is called the *support* of a U-turn (see Fig. 2). The subpaths of a smooth path between the supports of two consecutive U-turns are called *staircase chains*. Because the path is smooth, staircase chains can use only two (adjacent) directions or orientations. Two staircase chains have the same type if they use the same pair of orientations. Finally, a *shortcut* is defined by two points p and q on a smooth path π, not on the same link, with the following properties: (i) the orientation of $\overline{pq}$ is in $\mathcal{C}$, and (ii) the region enclosed by $\overline{pq}$ and the part of π between p and q, the *shortcut region*, does not contain an obstacle. We assume that $\overline{pq}$ does not cross π, for otherwise we can split $\overline{pq}$ up into multiple shortcuts. We can apply a shortcut to π, which means that we replace the subpath between p and q by $\overline{pq}$. It is easy to see that the resulting path is shorter, $\mathcal{C}$-oriented, and homotopic to π.

Lemma 1 *A path π is shortest w.r.t. its homotopy iff π has no shortcuts. All shortest paths $\pi' \sim_h \pi$ have only tight U-turns and the same sequence of supports.*

Rectilinear paths allow *smallest paths*, paths that minimize both the length and the number of links simultaneously, which is vital to the algorithm in [12]. The same property holds for smooth paths.

Lemma 2 *For every path π, there is a smooth path $\pi' \sim_h \pi$ that minimizes both the length and the number of links for all smooth paths homotopic to π.*

Proof (sketch). Let π' be the path constructed as follows. We start with a minimum-link smooth path homotopic to π. Then we keep applying shortcuts to this path until it has no more shortcuts. By Lemma 1, π' must be shortest. We need to show that applying a shortcut does not increase the number of links. Consider a shortcut between p and q on links ℓ_i and ℓ_j ($i < j$), respectively. Assume w.l.o.g. that there are at least as many CW bends as CCW bends between ℓ_i and ℓ_j. Then $\overline{pq}$ must lie to the right of ℓ_i[1]. Because the path is smooth, all directions clockwise from $\vec{c}(\ell_i)$ to $\vec{c}(\ell_j)$ must be used. After applying the shortcut, we use each of these directions exactly once. Hence the number of links cannot increase. In the special case that q (or p) is an endpoint of π', we can replace $\vec{c}(\ell_j)$ by $\overrightarrow{pq}$ and the same argument works. □

Smallest paths are not unique and it is NP-hard to decide if there exist non-crossing smallest paths [12] (the result easily extends to smooth paths). Still, smallest paths are a good starting point for our algorithm.

[1] A formal proof of this is quite tedious and hence omitted from this version of the paper.

3 Algorithm

Our algorithm consists of two steps. First we compute the smallest paths, and then we untangle the paths.

Computing smallest paths. We first compute *lowest* and *highest* paths. A staircase chain ρ between a point p and a point q is *lowest* (*highest*) if ρ is the lower (upper) envelope of all staircase chains homotopic to ρ. A smooth path π is *lowest* (*highest*) if it is shortest and all its staircase chains are lowest (highest). Note that lowest and highest paths are well-defined and unique.

Lemma 3 *Lowest (Highest) paths are non-crossing.*

To compute lowest and highest paths we use the algorithm of [4]. Their algorithm first finds the endpoints of x-monotone chains of Euclidean shortest paths. Note that the endpoints of x-monotone chains of shortest smooth paths are the same: they are exactly at the supports of tight U-turns for which the middle link has orientation c_1. There are only $O(n)$ homotopically different x-monotone chains. Next we compute lowest and highest paths for each x-monotone chain. By Lemma 1, lowest and highest paths must have the same sequence of supports. To obtain smallest paths, we compute minimum-link staircase chains between every two consecutive supports, which must be bounded by the lowest and highest paths. This way we can compute all smallest paths in $O(n^2 \log \kappa + k_{in} \log n)$ time. In the following we assume that paths are x-monotone and directed from left to right.

Untangling. We reduce the problem to rectilinear paths so that we can invoke the algorithm of [12].

Lemma 4 *Given a set of shortest smooth paths, crossings can occur only between staircase chains of the same type.*

Proof. Assume that two staircase chains ρ and ρ' cross. If they cross only once, then the only way to remove the crossing is to move ρ over an endpoint of ρ' or vice versa. Because the paths are shortest, there must be obstacles at the endpoints of ρ and ρ', so staircase chains homotopic to ρ and ρ' always cross. But, by Lemma 3, lowest paths are non-crossing, so

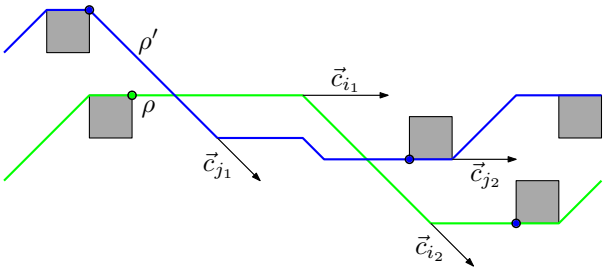


Figure 3: Only staircases of the same type cross.

ρ and ρ' must cross at least twice. Let the links of ρ and ρ' involved in the first crossing have directions $\vec{c}_{i_1}$ and $\vec{c}_{j_1}$, respectively. Assume w.l.o.g. that $i_1 < j_1$. Similarly define i_2 and j_2 for the second crossing, for which $j_2 < i_2$ (see Fig. 3). Because ρ and ρ' can use only two adjacent directions, we get that $|i_2 - i_1| \leq 1$ and $|j_2 - j_1| \leq 1$. But then $i_1 = j_2$ and $i_2 = j_1$. □

We can now use the following approach. Choose a single type of staircase chains and remove staircase chains of different types. We rotate the plane such that one of the orientations aligns with the x-axis, and then we apply a shearing transformation to align the other orientation with the y-axis. Because this is an affine transformation, straight lines (links) and crossings are preserved by the transformation (see Fig. 4). Using the incremental untangling algorithm of [12], we untangle the staircase chains using at most twice the number of links. We do this for all types of staircase chains. By Lemma 4, all crossings will be removed.

Note that we do not need to explicitly compute the transformation. We can apply the algorithm to complete x-monotone chains directly. If we add the paths from top to bottom, the algorithm adds two links per CW bend. In the opposite order, the algorithm adds two links per CCW bend. Choosing the right order ensures a 2-approximation.

Theorem 5 *We can compute a 2-approximation for the smooth routing problem in $O(n^2(n + \log \kappa) + k_{in} \log n)$ time, which is also a $(2\kappa - 2)$-approximation for the $\mathcal{C}$-oriented routing problem.*

Proof. The first part follows from the discussion above and the result in [12]. Consider a minimum-link $\mathcal{C}$-oriented path π with L links. We can make π

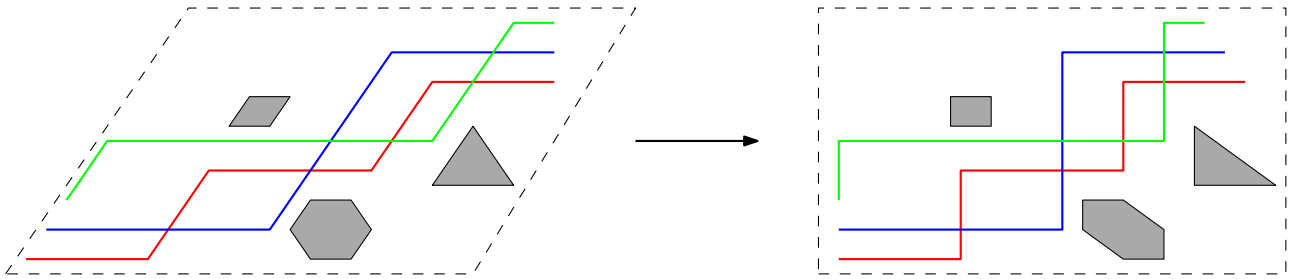


Figure 4: Shearing transformation to rectilinear paths.

smooth by adding at most $\kappa - 2$ links per bend, so a minimum-link smooth path homotopic to π has at most $(\kappa-1)L$ links. This implies the second part. □

We should note that the untangling algorithm is not restricted to smallest paths. In fact, the untangling algorithm just requires Lemma 4 to hold. Unfortunately, we have not yet been able to exploit this fact to improve the approximation factor for the $\mathcal{C}$-oriented routing problem. As in [12], we can extend our algorithm to thick paths, but we omit the details from this version of the paper.

4 Untangling bounds

We have shown that a set of $\mathcal{C}$-oriented paths with a total of L links can be untangled using at most $O(L\kappa)$ links. But how many links are necessary? For smooth paths with L links, a similar construction as for rectilinear paths [12] shows that $2L$ links can be necessary. For general $\mathcal{C}$-oriented paths, we give a lower bound that is based on a construction from [6].

Theorem 6 *There is a set of orientations $\mathcal{C}$ and a set of $\mathcal{C}$-oriented paths with a total of L links such that we need $\Omega(L \log \kappa)$ links to untangle the paths.*

Proof. Consider the quadrilateral enclosed by two paths shown in Fig. 5 (left). We can replace this quadrilateral by two crossing quadrilaterals as shown in the figure. We repeat this operation recursively for each quadrilateral. After k rounds, there are 2^k quadrilaterals formed by 2^{k+1} paths with 2 links each. To be able to untangle these paths, each quadrilateral that occurred during the construction must be free of obstacles. The rest of the plane can be filled with obstacles, so we assume that this is the case. As is shown in [6], we need at least $\Omega(k2^k)$ links to untangle the paths. Let $\mathcal{C}$ be the set of all orientations formed by the 2^{k+2} links. Then this construction consists of κ links and requires $\Omega(\kappa \log \kappa)$ links to untangle the paths. By copying this construction $\lfloor L/\kappa \rfloor$ times, we obtain the claimed bound, for L large enough. □

The above construction requires a specific set $\mathcal{C}$ to work. Usually, the set $\mathcal{C}$ is uniform, that is, the angle between two adjacent orientations is constant. The bound of Theorem 6 also holds if $\mathcal{C}$ is uniform. The idea is to find a subset of $\Omega(\sqrt{\kappa})$ orientations with which we can make the above construction. We omit the details from this version of the paper.

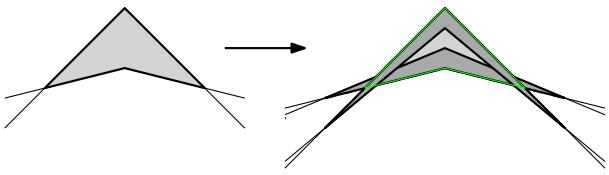


Figure 5: Lower bound construction.

5 Conclusion

We presented an $O(\kappa)$-approximation algorithm for the $\mathcal{C}$-oriented routing problem. To that end, we introduced smooth paths, and gave a 2-approximation algorithm for the smooth routing problem. Although we use smooth paths only as a tool for our algorithm, smooth paths might actually be better candidates for our application. Unlike smooth paths, general $\mathcal{C}$-oriented paths can have sharp bends which are hard to follow. So perhaps, for schematic maps, smooth paths are the better choice. We also studied how many links are necessary to untangle a set of $\mathcal{C}$-oriented paths with a total of L links. We know that $O(L\kappa)$ links are always enough and that $\Omega(L \log \kappa)$ links can be necessary. Finding a tight bound is an interesting open problem.

References

[1] S. Bespamyatnikh. Computing homotopic shortest paths in the plane. *J. Alg.*, 49(2):284–303, 2003.

[2] S. Cabello, M. de Berg, and M. van Kreveld. Schematization of networks. *Comp. Geom.: Theory and Appl.*, 30(3):223–238, 2005.

[3] R. Cole and A. Siegel. River routing every which way, but loose. In *Proc. 25th Symp. Found. Comp. Sci.*, pp. 65–73, 1984.

[4] A. Efrat, S. G. Kobourov, and A. Lubiw. Computing homotopic shortest paths efficiently. *Comp. Geom.: Theory and Appl.*, 35(3):162–172, 2006.

[5] S. Gao, M. Jerrum, M. Kaufmanm, K. Mehlhorn, and W. Rülling. On continuous homotopic one layer routing. In *Proc. 4th Symp. Comp. Geom.*, pp. 392–402, 1988.

[6] H. Gupta and R. Wenger. Constructing pairwise disjoint paths with few links. *ACM Trans. Alg.*, 3(3):26, 2007.

[7] C. E. Leiserson and F. M. Maley. Algorithms for routing and testing routability of planar vlsi layouts. In *Proc. 17th Symp. Theory Comp.*, pp. 69–78, 1985.

[8] M. Malley. Single-layer wire routing and compaction. MIT Press, Cambridge, MA, USA, 1990.

[9] D. Merrick and J. Gudmundsson. Path Simplification for Metro Map Layout. In *Proc. 14th Intern. Symp. Graph Drawing*, LNCS 4372, pp. 258–269, 2007.

[10] G. Neyer. Line Simplification with Restricted Orientations. In *Proc. 6th Int. Workshop on Alg. and Data Str.*, pp. 13–24, 1999.

[11] M. Nöllenburg and A. Wolff. Drawing and Labeling High-Quality Metro Maps by Mixed-Integer Programming. *IEEE Transactions on Visualization and Computer Graphics*, 17(5):626–641, 2011.

[12] B. Speckmann and K. Verbeek. Homotopic Rectilinear Routing with Few Links and Thick Edges. In *Proc. 9th Latin Am. Theor. Inf. Symp.*, LNCS 6034, pp. 468–479, 2010.

Exact and approximate algorithms for resultant polytopes

Ioannis Z. Emiris* Vissarion Fisikopoulos* Christos Konaxis†

Abstract

We develop an incremental algorithm to compute the Newton polytope of the resultant, aka resultant polytope, or its projection along a given direction. Our algorithm exactly computes vertex- and halfspace-representations of the resultant polytope using an oracle producing resultant vertices in a given direction. It is output-sensitive as it uses one oracle call per vertex. We implement our algorithm using the experimental CGAL package `triangulation`. A variant of the algorithm computes successively tighter inner and outer approximations: when these polytopes have, respectively, 90% and 105% of the true volume, runtime is reduced up to 25 times. Compared to tropical geometry software, ours is faster up to dimensions 5 or 6, and competitive in higher dimensions. The resultant is fundamental in algebraic elimination and in implicitizing parametric hypersurfaces: we compute the Newton polytope of surface equations in $<$ 1sec, when there are $<$ 100 terms in the parametric polynomials, which includes all common instances in geometric modeling.

Keywords. general dimension, convex hull, regular triangulation, polynomial resultant, CGAL implementation, experimental complexity

1 Introduction

Given pointsets $A_0, \ldots, A_n \subset \mathbb{Z}^n$, we define the Cayley pointset

$$\mathcal{A} := \bigcup_{i=0}^{n} (A_i \times \{e_i\}) \subset \mathbb{Z}^{2n},\ e_i \in \mathbb{N}^n, \qquad (1)$$

where $e_0, \ldots, e_n$ form an affine basis of $\mathbb{R}^n$: e_0 is the zero vector, $e_i = (0, \ldots, 0, 1, 0, \ldots, 0), i = 1, \ldots, n$. Clearly, $|\mathcal{A}| = \sum_i |A_i|$, where $|\cdot|$ denotes cardinality. By Cayley's trick ([14, sec.5]) the regular tight mixed subdivisions of the Minkowski sum $A_0 + \cdots + A_n$ are in bijection with the regular triangulations of $\mathcal{A}$; the latter correspond to the vertices of the *secondary polytope* $\Sigma(\mathcal{A})$.

The *Newton polytope* of a polynomial is the convex hull of the set of exponent vectors of monomials with nonzero coefficient. Given $n+1$ polynomials in n variables, with fixed exponent sets A_i, $i = 0, \ldots, n$, and symbolic coefficients, their *sparse (or toric) resultant* $\mathcal{R}$ is a polynomial in these coefficients which vanishes exactly when the polynomials have a common root. The resultant is the most fundamental tool in elimination theory, and is instrumental in system solving; it is also important in changing representation of parametric hypersurfaces. Our algorithms compute the Newton polytope of the resultant $N(\mathcal{R})$, or *resultant polytope*, in particular when some of the coefficients are not symbolic, in which case we seek a projection of the resultant polytope.

We exploit an equivalence relation defined on the secondary vertices. The class representatives correspond bijectively to the resultant vertices. This information defines an oracle producing a resultant vertex in a given direction, and avoids computing $\Sigma(\mathcal{A})$, which has much more vertices than $N(\mathcal{R})$. Although there exist efficient software computing $\Sigma(\mathcal{A})$ [13], it is useless in computing resultant polytopes. For instance, in implicitizing parametric surfaces with $<$ 100 input terms, we compute the Newton polytope of the equations in $<$ 1sec, whereas $\Sigma(\mathcal{A})$ is intractable.

Moreover we compute some orthogonal projection of $N(\mathcal{R})$, denoted Π, in $\mathbb{R}^m$:

$$\pi : \mathbb{R}^{|\mathcal{A}|} \to \mathbb{R}^m : N(\mathcal{R}) \to \Pi,\ m \leq |\mathcal{A}|.$$

By reindexing, this is the subspace of the first m coordinates. It is possible that none of the coefficients is specialized, hence $m = |\mathcal{A}|$, π is trivial, and $\Pi = N(\mathcal{R})$. Assuming the specialized coefficients take sufficiently generic values, Π is the Newton polytope of the corresponding specialization of $\mathcal{R}$.

1.1 Combinatorial characterization of $N(R)$

Sparse (or toric) elimination theory was introduced in [10], where $N(\mathcal{R})$ is described via Cayley's trick. In [14, sec.6] is proven that $N(\mathcal{R})$ is 1-dimensional iff $|A_i| = 2$, for all i, the only planar polytope is the triangle, whereas the only 3-dimensional ones are the

*Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, Athens, Greece. {emiris,vissarion}@di.uoa.gr. Partial support from project "Computational Geometric Learning", which acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the FP7 for Research of the European Commission, under FET-Open grant number: 255827.

†Archimedes Center for Modeling, Analysis & Computation (ACMAC), University of Crete, Heraklio, Greece, ckonaxis@acmac.uoc.gr. Enjoys support from the FP7-REGPOT-2009-1 project "Archimedes Center for Modeling, Analysis and Computation"

tetrahedron, the square-based pyramid, and the polytope of two univariate trinomials.

Following these results we study the combinatorial characterization of 4-dimensional resultant polytopes using our implementation as a tool. These are the 4-simplex [14, rmk.6.1] which corresponds to $n = 0$, $|\mathcal{A}| = 5, |A_0| = 5$, the polytopes of the univariate case $(n = 1, m = 7)$ as described in [9] and those of 3 trinomials $(|A_0| = |A_1| = |A_2| = 3)$.

Our experimental results shed more light in the combinatorial structure of $N(R)$. Their f-vectors (f_0, f_1, f_2, f_3) have the following properties: $5 \leq f_0, f_3 \leq 21$, $10 \leq f_1, f_2 \leq 63$, i.e., the minimum f-vector is $(5, 10, 10, 5)$ (4-simplex), while the maximum is $(21, 63, 63, 21)$. Moreover, for maximal f-vectors (i.e. the maximum values of f_1, f_2, f_3 if we fix f_0) it holds $f_0 = f_3, f_1 = f_2$.

Confirming the general characterization of resultant polytopes [14], the facets are either one of the 3-dimensional $N(R)$, or a triangular prism (Minkowski sum of a segment and a triangle), or a cube (Minkowski sum of 3 non-parallel edges). In the special case when $A_0 = A_1 = A_2 = \{(0,0), (1,0), (0,1)\}$, $N(R)$ is the 4-dimensional Birkhoff polytope [15]; its f-vector is $(6, 15, 18, 9)$.

For a more detailed presentation of the background of this paper see [5].

2 Algorithms and complexity

This section presents our exact and approximate algorithms for computing an orthogonal projection Π of $N(\mathcal{R})$ without computing $N(\mathcal{R})$, and analyzes their asymptotic complexity.

Given pointset V, reg_subdivision(V, w) computes the regular subdivision of its convex hull by projecting the upper hull of V lifted by the linear functional w, and conv(V) computes the H-representation of its convex hull. Extreme$\Pi(\mathcal{A}, w, \pi)$ computes a point in Π, extremal in the direction w, by refining the output of reg_subdivision$(\mathcal{A}, w)$ into a regular triangulation T of $\mathcal{A}$, then returns $\pi(\rho_T)$, where ρ_T is the resultant vertex corresponding to T. It is clear that, triangulation T constructed by ExtremeΠ, is regular and corresponds to some vertex ϕ_T of $\Sigma(\mathcal{A})$ which maximizes inner product with $\widehat{w} = (w, \vec{0}) \in (\mathbb{R}^{|\mathcal{A}|})^{\times}$.

The *initialization algorithm* computes an inner approximation of Π in both V- and H-representations (denoted Q, Q^H, resp.), and triangulated. First, it calls Extreme$\Pi(\mathcal{A}, w, \pi)$ for $w \in W \subset (\mathbb{R}^m)^{\times}$; set W is either random or contains, say, vectors in the $2m$ coordinate directions. Then, it updates Q by adding Extreme$\Pi(\mathcal{A}, w, \pi)$ and Extreme$\Pi(\mathcal{A}, -w, \pi)$, where w is normal to hyperplane $H \subset \mathbb{R}^m$ containing Q, as long as either of these points lies outside H. We stop when these points do no longer increase $\dim(Q)$.

Algorithm 1: ComputeΠ $(A_0, \ldots, A_n, \pi)$

Input : $A_0, \ldots, A_n \subset \mathbb{Z}^n$, $\pi : \mathbb{R}^{|\mathcal{A}|} \to \mathbb{R}^m$,
H-, V-rep. Q^H, Q, triang. T_Q of $Q \subseteq \Pi$
Output: H-, V-rep. Q^H, Q, triang. T_Q of $Q = \Pi$

$\mathcal{A} \leftarrow \bigcup_0^n (A_i \times e_i)$; $\mathcal{H}_{illegal} \leftarrow \emptyset$
foreach $H \in Q^H$ **do** $\mathcal{H}_{illegal} \leftarrow \mathcal{H}_{illegal} \cup \{H\}$
while $\mathcal{H}_{illegal} \neq \emptyset$ **do**
  select $H \in \mathcal{H}_{illegal}$; $\mathcal{H}_{illegal} \leftarrow \mathcal{H}_{illegal} \setminus \{H\}$
  w is the outer normal vector of H
  $v \leftarrow$ Extreme$\Pi(\mathcal{A}, w, \pi)$
  if $v \notin H \cap Q$ **then**
    $Q^H_{temp} \leftarrow \text{conv}(Q \cup \{v\})$
    foreach $(d-1)$-*face* $f \in T_Q$, $f \subset \partial H$ **do**
      $T_Q \leftarrow T_Q \cup \{\text{faces of } conv(f, v)\}$
    foreach $H' \in \{Q^H \setminus Q^H_{temp}\}$ **do**
      $\mathcal{H}_{illegal} \leftarrow \mathcal{H}_{illegal} \setminus \{H'\}$
    foreach $H' \in \{Q^H_{temp} \setminus Q^H\}$ **do**
      $\mathcal{H}_{illegal} \leftarrow \mathcal{H}_{illegal} \cup \{H'\}$
    $Q \leftarrow Q \cup \{v\}$; $Q^H \leftarrow Q^H_{temp}$
return Q, Q^H, T_Q

Lemma 1 *The initialization algorithm computes $Q \subseteq \Pi$ s.t. $\dim(Q) = \dim(\Pi)$.*

Incremental Alg. 1 computes both V- and H-representations of Π and a triangulation of Π, given an inner approximation Q, Q^H of Π computed at initialization. A hyperplane H is *legal* if it is a supporting hyperplane to a facet of Π, otherwise it is *illegal*. At every step of Alg. 1, we compute $v =$ Extreme$\Pi(\mathcal{A}, w, \pi)$ for a supporting hyperplane H of a facet of Q with normal w. If $v \notin H$, it is a new vertex thus yielding a tighter *inner approximation* of Π by inserting it to Q, i.e. $Q \subset \text{CH}(Q \cup v) \subseteq \Pi$, where CH$(\cdot)$ denotes convex hull. This happens when the preimage $\pi^{-1}(f) \subset N(\mathcal{R})$ of the facet f of Q defined by H, is not a Minkowski summand of a face of $\Sigma(\mathcal{A})$ having normal $\widehat{w}$. Otherwise, there are two cases: either $v \in H$ and $v \in Q$, thus the algorithm simply decides hyperplane H is legal, or $v \in H$ and $v \notin Q$, in which case the algorithm again decides H is legal but also insert v to Q.

Lemma 2 *Let vertex v be computed by Extreme$\Pi(\mathcal{A}, w, \pi)$, where w is normal to a supporting hyperplane H of Q, then $v \notin H \Leftrightarrow H$ is not a supporting hyperplane of Π.*

We now bound the *complexity* of our algorithm. Beneath-beyond, given a k-dimensional polytope with l vertices, computes its H-representation and a triangulation in $O(k^5 l t^2)$, where t is the number of full-dimensional faces (cells) [12]. Let $|\Pi|, |\Pi^H|$ be the number of vertices and facets of Π.

Lemma 3 *Alg. 1 computes at most* $|\Pi| + |\Pi^H|$ *regular triangulations of* $\mathcal{A}$.

Let the size of a triangulation be the number of its cells. Let $s_{\mathcal{A}}$ denote the size of the largest triangulation of $\mathcal{A}$ computed by ExtremeΠ, and s_Π that of Π computed by Alg. 1. In ExtremeΠ, the computation of a regular triangulation reduces to a convex hull, computed in $O(n^5|\mathcal{A}|s_{\mathcal{A}}^2)$; to compute ρ_T we need to compute the volume of all cells in T; this is done in $O(s_{\mathcal{A}}n^3)$. The overall complexity of ExtremeΠ becomes $O(n^5|\mathcal{A}|s_{\mathcal{A}}^2)$. Alg. 1 calls, in every step, ExtremeΠ to find a point on $\partial\Pi$ and insert it to Q, or conclude that a hyperplane is legal. By Lem. 3 it executes ExtremeΠ as many as $|\Pi| + |\Pi^H|$ times, in $O((|\Pi| + |\Pi^H|)n^5|\mathcal{A}|s_{\mathcal{A}}^2)$, and computes the H-representation of Π in $O(m^5|\Pi|s_\Pi^2)$. Now we have, $|\mathcal{A}| \leq (2n+1)s_{\mathcal{A}}$ and as the input $|\mathcal{A}|, m, n$ grows large we can assume that $|\Pi| \gg |\mathcal{A}|$ and thus s_Π dominates $s_{\mathcal{A}}$. Moreover, $s_\Pi(m+1) \geq |\Pi^H|$. Now, let $\widetilde{O}(\cdot)$ imply that polylogarithmic factors are ignored.

Theorem 4 *The time complexity of Alg. 1 is* $O(m^5|\Pi|s_\Pi^2 + (|\Pi| + |\Pi^H|)n^5|\mathcal{A}|s_{\mathcal{A}}^2)$, *which becomes* $\widetilde{O}(|\Pi|s_\Pi^2)$ *when* $|\Pi| \gg |\mathcal{A}|$.

This implies our algorithm is output sensitive. The performance of our algorithm confirms this property, see experimental evidence in Sect. 3.

Our algorithm readily yields an approximate variant: for each supporting hyperplane, we use its normal w to compute v =Extreme$\Pi(\mathcal{A}, w, \pi)$. Instead of computing a convex hull, now simply take the hyperplane parallel to H through v. The set of these hyperplanes defines a polytope $Q_o \supseteq \Pi$, i.e. an *outer approximation* of Π. Thus, we have an approximation algorithm by stopping Alg. 1 when $\text{vol}(Q)/\text{vol}(Q_o)$ achieves a user-defined threshold. Then, $\text{vol}(Q)/\text{vol}(\Pi)$ is bounded by the same threshold. Of course, $\text{vol}(Q)$ is available by our incremental convex hull algorithm. However, $\text{vol}(Q_o)$ is the critical step; we plan to examine algorithms that update (exactly or approximately) this volume.

3 Implementation and Experiments

We implement Alg. 1 in C++ to compute Π; our code is available in `http://respol.sourceforge.net`. All timings are on an Intel Core i5-2400 3.1GHz, with 6MB L2 cache and 8GB RAM, running 64-bit Debian GNU/Linux. We rely on CGAL, using mainly a preliminary version of package `triangulation` under development, working in general dimension, for both regular triangulations as well as for the vertex and halfspace-representation of Π.

We first compare 4 state-of-the-art exact convex hull packages: `triangulation (triang)` [3], `polymake`'s `beneath-beyond (bb)` [8]; `cdd` [7], and `lrs` [1], to compute the H-rep. from the vertices of Π (`triang/off`) actually extending the work in [2] for the new class of polytopes Π. We also test `triang` by inserting points in the order that Alg. 1 computes them (`triang/on`). The experiments show that `triang`, `bb` are faster than `lrs`, `cdd` and `triang/on` is 2.5 times faster than `triang/off` (Table 1).

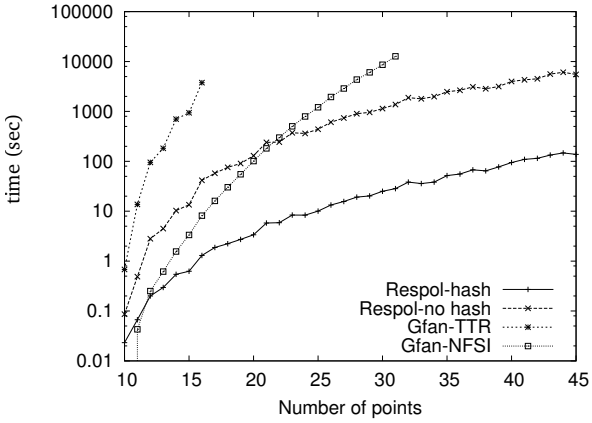


Figure 1: Comparison of `respol` (hashing and not hashing determinants) and 2 algorithms of `Gfan` for $m = 4$; y-axis in logarithmic scale.

We perform an experimental analysis of our algorithm confirming its *output-sensitivity*: its behaviour is subexponential wrt to both input and output and its output is subexponential wrt the input. Moreover, the size of the input bounds polynomially the size of the triangulation of the output.

The resultant is fundamental in *elimination*, and in *implicitizing* parametric hypersurfaces: we compute the polytope of surface equations in $<$ 1sec, assuming $<$ 100 terms in parametric polynomials, which includes all common instances in geometric modeling, whereas the corresponding $\Sigma(\mathcal{A})$ are intractable. By using the *hashing determinants* scheme [6] we gain a speedup of $18,100$ times when $m = 3, 4$ respectively. For $m = 4$ we computed in $<$ 2min an instance where $|\mathcal{A}| = 37$ and would take $>$ 1hr to compute otherwise, hence this method allows us to compute instances of the problem that would be intractable otherwise. We explore the *limits* of our implementation. By bounding runtime to $<$ 2hr, we compute instances of 5-, 6-, and 7-dimensional Π with 35K, 23K and 500 vertices, resp. (Table 1). We also compare with the implementation of [11], based on `Gfan` library. Our code is faster up to dimensions 5, 6, and competitive in higher dimensions.

We analyze the computation of inner and outer *approximations* Q and Q_o^H. We test the variant of Sect. 2 by stopping it when $\frac{\text{vol}(Q)}{\text{vol}(\Pi)} > 0.9$. In the experiments, the number of Q vertices is $<$ 15% of the Π vertices, thus the speedup is up to 25 times faster than the exact algorithm and $\frac{\text{vol}(Q_o^H)}{\text{vol}(\Pi)} < 1.04$. Next,

$dim(\Pi)$	$\lvert\mathcal{A}\rvert$	# of Π vertices	time					
			`respol`	`triang/on`	`triang/off`	`bb`	`cdd`	`lrs`
3	2490	318	85.03	0.07	0.10	0.067	1.20	0.097
4	37	2852	97.82	2.85	3.91	2.29	335.23	39.41
5	24	35768	4610.31	238.76	577.47	339.05	$>$ 1hr	$>$ 1hr
6	19	23066	6556.42	1191.8	2754.3	$>$ 1hr	$>$ 1hr	$>$ 1hr
7	17	500	302.61	267.01	614.34	603.12	10495.14	358.79

Table 1: Total time of our code (`respol`) and comparison of online/offline version of `triangulation (on/off)` and offline versions of all convex hull packages for computing the H-representation of Π.

we study procedures that compute only V-rep. of Q by counting the *random vectors* uniformly distributed on the m-sphere needed to obtain $\frac{\text{vol}(Q)}{vol(\Pi)} > 0.9$. This procedure runs up to 10 times faster than the exact algorithm but does not provides guarantees for $\frac{\text{vol}(Q)}{\text{vol}(\Pi)}$.

4 Future work

The resultant edges are associated to certain flips on mixed cells [14]. We have studied them algorithmically [4] and may revisit them in conjunction with the current approach: for every computed vertex of $N(\mathcal{R})$ apply all such flips to generate its neighbors.

A theoretical question is whether there is a *polynomial total time* incremental algorithm for Π. For this we wish to study the structure of $N(\mathcal{R})$; [10, 14], whereas our code sheds light to this issue by computing examples.

Acknowledgment. We thank L. Peñaranda for his useful help on implementation and experiments, A. Dickenstein for discussions on characterization of 4-dimensional $N(R)$, O. Devillers and S. Hornus for discussions on `triangulation` and A. Jensen and J. Yu for discussions on their approach and for sending us a beta version of their code.

References

[1] D. Avis. lrs: A revised implementation of the reverse search vertex enumeration algorithm. In *Polytopes - Combinatorics and Computation*, volume 29 of *Oberwolfach Seminars*, pp. 177–198. Birkhäuser-Verlag, 2000.

[2] D. Avis, D. Bremner, and R. Seidel. How good are convex hull algorithms? *Comput. Geom.: Theory & Appl.*, 7:265–301, 1997.

[3] J.-D. Boissonnat, O. Devillers, and S. Hornus. Incremental construction of the Delaunay triangulation and the Delaunay graph in medium dimension. In *SoCG*, pp. 208–216, 2009.

[4] I.Z. Emiris, V. Fisikopoulos, and C. Konaxis. Regular triangulations and resultant polytopes. In *Proc. Europ. Workshop Computat. Geometry*, Dortmund, Germany, 2010.

[5] I.Z. Emiris, V. Fisikopoulos, C. Konaxis, and L. Peñaranda. An output-sensitive algorithm for computing projections of resultant polytopes. arXiv:1108.5985v2 [cs.SC], 2011.

[6] I.Z. Emiris, V. Fisikopoulos, and L. Peñaranda. Optimizing the computation of sequences of determinantal predicates. In *Proc. Europ. Workshop Computat. Geometry*, Assisi, Italy, 2012.

[7] K. Fukuda. cdd and cdd+ home page. ETH Zürich. http://www.ifor.math.ethz.ch/~fukuda/cdd_home, 2008.

[8] E. Gawrilow and M. Joswig. Polymake: an approach to modular software design in computational geometry. In *Proc. Annual ACM Symp. Comp. Geom.*, pp. 222–231. ACM Press, 2001.

[9] I.M. Gelfand, M.M. Kapranov, and A.V. Zelevinsky. Newton polytopes of the classical resultant and discriminant. *Advances in Math.*, 84:237–254, 1990.

[10] I.M. Gelfand, M.M. Kapranov, and A.V. Zelevinsky. *Discriminants, Resultants and Multidimensional Determinants.* Birkhäuser, Boston, 1994.

[11] A. Jensen and J. Yu. Computing tropical resultants. *arXiv:math.AG/1109.2368v1*, 2011.

[12] M. Joswig. Beneath-and-beyond revisited. In M. Joswig and N. Takayama, eds., *Algebra, Geometry, and Software Systems*, Mathematics and Visualization. Springer, Berlin, 2003.

[13] J. Rambau. TOPCOM: Triangulations of point configurations and oriented matroids. In A.M. Cohen, X-S. Gao, and N. Takayama, eds., *Math. Software: ICMS*, pp. 330–340. World Scientific, 2002.

[14] B. Sturmfels. On the Newton polytope of the resultant. *J. Algebraic Combin.*, 3:207–236, 1994.

[15] G.M. Ziegler. *Lectures on Polytopes.* Springer, 1995.

Maximizing k-influence regions

Marta Fort* J. Antoni Sellarès*

Abstract

Let S be a finite set of points included in a bounded polygonal domain D of the plane. The order-k nearest influence region of a point in S is the set of points of D having the given point between their k-nearest neighbors in S. Let $\mathcal{P}$ be a weighted partition of the domain D into polygonal regions with an associated non-negative weight per region. We want to solve the problem of finding a new point s of D maximizing the weighted area of its order-k nearest influence region with respect to $S \cup \{s\}$, which is the sum of the weighted areas of the regions obtained intersecting the order-k nearest influence region with the regions of $\mathcal{P}$. We present a GPU parallel approach, designed under CUDA architecture, for approximately solving the problem and we also provide experimental results obtained with its implementation that show the efficiency and scalability of the approach.

1 Introduction

The main objective of single facility location problems is to place a new facility with respect to a given set of customers optimizing a certain objective function [4, 9]. In the context of competitive facility location, the new facility competes with pre-existing facilities. Competitive location models have been studied in several disciplines such as geography, economics, marketing and operations research [5, 11].

Example. Suppose that we want to find the optimal location for a new opening restaurant so that it takes over as many customers as possible from the existing competitors. If we assume that clients, that are nor equally distributed, eat at one of their 4 nearest restaurants, the new restaurant will be best located at the point having the biggest number of people living/working in the region having the new restaurant among their 4-nearest restaurants.

In this paper we address a single competitive facility location problem in which facilities and customers are modelled by points continuously dispersed over a planar domain, and so that costumers can move without barriers. The location of a facility is chosen supposing that costumers select facilities depending on distance. Under the assumption that customers are indifferent when it comes to choose among their k-nearest facilities, we define the influence region of a facility as the set of points of the domain having the given facility among their k-nearest neighbors. Consequently influence regions are not disjoint regions. On the other hand, we partition the domain into regions so that each region has associated a non-negative weight which represents its potentiality, for example population density or buying capacity of people living in the region. For any region included in the domain, we define its weighted area as the sum of the weighted areas of the subregions obtained intersecting the given region with the subregions of the partition of the domain. We want to solve the problem of locating a new facility in the domain optimizing the weighted area of its influence region.

Previous papers maximizing the Voronoi region of a new facility include [3, 2, 1]. In [3] a pixel based solution obtained by using graphics hardware is presented. Dehne et al. [2] show that the area function has only a single local maximum inside a region where the set of Voronoi neighbors does not change and is in convex position. They give an algorithm for approximately finding the optimal new facility based on Newton approximation. Cheong et al. [1] present a near-linear time algorithm for facilities in general position, that locates a new facility whose Voronoi region approximates the maximum area up to a $(1-\epsilon)$ factor.

The development of GPUs (Graphics Processing Units) hardware design, together with CUDA (Compute Unified Device Architecture) and some programming languages which use this architecture such as OpenCL, make them attractive to solve problems which can be treated in parallel as an alternative to CPUs in different computational demanding tasks, where a big amount of data or operations need to be done. General-purpose computing on GPUs (GPGPU) is capturing the attention of researchers in many computational fields ranging from numeric computing operations and physical simulations to bioinformatics and data mining [10].

The difficulty of optimizing, either analytically or by a numerical optimization procedure, the function that gives the weighted area of the nearest influence region determined by the inclusion of a new facility has motivated us to explore an alternative GPU parallel approach, designed under CUDA architecture, for approximately solving the problem. In this paper we describe the proposed solution and also provide

*Departament d'Informàtica i Matemàtica Aplicada, Universitat de Girona, Spain, {mfort,sellares}@ima.udg.edu.

experimental results obtained with the implementation that show the efficiency and scalability of the approach.

2 Preliminaries

2.1 Higher order Voronoi diagrams

Let S be a set of n points included in a bounded polygonal domain D of the Euclidean plane $\mathbb{R}^2$. We will denote by $d(p,q)$ the Euclidean distance between points $p, q \in \mathbb{R}^2$.

Given a subset S_k of k points of S, $k \in \{1, \cdots, n-1\}$, the order-k Voronoi region $V(S_k, S)$ of S_k is the set, possibly empty, of points of D lying closer to each point of S_k than to any other point of S:

$$V(S_k, S) = \{q \in D \mid d(s', q) \leq d(s, q), \forall s' \in S_k, s \in S - S_k\}.$$

The order-k Voronoi region $V(S_k, S)$ may not contain the points in S_k. Order-k Voronoi regions are convex polygonal regions as they arise as the intersection of halfplanes bounded by perpendicular bisectors of the points in S. The number of order-k Voronoi regions is $O(k(n-k))$ [7].

The order-k Voronoi diagram of S, denoted $\mathcal{V}_k(S)$, is the set of the order-k Voronoi regions $V(S_k, S)$ of all subsets S_k of k points:

$$\mathcal{V}_k(S) = \{V(S_k, S) \mid S_k \subseteq S, |S_k| = k\}.$$

2.2 The weighted area of a region

Let $\mathcal{P} = \{P_1, \cdots, P_m\}$ be a partition of the bounded polygonal domain D in the plane into polygonal regions that do not intersect except possibly along their boundaries.

We associate with each region P_i a non-negative number w_i, called the weight of P_i.

We define the weighted area of a subregion $R \subset D$, denoted $w(R)$, by:

$$w(R) = \sum_{i=1,\cdots,m} w_i \; \mu(R \cap P_i)$$

where $\mu(\overline{R})$ denotes the area of the subregion $\overline{R}$.

2.3 CUDA and GPU computing

CUDA is a parallel computing architecture that makes GPUs accessible for computation like CPUs. The CUDA processors, which can be executed in parallel, are referred as threads, and each thread executes the instructions contained in the so called kernels in parallel. Each thread computation is independent from the others, however, there exist some read-modify-write atomic operations called atomic functions. They read and return the value stored in a memory position, operate on it and store the result without allowing, during the whole process, any other access to that memory position. These operations allow the users to obtain global results when several threads access to the same memory position. For instance we can obtain a global sum, maximum or minimum in a specific position by using them.

Several types of memories can be used, data stored in global memory are accessible by every thread and are visible from the CPU, global memory is where more data can be allocated, but is the slowest access time memory. Registers are the fastest memory and store the local variables of each thread. Since the number of accesses to global memory are reflected in the execution times of the algorithms, ν readed, written or transferred values to global memory are represented by r_ν^g, w_ν^g and t_ν^g, in the complexity analysis.

Serial and parallel programming paradigms are focused on very different purposes making efficient serial algorithms unattractive candidates for GPUs [8]. When working with serial algorithms the total number of operations have to be minimized to guarantee time-efficiency, thus serial algorithms have to be work-efficient. When working in parallel, work-efficiency does not guarantee time-efficiency, and elaborated algorithms using complex data structures can not be built nor managed in parallel machines like GPUs.

3 Order-k influence regions

The order-k influence region of a point $s \in S$, denoted $I_k(s, S)$, is the set of points of D having s among their k-nearest points in S (see Figure 1). For any point $q \in D$, let $d_k(q, S)$ denote the distance to the k-th nearest point to q in S. Clearly:

$$I_k(s, S) = \{q \in D \mid d(s, q) \leq d_k(q, S)\} = \\ = \bigcup_{\substack{s \in S_k \subseteq S \\ |S_k| = k}} V(S_k, S).$$

Lemma 1

a) $I_k(s, S) \subset I_{k+1}(s, S)$.

b) Given k, for any $q \in D$ there exists $S_k \subseteq S, |S_k| = k$, so that $q \in I_k(s, S)$ for any $s \in S_k$.

c) $I_k(s, S)$ is a star-shaped polygonal region whose kernel contains s.

Proof.

a) This follows from influence regions definition.

b) Given $q \in D$, there exists a subset $S_k \subseteq S, |S_k| = k$, so that $q \in V(S_k, S)$, and consequently $q \in \bigcap_{s \in S_k} I_k(s, S)$.

c) We want to prove that for each point q in $I_k(s, S)$ the segment sq lies entirely in $I_k(s, S)$. Denote by $D_q(s)$ the disk of center q and radius $d(q, s)$. It is easy

to see that $q \in I_k(s,S)$ if and only if $|D_q(s) \cap S| \leq k$. Let q' be a point of segment sq. Since $D_{q'}(s) \subset D_q(s)$, we have $|D_{q'}(s) \cap S| < |D_q(s) \cap S| \leq k$, consequently $q' \in I_k(s,S)$. □

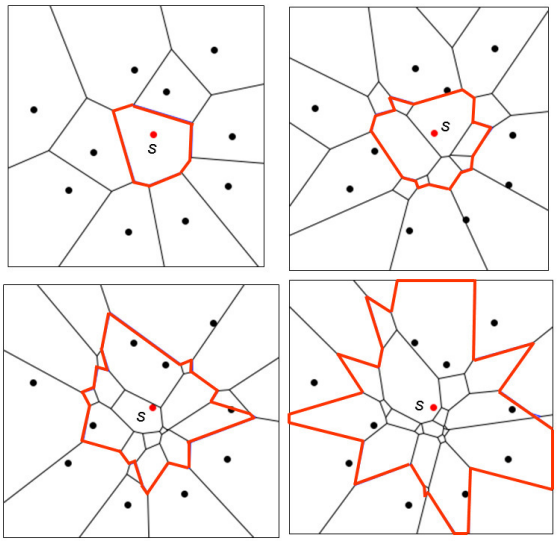


Figure 1: Nearest influence regions of point s for $k = 1$ to $k = 4$.

4 Maximizing the weighted area of a new order-k influence region

We want to solve the following problem:

Find the positions for a new point s maximizing the weighted area $w(I_k(s, S \cup \{s\}))$ over all possible points $s \in D - S$.

For a given s, the value of $w(I_k(s, S \cup \{s\}))$ can be obtained by computing the overlay intersection of the star-shaped polygonal region $I_k(s, S \cup \{s\})$ and the weighted partition $\mathcal{P}$. Since each maximal connected polygonal region of the resulting intersection is contained in a region of $\mathcal{P}$, it has univocally associated a weight. Consequently, the weighted area $w(I_k(s, S \cup \{s\}))$ can be written as a sum of functions that depends on the location of s.

4.1 Solving the problem on the GPU using CUDA

We approximately solve the maximization problem discretizing the polygonal domain D by superimposing on it an axis-parallel rectangular grid of size $G = H \times W$. We denote by Q the set of grid points corresponding to the geometric center of the grid cells. First, for each grid point q of Q we compute an estimation of the weighted area $w(I_k(q, S \cup \{q\}))$ from the weight of the grid points that are contained in $I_k(q, S \cup \{q\})$. Next we find the points q of Q maximizing $w(I_k(q, S \cup \{q\}))$. All the 1D-arrays we use in the presented algorithm are stored in GPU global memory.

The input of the process consists on the grid dimensions H, W, the coordinates of the points in S, and the weights associated to the grid points of Q according to the weighted partition $\mathcal{P}$ of D. This information is transferred from the CPU to the GPU storing the coordinates of the points in S and the weights of points of Q in $1D$-arrays of size n and G, respectively. Notice that the grid points are not explicitly stored but obtained when needed from the grid dimensions and the domain vertices. The output contains the maximal value together with the possible optimal placements between the grid points of Q.

We start by computing the distance of each grid point q to its k-th neighbor in S, considering all the grid points in parallel and using the algorithm presented in [6]. The obtained distances are stored in a 1D-array of size G.

Next, the weighted area $w(I_k(q, S \cup \{q\}))$ of each grid point q is estimated by using the precomputed distances from grid points to k-th neighbors and the weights associated to grid points. It is done by considering all the grid points $q' \in Q$ in parallel. For each $q' \in Q$ we find all the points $q \in Q$ such that $q' \in I_k(q, S \cup \{q\})$ using the fact that in this case $d(q,q') \leq d_k(q', S \cup \{q\})$, or equivalently $d(q,q') \leq d_k(q', S)$. Thus, for each grid point q' we find the points $q \in Q$ contained in the disk of center q' and radius $d_k(q', S)$ by using a multiple disk range query using the ideas presented in [6]. The k-influence values of all these points are updated by adding at the existing value the weight w' associated to q' using an atomic function. The k-influence values of each grid point are stored in a 1D-array of size G.

Once the k-influence values of all the grid points are known, the optimal placements, among the considered ones, are determined by using a reduction type algorithm [12] to find the maximal k-influence value and the number of points where this maximal value is achieved. Then an array of the corresponding size is allocated to store in a second step the points where this maximal value is obtained.

4.2 Complexity analysis

We only transfer from the CPU to GPU the sites S and the weights associated to the grid points, which takes $O(t^g_{n+G})$ time. No other CPU-GPU communication is needed. We read from global memory the weight and distance to the k-nearest neighbor of each grid point once, thus the number of reads is $O(r^g_{2G})$. We update the k-influence values, which are stored in global memory, by using atomic functions I times, where I is the total number of grid points contained in all the considered k-influence regions, which takes $O(w^g_{\rho I})$ time, ρ denotes writings done using atomic functions. Thus we have a time complexity of $O(t^g_{n+G} + r^g_G + w^g_{\rho I})$. Even though atomic operations slow down the running time of the algorithm they are necessary to guarantee correct results.

Next we analyze the work complexity. We execute

G parallel threads, each of them finds the k-th distance to S in $O(n)$ worst case time. Then the weighted areas are estimated using G threads executed in parallel. The number of grid points explored by each thread depends on $d_k(q', S)$, however, the total work in both steps is $O(Gn+I)$. In the worst case each grid point will explore $O(G)$ grid points and $I = O(G^2)$, yielding a work complexity of $O(G(G+n))$.

We use $O(n+G)$ space to store the coordinates of the n points of S and three arrays of size G. The GPU memory space limits the information that can be stored, which directly bounds the discretization size. However, if necessary, the domain can be partitioned into subdomains, the problem can be partially solved in each subdomain, and finally the solution of the problem can be obtained analyzing the results globally.

5 Experimental results

Table 1 provides times obtained by our implementation of the presented algorithm. We used OpenCL and the executions are done using a Intel Core 2 CPU 2.13GHz, 2GB RAM and a GPU NVidia GeForce GTX 480. The provided times are obtained as the median of 10 executions and show the scalability of our algorithm for several grid sizes and n. In order to provide realistic inputs, we used small values of k. However, considering $n = 10000$, $G = 1000 \times 1000$ and $k = 100$ they run in 0.18(s). CPU-GPU transferring times involved are not included and vary between 0.005(ms) and 0.062(ms).

G	100×100			500×500		
n / k	1	5	10	1	5	10
10	0.009	0.034	-	0.964	5.139	-
100	0.005	0.009	0.015	0.277	0.837	1.54
1000	0.003	0.005	0.007	0.072	0.159	0.257
10000	0.008	0.010	0.012	0.112	0.149	0.195

Table 1: Computational times in *seconds*.

Fig 2. a) presents a map colored from dark to light green according to the increasing weight assigned to the regions of the domain. Red points represent the 50 points of S. Fig 2. b) presents the same map colored from dark to light violet according to the increasing weight $w(I_{10}(q, S \cup \{q\}))$ of each grid point q. The white point shows the optimal placement for a new facility.

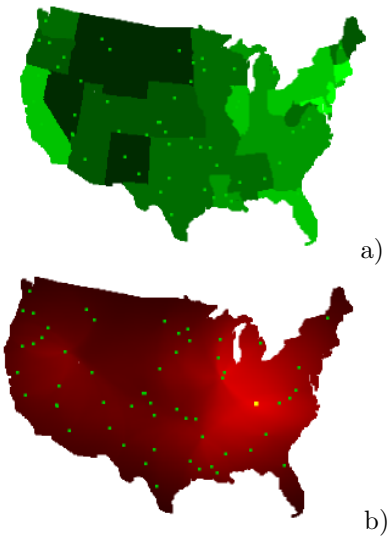


Figure 2: Optimal placement for a new facility.

Acknowledgments

Work partially supported by the Spanish Ministerio de Ciencia e Innovación under grant TIN2010-20590-C02-02.

References

[1] Otfried Cheong, Alon Efrat, and Sariel Har-Peled, Finding a guard that sees most and a shop that sells most, *Discrete Comput. Geom.* **37** (2007), no. 4, 545–563.

[2] Frank K. H. A. Dehne, Rolf Klein, and Raimund Seidel, Maximizing a Voronoi region: the convex case, *Int. J. Comput. Geometry Appl.* **15** (2005), no. 5, 463–476.

[3] Markus Denny, Solving geometric optimization problems using graphics hardware, *Comput. Graph. Forum* **22** (2003), no. 3, 441–452.

[4] Z. Drezner and Hamacher, *Facility location - applications and theory*, Springer Verlag, 2002.

[5] H. A. Eiselt, G. Laporte, and J. F. Thisse. Competitive location models: A framework and bibliography, *Transportation Science*, **27** (1993), 44–54.

[6] M. Fort and J.A. Sellarès, A parallel GPU-based approach for solving multiple proximity queries in 2d and 3d euclidean spaces, submitted.

[7] Der-Tsai Lee, On k-nearest neighbor Voronoi diagrams in the plane, *IEEE Transactions on Computers* **31** (1982), no. 6, 478–487.

[8] M.D. Lieberman and J. Sankaranarayanan and H Samet, A Fast Similarity Join Algorithm Using Graphics Processing Units, *International Conference on Data Engineering* (2008), 1111–1120.

[9] S. Nickel and J. Puerto, *Location theory - a unified approach*, Springer, 2009.

[10] John D. Owens, David Luebke, Naga Govindaraju, Mark Harris, Jens Krüger, Aaron E. Lefohn, and Timothy J. Purcell, A survey of general-purpose computation on graphics hardware, *Computer Graphics Forum* **26** (2007), no. 1, 80–113.

[11] F. Plastria. Static competitive location: an overview of optimisation approaches, *European Journal of Operational Research*, **129** (2001), 461–470.

[12] S. Sengupta, M. Harris, and M. Garland, Efficient Parallel Scan Algorithms for GPUs. *NVIDIA Technical Report NVR-2008-003* (2008).

Properties and Algorithms for Line Location with Extensions

Robert Schieweck* Anita Schöbel†

Abstract

In this work we extend some well-known properties for simple line location – namely an incidence and a halving property – to a more general setting and use them to develop algorithms in order to approximate a set of n given points by one or more lines. This yields an $O(n^{4/3}\log n)$ ($O(n^2)$) algorithm for locating a single line minimizing the sum of (weighted) distances to the n points for point-line distances induced by a norm. A trimmed line which minimizes the sum of (weighted) distances to the $m \leq n$ closest of the n points can be found in $O(n^2 \log n)$ ($O(n^3)$) time. Finally, K lines minimizing the sum of distances of each of the n points to the closest line is computed in $O(n^{2K+1}(K + \log n))$ time, where the $\log n$ factor only applies to the trimming case.

1 Introduction

Locating a line in the plane as to minimize the sum of distances (measured in a certain way) to n given points in the plane is a well-researched problem with many applications for example in location theory and statistics, when performing a linear regression. In this case n is often very large and fast algorithms to compute such a line are needed.

The classical problem

$$\min_{L} \quad \sum_{i=1}^{n} w_i d(p_i, L) \tag{1}$$

for points $P = \{p_1, \ldots, p_n\}$ with associated weights $w_1, \ldots, w_n > 0$ and a line L has been tackled for several distances d. Zemel [13] gave a linear time algorithm for the vertical distance case and the case where d is induced by the L_1 norm by generalizing a search technique for solving linear programs by Megiddo [7]. Later Yamamoto et al. [12] proposed algorithms for the L_2 norm case which run in $O(n^{4/3}\log n)$ time in the unweighted case and in $O(n^2)$ time in the weighted case using sweeping techniques by Edelsbrunner and Welzl [5] and Edelsbrunner and Guibas [4], respectively. Speed improvements for the former case are due to an improved computation of dynamic convex hulls by Brodal and Jacob [2] and a better upper bound on the computation time is due to Dey [3].

The algorithms of Yamamoto et al. [12] rely on two properties of the problem which have been observed by Morris and Norback [9] reducing (1) to a purely combinatorial problem:

Theorem 1 *If d is induced by the L_2 norm, there is an optimal line for* (1) *which passes through two of the points in P. Furthermore, any optimal line is pseudo-halving, i. e. the two open half-spaces induced by an optimal line cannot contain more than half of the total weight.*

In fact, these properties apply also to a larger class of distances d due to Schöbel [11]:

Theorem 2 *Theorem 1 holds true for any d induced by some norm $\|\cdot\|$ (and some other distances).*

It even holds that $\|\cdot\|$ is a smooth norm if and only if every optimal line for any instance of (1) must pass through two of the given points.

2 The classical case

Similar to the approach in [12] Theorem 2 can be used to develop an efficient algorithm for the case of a distance d induced by an arbitrary norm $\|\cdot\|$ using a sweeping technique by [5]. Denote by P^* the set of lines dual to the points in P and by $\mathcal{A}(P^*)$ the arrangement of those lines. Then $\mathcal{A}(P^*)$ is swept by a vertical line and the objective function in (1) is updated in constant time per vertex of $\mathcal{A}(P^*)$ which corresponds to the intersection of two lines of P^* which again corresponds to a line through two points of P by point-line duality. Hence we can sweep the dual arrangement to find an optimal L^* corresponding to an optimal line L for (1).

To this end denote by L^+ and L^- the two open half-spaces induced by a line L. Further let I_L^+ and I_L^- be the subsets of $\{1, \ldots, n\}$ containing the indices of the p_i in L^+ and L^-, respectively. Finally let W_L^+ and W_L^- be the sums of the w_i over the respective index sets as well as

$$X_L^{\pm} = \sum_{i \in I^{\pm}} w_i p_{i1}$$

$$Y_L^{\pm} = \sum_{i \in I^{\pm}} w_i p_{i2}.$$

*Institute for Numerical and Applied Mathematics, University of Göttingen, r.schieweck@math.uni-goettingen.de

†Institute for Numerical and Applied Mathematics, University of Göttingen, schoebel@math.uni-goettingen.de

This is an extended abstract of a presentation given at EuroCG 2012. 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 in a conference with formal proceedings and/or in a journal.

Then the objective function in (1) can be rewritten as

$$\sum_{i=1}^{n} w_i d(p_i, L) = \frac{1}{\|(s,-1)\|^\circ} \sum_{i=1}^{n} w_i |p_{i2} - sp_{i1} - b| = \frac{1}{\|(s,-1)\|^\circ} \Big(Y_L^+ - Y_L^- - s(X_L^+ - X_L^-) - b(W_L^+ - W_L^-) \Big)$$

where the first equality holds according to Plastria and Carrizosa [10] with the dual or polar norm $\|\cdot\|^\circ$ of $\|\cdot\|$, s the slope of L and b its intercept. Thus the objective can be updated in constant time by keeping track of the variables above which also takes constant time.

In fact, the type of sweeping depends on whether the weights are all equal or not. In the unweighted case $w_1 = \ldots = w_n$ only solutions on the k-levels of $\mathcal{A}(P^*)$ for $k = n/2$ and $k = n/2 + 1$ when n is even and $k = \lfloor n/2 \rfloor + 1$ when n is odd need to be considered since the dual L^* of a line L optimal for (1) must lie on this k-level according to the pseudo-halving property in Theorem 2. Evaluating the objective at vertices suffices due to the incidence property in Theorem 2. Therefore an algorithm for the unweighted case runs in $O(n^{4/3} \log n)$ time applying a sweeping technique by [5] with improvements by [2, 3]. For the weighted case one cannot restrict the evaluation to only the above mentioned k-levels, since weighting also allows vertices on other levels closer to the 1-level or the n-level to become candidate solutions for (1). This causes the complexity of the sweeping to increase to $O(n^2 \log n)$ which can be outperformed by not only sweeping certain levels of $\mathcal{A}(P^*)$ but the whole arrangement with the topological sweeping technique in [4] yielding a running time of $O(n^2)$.

Note that the degenerate case of an optimal vertical line can be treated separately without compromising the time complexity.

3 The concept of trimming

An extension of the classical problem is the incorporation of the concept of trimming which is often needed in robust statistics. This means, a line is sought as to minimize the weighted distances to the $m \leq n$ closest points. The remaining $n - m$ points are called *outliers* and may have an arbitrary distance to the line without influencing the objective function, i. e.

$$\min_L \quad \sum_{i=1}^{m} w_{(i)} d(p_{(i)}, L) \tag{2}$$

where $(\cdot)$ is a permutation of $\{1, \ldots, n\}$ that sorts point-line distances in ascending order

$$d(p_{(1)}, L) \leq d(p_{(2)}, L) \leq \ldots \leq d(p_{(n)}, L),$$

(i) denoting the ith index with respect to this order. From a statistician's point of view this ensures a regression line to be unaffected by some very inaccurate observations. Note that the permutation $(\cdot)$ depends on L but we will omit an additional index for the sake of shortness of notation.

With some additional notions Theorem 2 can be modified to accomodate the new setting. Let $I^{\pm}_{L,m}$ and $W^{\pm}_{L,m}$ be defined similarly to the previous section but restricted to the none-outlier points, e. g.

$$I^{\pm}_{L,m} = I^{\pm}_L \cap I_{L,m}$$

where $I_{L,m} = \{(1), \ldots, (m)\}$. Further let $W_{L,m} = \sum_{i=1}^{m} w_{(i)}$. Then it holds

Corollary 3 *Let d be a distance induced by a norm. Then there is an optimal line for (2) passing through two of the points in P and any optimal line L satisfies $W^{\pm}_{L,m} \leq W_{L,m}/2$.*

Proof. Let L be optimal for (2). Then L is optimal for a problem of type (1) with respect to the point set $P' = \{p_i : i \in I_{L,m}\} \subseteq P$. Otherwise we had a contradiction to the optimality of L for (2). Applying Theorem 2 yields the second assertion.

Another application of Theorem 2 guarantees the existence of an optimal line L' for a problem of type (1) with respect to the point set $P' \subseteq P$ that passes through two of the points in P' and is also optimal for the original problem. □

By this result, (2) is also reduced to a purely combinatorial problem allowing a brute force approach which requires $O(n^3 \log n)$ time. This can be reduced again using a sweeping technique by [5]. To this end let

$$X^j_{s,m} = \sum_{i=1}^{j} w_{(i)_{\mathrm{td},s}} p_{(i)_{\mathrm{td},s}1}, \qquad Y^j_{s,m} = \sum_{i=1}^{j} w_{(i)_{\mathrm{td},s}} p_{(i)_{\mathrm{td},s}2} \quad \text{and} \qquad W^j_{s,m} = \sum_{i=1}^{j} w_{(i)_{\mathrm{td},s}}$$

for $j = 1, \ldots, n$, similar to $X^{\pm}_L$ in the previous section. $(\cdot)_{\mathrm{td},s}$ is the permutation which gives the top-down order of the lines P^* in the arrangement $\mathcal{A}(P^*)$ at the horizontal coordinate s. These $3n$ variables can be updated in constant time per vertex of $\mathcal{A}(P^*)$ during a complete sweep with a vertical line and allow for updating the objective function of (2) in constant time per vertex by rewriting

$$\sum_{i=1}^{m} w_{(i)} d(p_{(i)}, L) = \frac{1}{\|(s,-1)\|^\circ} \sum_{i=1}^{m} w_{(i)} |p_{(i)2} - sp_{(i)1} - b|$$

$$
\begin{aligned}
&= \frac{1}{\|(s,-1)\|^\circ} \sum_{i \in I^+_{L,m}} w_i(p_{i2} - sp_{i1} - b) \\
&\quad - \frac{1}{\|(s,-1)\|^\circ} \sum_{i \in I^-_{L,m}} w_i(p_{i2} - sp_1 - b) \\
&= \frac{1}{\|(s,-1)\|^\circ} \Big(Y^{k+j}_{L,m} - Y^k_{L,m} - s(X^{k+j}_{L,m} - X^k_{L,m}) \\
&\quad - b(W^{k+j}_{L,m} - W^k_{L,m}) \\
&\quad + Y^{k-(m-j)}_{L,m} - Y^k_{L,m} - s(X^{k-(m-j)}_{L,m} - X^k_{L,m}) \\
&\quad - b(W^{k-(m-j)}_{L,m} - W^k_{L,m}) \Big)
\end{aligned}
$$

for some j, where s is the slope of L and k the number of lines above or through L^* in $\mathcal{A}(P^*)$ at horizontal coordinate s. This can be evaluated in constant time per vertex in the unweighted case since the extension of the pseudo-halving property in Corollary 3 yields a strong restriction on the difference of the number of points in P below and above L and hence on the value of j. In fact, if L^* is a vertex of $\mathcal{A}(P^*)$ and the points P are in general position, then j must take one of the values $\{m/2-2, m/2-1, m/2\}$ if m is even and one of the values $\{(m-5)/2, (m-3)/2, (m-1)/2\}$ if m is odd. This gives a total running time of $O(n^2 \log n)$ to locate a line in the unweighted case with trimming. In the weighted case there is no such strong restriction on j but there are obviously not more than m possible choices of j since there are only $m = O(n)$ points which need to be close to L. Thus we need linear time to evaluate at one vertex and since there are $O(n^2)$ vertices this gives an $O(n^3)$ time algorithm for the weighted case.

Note, that the degenerate case of an optimal vertical line can again be treated separately without compromising the complexity.

4 Locating multiple lines

A further extension of the classical problem is to ask for more than one, say K, lines to be located in order to approximate n points. The distance of a point to the set of K lines is the distance to the closest of these lines. Again, trimming is allowed, i. e. only $m \leq n$ points need to be close to the set of lines yielding

$$\min_{\mathcal{L}=\{L_1,\ldots,L_K\}} \quad \sum_{i=1}^{m} \min_{k=1,\ldots,K} \Big\{ w_{(i)} d(p_{(i)}, L_k) \Big\} \quad (3)$$

where $\mathcal{L}$ is a set of K lines.

This has again applications in statistics, more specifically in *latent class regression*: the observations fall into a priori unknown classes, each of which has its own linear relationship between the explanatory and the response variables belonging to it. For example during the test of a new pharmaceutical its effect may differ between subjects with different and unknown previous illnesses.

Unfortunately, this problem is NP-complete in the strong sense according to Megiddo and Tamir [8] so we cannot hope for fast exact algorithms for its solution. But again we can transfer some of the properties for problem (1) to this new setting.

Corollary 4 *Let d be a distance induced by a norm and $K \in \mathbb{N}$. Then there is an optimal set $\mathcal{L} = \{L_1, \ldots, L_K\}$ of lines for (3) so that every line L_k, $k = 1, \ldots, K$ passes through two of the points in P.*

Proof. The proof uses an argument analogous to the proof of Corollary 3. Let $\mathcal{L} = \{L_1, \ldots, L_K\}$ be an optimal set of lines. Then consider some $k \in \{1, \ldots, K\}$ and the line L_k with its allocated points $P^k \subseteq P$. According to Theorem 2 there is an optimal line L'_k for a problem of type (1) with respect to points P^k. Then $\mathcal{L}' = \{L_1, \ldots, L_{k-1}, L'_k, L_{k+1}, \ldots, L_K\}$ is also optimal for (3). In this way all lines in $\mathcal{L}$ can be replaced by lines passing through two points of P and optimality is maintained. □

And again we are left with a purely combinatorial problem that can be solved by enumerating and evaluating all possible solutions. In this case the size of the candidate set is

$$\binom{n}{2} \cdot \binom{n-2}{2} \cdot \ldots \cdot \binom{n-2(K-1)}{2} = O(n^{2K})$$

since we must choose two points out of n to determine the first line, another two from the remaining points to determine the second one and so on.

To evaluate the objective every point has to be assigned to its respective closest line. To this end the Voronoi diagram of the K lines from the candidate set is constructed in $O(K^2)$ time [1] and can then be searched for a closest line to one of the given points in $O(\log K)$ [6]. After computing all n distances from a point to its respective closest line the mth smallest of them can be found in linear time by a selection algorithm. This yields a total time of $O(K^2 + n \log K)$ for the evaluation and hence a total time of $O(n^{2K}(K^2 + n \log K))$ for the brute-force solution of the problem.

5 Conclusion and outlook

We proposed several algorithms for different line location problems in the plane, all of which rely on two geometric properties, namely an incidence property and a halving property. Even though there are specialized algorithms which outperform the ones proposed here in their respective special cases (e. g. the linear programming approach for the L_1 norm case), our algorithms are general in the sense, that they allow for an arbitrary norm induced point-line distance and other variants like trimming and locating multiple lines. Moreover, cases are included which have not

been considered yet, as far as the authors are aware of.

Experiments to compare running times of different approaches are subject of further research. Especially the enumeration of all candidate solutions for multiple lines needs to be compared to a mixed-integer programming approach as well as the performance of approximate solution methods for outlier detection like RANSAC to the exact procedures. From a statistician's point of view, automatically determining a good choice for K would be of interest as well as some good heuristics since the problem instances can be very large.

Furthermore, stronger bounds on the number of k-sets would yield stronger bounds on the complexity of some of our algorithms.

Acknowledgments

The fincancial support by *DFG RTG 1023 "Identification in Mathematical Models"* is gratefully acknowledged. The authors thank one of the anonymous referees for their valuable comments.

References

[1] G. Barequet and K. Vyatkina. On Voronoi diagrams for lines in the plane. In *International Conference on Computational Science and Its Applications*, pages 159–168, 2009.

[2] G. Brodal and R. Jacob. Dynamic planar convex hull. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, pages 617 – 626, 2002.

[3] T. K. Dey. Improved bounds for planar k-sets and related problems. *Discrete Comput. Geom.*, 19(3, Special Issue):373–382, 1998. Dedicated to the memory of Paul Erdős.

[4] H. Edelsbrunner and L. J. Guibas. Topologically sweeping an arrangement. *J. Computer and System Sciences*, 38(1):165–194, 1989.

[5] H. Edelsbrunner and E. Welzl. Constructing belts in two-dimensional arrangements with applications. *SIAM J. Comput.*, 15(1):271–284, 1986.

[6] D. G. Kirkpatrick. Optimal search in planar subdivisions. *SIAM J. Comp.*, 12(1):28–35, 1983.

[7] N. Megiddo. Linear programming in linear time when the dimension is fixed. *J. Assoc. Comput. Mach.*, 31(1):114–127, 1984.

[8] N. Megiddo and A. Tamir. On the complexity of locating linear facilities in the plane. *Oper. Res. Lett.*, 1(5):194–197, 1982.

[9] J. G. Morris and J. P. Norback. A simple approach to linear facility location. *Transportation Science*, 14(1):1–8, 1980.

[10] F. Plastria and E. Carrizosa. Gauge distances and median hyperplanes. *J. Optim. Theory Appl.*, 110(1):173–182, 2001.

[11] A. Schöbel. *Locating Lines and Hyperplanes*, volume 25 of *Applied Optimization*. Kluwer Academic Publishers, Dordrecht, 1999. Theory and Algorithms.

[12] P. Yamamoto, K. Kato, K. Imai, and H. Imai. Algorithms for vertical and orthogonal L_1 linear approximation of points. In *Proceedings of the 4th Annual Symposium on Computational Geometry (Urbana, IL, 1988)*, pages 352–361, 1988.

[13] E. Zemel. An $O(n)$ algorithm for the linear multiple choice knapsack problem and related problems. *Inform. Process. Lett.*, 18(3):123–128, 1984.

On the homotopy test on surfaces with boundaries

Julien Rivaud* Francis Lazarus†

Abstract

Let G be a graph cellularly embedded in a surface $\mathcal{S}$ orientable or not, and with nonempty boundary. Given two closed walks c and d in G, we describe linear time algorithms to decide if c and d are homotopic in $\mathcal{S}$, either freely or with fixed basepoint. After $O(|G|)$ time preprocessing independent of c and d, our algorithms answer the homotopy test in $O(|c| + |d|)$ time, where $|G|$, $|c|$ and $|d|$ are the respective numbers of edges of G, c and d.

1 Introduction

Computational topology of surfaces has received much attention in the last two decades. Among the notable results we may mention the test of homotopy between two cycles on a surface [2], the computation of a shortest cycle homotopic to a given cycle [1], or the computation of optimal homotopy and homology bases [3]. In their 1999 paper, Dey and Guha announced a linear time algorithm for testing whether two curves are freely homotopic on a triangulated surface without boundary. In [4] we showed that their method is invalidated by subtle flaws and provided a new geometric approach that confirms the linear time bound on the free homotopy test. This technique can be extended to handle surfaces with boundaries by gluing a punctured torus to each boundary cycle. We nevertheless present a much simpler and self-contained method for surfaces with nonempty boundary which also answers the free homotopy test on non-orientable surfaces with boundary.

Let G be a graph cellularly embedded in a surface $\mathcal{S}$ with at least one boundary. Each face of G in $\mathcal{S}$ is thus a disk or an annulus. By extending its boundaries we can retract $\mathcal{S}$ onto a subgraph G' of G. Each homotopy class in G' has a canonical reduced representation obtained by removing spurs. To know whether two cycles of G are homotopic in $\mathcal{S}$ it is thus sufficient to compute their deformation retract on G', remove spurs until they are reduced and check them for equality (up to circular permutation).

An edge e of G does not necessarily retract on a single edge of G' but rather on a subwalk w_e of a boundary cycle of G' — that is a facial cycle of its embedding in $\mathcal{S}$. To achieve the claimed time bound we do not expand w_e down to edges of G' but keep it under the following abstract representation: a reference to a boundary walk along with start and end indices. In $\mathcal{O}(|G|)$ total time we can compute for all $e \in G$ the abstract subwalk which e retracts on. Retracting a cycle c of G then yields a sequence of such subwalks. The removal of a spur in the underlying expanded cycle can be expressed as an operation on these subwalks. These operations define a rewriting system that we run on the sequence of subwalks until its underlying cycle is reduced. This takes linear time in the initial number of subwalks, that is $\mathcal{O}(|c|)$ time.

The contractibility test easily reduces to the (free) homotopy test and we only consider this last test in this abstract. Given two cycles c and d in G, we first compute two reduced sequences s and t of abstract subwalks whose underlying cycles c' and d' are freely homotopic to c and d respectively. Even if c' and d' are cyclically equal, the sequences s and t may not be literal permutations of each other. If σ is a sequence of subwalks whose underlying reduced cycle is u, we define its ***canonical (cyclic) sequence*** $\mathrm{Can}(\sigma) = \mathrm{Can}(u)$ that only depends on u. We can now decide if c and d are homotopic by comparing $\mathrm{Can}(s)$ and $\mathrm{Can}(t)$ up to circular permutation of their subwalks. Since we can compute $\mathrm{Can}(s)$ and $\mathrm{Can}(t)$ in time proportional to their number of subwalks, we obtain:

Theorem 1 (Homotopy test) *Let G be a graph cellularly embedded in a surface $\mathcal{S}$ with at least one boundary. Let c and d be two cycles with a total of k edges in G. After a $\mathcal{O}(|G|)$ time preprocessing of G, independent of c and d, we can decide if c and d are freely homotopic in $O(k)$ time.*

2 Background

We provide some definitions and properties; see [5] or [6, chapter 3] for details on rotation systems.

Cellular embedding of graphs A graph is ***cellularly embedded*** in a surface $\mathcal{S}$ without boundary if every open face of its embedding is a disk. A cellular embedding can be encoded by a ***rotation system***, that is a set of ***half-edges*** with two unary operations: an involution exchanging the direction on edges, and a cyclic permutation around vertices. Each (half)-edge is associated a ***signature*** $\in \{-1, 1\}$ indicating whether

*GIPSA-Lab, CNRS, Grenoble, France;
julien.rivaud@gipsa-lab.grenoble-inp.fr
†GIPSA-Lab, CNRS, Grenoble, France;
francis.lazarus@gipsa-lab.grenoble-inp.fr

This is an extended abstract of a presentation given at EuroCG 2012. 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 in a conference with formal proceedings and/or in a journal.

the orientation of the cyclic permutation is the same or not around its endpoints. The face traversal procedure described in [6] allows to traverse all the facial cycles in $O(|G|)$ time. In particular, we can determine whether each edge of G is incident to only one facial cycle or to two distinct facial cycles.

Surfaces with boundaries In order to handle surfaces with boundaries we allow every face of G in $\mathcal{S}$ to be either a disk or an annulus. In other words G is a cellular embedding in the closure $\hat{\mathcal{S}}$ of $\mathcal{S}$ obtained by attaching a disk to every boundary of $\mathcal{S}$. We record this information by storing a boolean for every facial cycle of G indicating whether the associated face is perforated or not. Assume that an edge e is incident to a perforated face f and a plain face f'. We can perform an elementary collapse of f' through its free edge e, thus extending the perforation in f. Equivalently, we can remove e from G and merge f and f' into a single (perforated) face. We obtain this way an embedding of $G - e$ into $\mathcal{S}$ that simulates a deformation retraction of $\mathcal{S}$ without actually modifying $\mathcal{S}$.

Homotopy in embedded graphs We consider homotopy of closed walks in G with respect to $\mathcal{S}$. Hence two cycles of G are homotopic if one can be continuously transformed to the other on $\mathcal{S}$. If all the facial cycles are tagged as boundaries, then $\mathcal{S}$ is deform retracts onto G and homotopy on $\mathcal{S}$ reduces to homotopy on G. In particular, every cycle of G has a canonical homotopic cycle obtained by removing spurs until the cycle is reduced. As usual, a ***spur*** is the concatenation of two opposite oriented edges and a cycle is ***reduced*** if it contains no spur.

3 Retracting $\mathcal{S}$ to a thick graph

We first reduce the number of vertices of G:

Lemma 2 *Let G be a graph embedded on a surface $\mathcal{S}$. We can contract the edges of a spanning tree of G in $\mathcal{O}(|G|)$ time. We obtain this way a graph G_1 embedded on $\mathcal{S}$ with a single vertex, fewer edges than G and as many faces. Cycles in G are homotopic if and only if their contractions in G_1 are homotopic.*

Proof. We assume that every edge of G points to its incident faces. Updating and contracting each edge of the spanning tree can be done in constant time per edge by updating the rotation system: no face disappears or changes of boundary status. Computing and contracting a spanning tree of G thus takes $\mathcal{O}(|G|)$ time and produces an embedded graph G_1. □

A retraction of $\mathcal{S}$ From now on we suppose that G has a single vertex. Let us call an edge ***free*** if it is incident to two distinct faces, exactly one of which is perforated. We will simulate a sequence of elementary collapses in order to retract $\mathcal{S}$ onto a subgraph G' of G. We will thus obtain an embedding of G' into $\mathcal{S}$ such that all faces are perforated. To this end we maintain a list L of free edges. We start by putting all the free edges of G into L. We then pick an edge $e \in L$ and simulate the collapse of its plain incident face by removing e from G and merging its two incident faces. In practice, we just mark e as a ***merging*** edge and tag e as well as its plain incident face f with the name of the incident perforated face. We next update L, removing e from L and adding in or removing from L the other edges of f according to the new status of their incident faces. We repeat this procedure until L is empty. This ensures that all the faces are perforated since otherwise the connectivity of $\mathcal{S}$ would imply the existence of a free edge. Note that the handling of a free edge always involves an incident plain face that was not merged before. It easily follows that the complexity of the whole retraction is bounded, up to a multiplicative constant, by the sum of the lengths of the facial cycles, hence to $|G|$.

We call G' the resulting embedded graph, *i.e.*, the graph G minus the merging edges. If b is a facial cycle of G' of length $|b|$ and $i \in \mathbb{Z}/|b|\mathbb{Z}$ we denote by $b_{[i]}$ the $(i+1)$-th edge of b. An ***abstract subwalk*** of b — or just subwalk when there is no ambiguity — is a triplet $(i,j)_b$ where $i, j \in \mathbb{Z}/|b|\mathbb{Z}$. The ***underlying path*** of $(i,j)_b$ is the path $b_{[i]}b_{[i+1]}\cdots b_{[j]}$. Call E_b the set of merging edges tagged with the facial cycle b. Those edges are incident to a tree T_b of faces (also tagged with b) of G whose union is bounded by b and only one among those faces is perforated. Any $e \in E_b$ cuts b into two subpaths b_p, b_e such that the concatenation $b_p \cdot e$ surrounds the perforated face. Clearly, e retracts onto b_e. We can express b_e as an abstract subwalk w_e of b as follows. When the merging edge e is removed during the retraction phase we keep two pointers from e to the previous and next edge in the incident plain face f to be collapsed. Those pointers delimitate the complementary subpath of f onto which e retracts. We can differentiate a `start` and an `end` between those pointers by taking into account the orientation of the incident perforated face and the signature of e. At the end of the whole retraction we can obtain w_e by following the `start` and `end` pointers respectively, until we hit a non-free edge. We summarize the discussion into the following

Proposition 3 *Let G be a graph embedded on a surface $\mathcal{S}$ with at least one boundary. In $\mathcal{O}(|G|)$ time we can compute:*

- *a subgraph G' of G on which $\mathcal{S}$ retracts,*
- *a set B of boundary cycles, one per boundary cycle of G',*
- *for each oriented edge $e \in G$, an abstract subwalk $(i,j)_b$ whose underlying path is the deformation retract of e onto G', where $b \in B \cup B^{-1}$.*

4 Reducing a sequence of subwalks

The ***length*** $|a|$ of an (abstract) subwalk a is the length of its underlying path. The ***underlying cycle*** of a sequence of subwalks is the cycle obtained by concatenation of the individual underlying paths. A sequence of subwalks is ***reduced*** if its underlying cycle is.

Our goal is to cyclically search and remove spurs, preserving the free homotopy class. We express these simplifications with the following set of rules: for all $(i,j)_b$ and $(k,l)_d$ such that $b_{[j]} = d_{[k]}{}^{-1}$:

$$(i,j)_b \cdot (k,l)_d \longrightarrow \begin{cases} \epsilon & \text{if } i=j \text{ and } k=l \\ (i,j-1)_b & \text{if } i\neq j \text{ and } k=l \\ (k+1,l)_d & \text{if } i=j \text{ and } k\neq l \\ (i,j-1)_b \cdot (k+1,l)_d & \text{otherwise} \end{cases} \quad (1)$$

The following lemma ensures the correctness of our simulation:

Lemma 4 *Let s be a sequence of subwalks. If s is not reduced then there exist two cyclically consecutive subwalks in s on which one of the above rules apply.*

Proof. Since G' has only one vertex no boundary cycle can contain a spur. □

Running the rewriting system until no rule can cyclically apply gives us a new sequence of subwalks whose underlying loop is cyclically reduced and remains in the same free homotopy class. To better control the number of rewrites needed to reach a reduced sequence, we add a special case to the previous rule set:

$$(i,j)_b \cdot (-j-1,l)_{b^{-1}} \longrightarrow \begin{cases} \epsilon & \text{if } |(i,j)_b| = |(-j-1,l)_{b^{-1}}| \\ (i,l-1)_b & \text{if } |(i,j)_b| > |(-j-1,l)_{b^{-1}}| \\ (i+1,l)_{b^{-1}} & \text{if } |(i,j)_b| < |(-j-1,l)_{b^{-1}}| \end{cases} \quad (2)$$

These new rules recognize right away when the second subwalk undoes a whole chunk of the first along the same boundary cycle, and compute in a single step the result of removing spurs until only one subwalk remains. In particular lemma 4 stays true. Rules of this second type take precedence over the rules of set (1); if both types apply then we use a rule of set (2).

Lemma 5 *A path of length 2 in G' appears at most once as a subwalk of boundary cycles.*

Proof. If y follows an oriented edge x in both facial walks containing x, then ρ is an involution around the common vertex v of x and y; in particular v has degree 2. Because G' has only one vertex, G' is a single loop; but then boundary cycles have length 1. □

Lemma 6 *Let $s_1 s_2$ be a sequence of two subwalks on which some rule apply. Let s' be the resulting sequence, with precedence taken into account. Then no rule apply on s'.*

Proof. If s' has height 1, no rule can apply. Otherwise a rule of set (1) was used and $s' = (i,j-1)_b \cdot (k+1,l)_d$ where $s_1 = (i,j)_b$ and $s_2 = (k,l)_d$. In particular $b_{[j]} = d_{[k]}{}^{-1}$. If a rule of set (2) applies on s', then $d = b^{-1}$ and $k+1 = -(j-1)-1$. If a rule of type (1) applies on s', then $b_{[j-1]} = d_{[k+1]}{}^{-1} = (d^{-1})_{[-k-2]}$ and the subpath $b_{[j-1]}b_{[j]}$ appears both in b at position $j-1$ and in d^{-1} at position $-k-2$. Using lemma 5 we again get $b = d^{-1}$ and $k = -j-1$. This cannot be because no rule of type (2) applied on s. □

Lemma 7 *Suppose no rule apply on the sequence $s_1 s_2$. Let s_0 (resp. s_3) be a subwalk such that some rule apply on $s_0 s_1$ (resp. $s_2 s_3$), yielding with precedence a sequence of two subwalks $s'_0 s'_1$ (resp. $s'_2 s'_3$). Then no rule apply on $s'_1 s_2$ (resp. $s_1 s'_2$).*

Proof. The conditions on $s'_1 s_2$ (resp. $s_1 s'_2$) are exactly the same as on $s_1 s_2$. □

The ***height*** of a sequence of subwalks s is the number $h(s)$ of subwalks composing it. The ***inertia*** of $s = s_1 \cdots s_h$, denoted $i(s)$, is the maximum $k \leq h$ such that for all $1 \leq i \leq k$, $s_i s_{i+1}$ triggers no rule.[1] If $i(s) = h(s)$ then s is ***cyclically inert***.

Lemma 8 *Let $s = s_1 \cdots s_h$ be a sequence of subwalks of inertia $i < h$. Let r be the result of the rules applied on $s_{i+1}s_{i+2}$. If $i < h-1$ let $s' = s_1 \cdots s_i \cdot r \cdot s_{i+3} \cdots s_h$ else let $s' = s_2 \cdots s_{h-1} \cdot r$. Then $3h(s') - i(s') < 3h(s) - i(s)$.*

Proof. We first suppose $i < h-1$. If $h(s') = h(s)$ then $h(r) = 2$; lemmas 6 and 7 ensure $i(s') \geq i(s)+1$. Else $h(s') \leq h(s)-1$ and of course $i(s') \geq i(s)-1$. We now handle the case $i = h-1$. We always have $i(s') \geq i(s)-2$; if $h(s') < h(s)$ the result follows. Otherwise $r = r_1 r_2$. By lemma 6 $r_1 r_2$ triggers no rule, and neither do $s_{h+1} r_1$ nor $r_2 s_2$ by lemma 7. Hence $i(s') = h$. □

A direct consequence is:

Proposition 9 *Given a sequence s of subwalks we can compute in $\mathcal{O}(h(s))$ time a cyclically inert sequence s' of subwalks whose underlying cycle is freely homotopic to that of s. In particular s' is reduced.*

5 The free homotopy

Let c and d be two cycles on $\mathcal{S}$. Using propositions 3 and 9 we get two sequences s and t of subwalks whose underlying loops c' and d' are freely homotopic to c and d respectively. In particular c and d are freely homotopic if and only if $c' \equiv d'$ up to cyclic permutation. Explicitly comparing the underlying loops is too costly. We thus define a *canonical* representation of any reduced cycle as a sequence of subwalks, and show how to derive this canonical representation from s and t.

[1] By convention $s_{h+1} = s_1$.

A ***boundary mapping*** of an edge $e \in G'$ is any pair (b, i) where $b \in B \cup B^{-1}$ and $i \in \mathbb{Z}/|b|\mathbb{Z}$ such that $b_{[i]} = e$. Every $e \in G'$ has exactly two boundary mappings. We choose an arbitrary total order on $B \cup B^{-1}$. We define as follows the ***canonical mapping*** $\mathrm{CM}(c, e)$ of $e \in c$ with respect to a reduced cycle c. Let p and n be the edges respectively preceding and following e in c. If en is a subpath of some boundary cycle then by lemma 5 there is a unique pair (b, i) such that en occurs at position i in b and we set $\mathrm{CM}(c, e) = (b, i)$. Else, if pe is a subpath of some boundary cycle then $\mathrm{CM}(c, e)$ is the unique (b, i) such that pe occurs at position $i - 1$ in b. Otherwise let $\mathrm{CM}(c, e)$ be the mapping of e with minimal $b \in B \cup B^{-1}$. Two consecutive edges e_1 and e_2 of c are said to ***agree*** with each other if $\mathrm{CM}(c, e_1) = (b, i)$ and $\mathrm{CM}(c, e_2) = (b, i + 1)$. Let c be a reduced cycle and $p = e_1 \cdots e_k \subset c$ a subpath of agreeing edges. Let $(b, i) = \mathrm{CM}(c, e_1)$ and $k = |b|\, q + r$ the Euclidean division of k by $|b|$. The ***leftmost*** sequence of p is $[(i, i-1)_b]^q$ if $r = 0$ and $[(i, i-1)_b]^q \cdot (i, i+r-1)_b$ otherwise. If c is not the power of a boundary cycle then there is a unique decomposition $c = p_1 \cdots p_h$ into maximal subpaths of agreeing edges, that is where the last edge of p_i does not agree with the first edge of p_{i+1}. The ***canonical sequence*** of c is then the concatenation $\mathrm{Can}(c)$ of the leftmost sequences of $p_1, \ldots, p_h$. If c is the q-th power of a boundary cycle then $\mathrm{Can}(c) = [(0, -1)_b]^q$ is the leftmost sequence of the subpath of c following q times the corresponding boundary walk b. By definition $\mathrm{Can}(c)$ is unique up to circular permutation.

Lemma 10 *Two reduced cycles c and d are equal if and only if $\mathrm{Can}(c)$ and $\mathrm{Can}(d)$ are circular permutations of each other.*

If s is a sequence of subwalks with underlying cycle c then a subwalk $(i, j)_b \in s$ of underlying path $x \subset c$ is ***admissible*** if the l-th edge x_i of x has $\mathrm{CM}(c, w_l) = (b, i + l - 1)$. In particular, x_l agrees with x_{l+1}. A sequence of subwalks is admissible if all its abstract subwalks are. Of course $\mathrm{Can}(c)$ is admissible.

Lemma 11 *Let s is a sequence of subwalks with underlying cycle c. Let $w = (i, j)_b \in s$ with $i \neq j$. Then $w = (i, j-1)_b \cdot (j, j)_b$ where $(i, j-1)_b$ is admissible.*

Two consecutive subwalks $(i, j)_b$ and $(i', j')_{b'}$ in an admissible sequence ***agree*** with each other if $(b', i') = (b, j+1)$ — in other words the last edge underlying $(i, j)_b$ agrees with the first underlying $(i', j')_{b'}$.

Lemma 12 *Let s is a sequence of subwalks with underlying cycle c. If $s_1 \cdots s_h \subset s$ is a sequence of agreeing subwalks of s then we can compute in $\mathcal{O}(h)$ time the corresponding leftmost sequence.*

Proof. Compute $|s_1| + \cdots + |s_h|$ and divide by $|b|$ □

Proposition 13 *Given a reduced sequence s of subwalks with underlying cycle c, we can compute $\mathrm{Can}(c)$ in $\mathcal{O}(|s|)$ time.*

Proof. By computing a single canonical mapping we can replace any subwalk of length 1 by an admissible one. Together with lemma 11 this ensures we can compute an admissible sequence $s' = s'_1 \cdots s'_h$ with underlying cycle c such that $h \leq 2\,|s|$. We then search for some k such that s_k disagrees with s_{k+1}. If there is none then c is the power of a boundary cycle b: return $\mathrm{Can}(c) = [(0, -1)_b]^q$ where $q = \frac{|c|}{|b|} = \mathcal{O}(|s|)$. Otherwise, cut s' into subsequences of agreeing subwalks, computing with lemma 12 and concatenating their respective leftmost sequences . □

Now we can prove theorem 1:

Proof. Use propositions 3, 9 and 13 to compute in $\mathcal{O}(h)$ time the canonical sequences of two reduced cycles c' and d' freely homotopic to c and d respectively. c and d are freely homotopic if and only if c' and d' are, which happens if and only if c' and d' are equal as cycles of G', or equivalently if and only if $\mathrm{Can}(c')$ and $\mathrm{Can}(d')$ are cyclic permutations of each other. That last test can be answered in $\mathcal{O}(h)$ time with a Knuth-Morris-Pratt string search of $\mathrm{Can}(c')$ in $\mathrm{Can}(d')\mathrm{Can}(d')$, with the added condition that $h(\mathrm{Can}(c')) = h(\mathrm{Can}(d'))$. Taking the initial preprocessing into account we have the claimed result. □

References

[1] É. Colin de Verdière and J. Erickson. Tightening non-simple paths and cycles on surfaces. *SIAM Journal on Computing*, 39(8):3784–3813, 2010.

[2] T. K. Dey and S. Guha. Transforming curves on surfaces. *Journal on Computer and System Sciences*, 58(2):297–325, 1999.

[3] J. Erickson and K. Whittlesey. Greedy optimal homotopy and homology generators. In *Proc. 16th ACM-SIAM Symp. on Discrete Algorithms. (SODA)*, pages 1038–1046, 2005.

[4] F. Lazarus and J. Rivaud. On the homotopy test on surfaces. arXiv:1110.4573v2 [cs.CG], 2011.

[5] W. S. Massey. *A Basic Course in Algebraic Topology*, volume 127 of *Graduate Texts in Mathematics*. Springer Verlag, 1991.

[6] B. Mohar and C. Thomassen. *Graphs on surfaces*. John Hopkins University Press, 2001.

On the Exploration Problem for Polygons with One Hole

Robert Georges* Frank Hoffmann* Klaus Kriegel*

Abstract

We study online strategies for autonomous mobile robots with vision to explore unknown polygons with at most one hole. We prove lower bounds on the competitive ratio of such strategies both for the orthogonal and the general case. Furthermore, we describe a competitive strategy for the case that the robot can distinguish between the edges of the outer boundary and the hole.

1 Introduction

A basic task for an autonomous mobile robot is to explore an unknown environment modeled by a polygon, possibly with holes. We assume the robot to be point shaped and to start from a given point, s, on the polygon's boundary. It is equipped with an unlimited 360° vision system that continuously provides the visibility polygon of its current position. When the robot has observed every point of the polygon it returns to s.

Assuming that the polygon is known, an optimal tour $\mathcal{T}_{opt}$ through s can be computed offline. The robot's performance exploring the unknown polygon online is evaluated through competitive analysis. Therefore we compare the length of the tour generated by the robot with the length of $\mathcal{T}_{opt}$. If this ratio is bounded from above by a constant $\mathcal{C}$ for any problem instance, we call the strategy $\mathcal{C}$–competitive.

Over the last two decades, the problem of designing competitive online exploration strategies for certain polygon classes has received a lot of attention. A simple greedy strategy is almost optimal for simple orthogonal polygons as shown in a seminal paper by Deng et al. [2], see also [5]. Later, Hoffmann et al. [6] came up with a 26.5–competitive strategy for general simple polygons (in the following called HIKK–strategy). On the other hand, there is a lower bound for the competitive ratio of 1.28 in this case [4]. If one allows polygons with h holes there is a lower bound of $\Omega(\sqrt{h})$, even for orthogonal polygons [1]. Computing the optimal offline tour becomes NP–hard.

Thus, one can ask about polygons with a fixed but small number of holes, say one hole in the easiest case. The only result in this direction we are aware of is a $O(h)$–competitive strategy for orthogonal polygons with h holes [2]. This result implies a 14–competitive strategy (L_1–metric) for the case of one hole.

We make the following contributions to the problem of exploring polygons with one hole. In Section 2 we prove a rather simple lower bound of 2 for the orthogonal case and a lower bound of 2.618 for the competitive ratio in the general case. This latter bound also holds for a modified model, where the hole is specially colored and the robot can therefore distinguish between outer boundary edges and edges of the hole. Undoubtedly, this should be of great advantage for the robot to fulfill its task. Nevertheless, it seems to be nontrivial to come up with a competitive strategy $\mathcal{S}_1$ under this assumption.

We describe such a strategy in Section 3. It proceeds in two phases. In Phase 1 it follows the HIKK–strategy until the hole H is visible for the first time. Then it learns, based on a doubling strategy, the relative convex hull R of the set $H \cup \{s\}$. In contrast to the orthogonal case, a competitive strategy cannot always afford to circle H completely. In Phase 2 a hybrid approach is implemented to explore the remaining "caves" inside and outside of R. This is based on the knowledge of the length $|R|$ of R. As soon as $\mathcal{S}_1$ knows that $|R|$ is less than $c \cdot |\mathcal{T}_{opt}|$, for a suitable universal constant c, it basically switches again to the HIKK–strategy (Theorem 3). Otherwise, if it is not safe to circle H, the strategy $\mathcal{S}_1$ will eventually rely on a doubling approach.

Although the competitive ratio of our strategy is huge (≈ 610), we consider the model of a colored hole a reasonable intermediate step towards the general case. All missing details of the strategy description and its analysis can be found in [3].

One might ask, why should this problem be so complicated after all? First of all, there is a notorious lack of tools to evaluate online strategies, which is one reason for the gap between lower and upper bounds even in the case of simple polygons. For example, the HIKK–strategy is believed to perform much better than the proven upper bound. Additionally, we have to cope with another difficulty. Optimal tours in polygons with one hole are in general no longer unique and no longer relatively convex. In contrast, in a simple polygon one can identify vertices the optimal tour has to go through and use those points for the analysis.

*Freie Universität Berlin, Institut für Informatik, 14195 Berlin, Germany, {georges, hoffmann, kriegel}@mi.fu-berlin.de

This is an extended abstract of a presentation given at EuroCG 2012. 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 in a conference with formal proceedings and/or in a journal.

2 Lower bounds

Theorem 1 *Any deterministic online strategy $\mathcal{S}_1$ that computes valid watchman routes in orthogonal polygons with at most one hole has a competitive ratio of at least 2.*

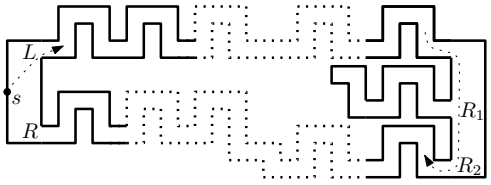


Figure 1: Lower bound example, orthogonal case

Proof. (Sketch) We confront $\mathcal{S}_1$ with a polygon composed of long thin winding corridors as indicated in Fig.1. In starting point s it has two choices, it can follow corridor L or R. Exploring them simultaneously, say by using a doubling strategy, see [7], is too expensive and because of the windings $\mathcal{S}_1$ cannot look far ahead. After travelling a corridor, say L, a distance $d \gg 0$, $\mathcal{S}_1$ encounters a new branching region with two new corridors R_1 and R_2 pointing back. Again, $\mathcal{S}_1$ has to decide which one eventually to follow, say it chooses R_2. Notice, in that moment R_1 has not been completely explored and we make R_1 a dead end corridor by cutting it off behind the next winding. $\mathcal{S}_1$ follows R_2 (which is in fact R) until it reaches the proximity of s again. Now $\mathcal{S}_1$ will notice that it has circled the hole completely and missed the very end of R_1 only. To accomplish its task it has to return to R_1, therefore travelling additionally a distance of $2d$ plus twice the length of R_1. An optimal tour will first learn R_1 before it returns to s. □

Notice, that this lower bound construction does not hold for the case of a colored hole. The strategy then could identify the dead end and explore it first.

If the robot discovers a new reflex vertex v that has an invisible incident edge, we call the extension of that edge into the polygon's interior the *cut* of v. The cut of a reflex vertex on the outer boundary could either hit the hole or pass by on the left/right side. This leads to the following lower bound construction.

Theorem 2 *Any deterministic online strategy $\mathcal{S}_1$ that computes valid watchman routes in general polygons with at most one hole has a competitive ratio of at least $\frac{3+\sqrt{5}}{2} \approx 2.618$.*

Proof. (Sketch) Consider the polygon given in Fig.2(a). After travelling a distance of 3ϵ an online strategy $\mathcal{S}_1$ has learned the hole completely and has

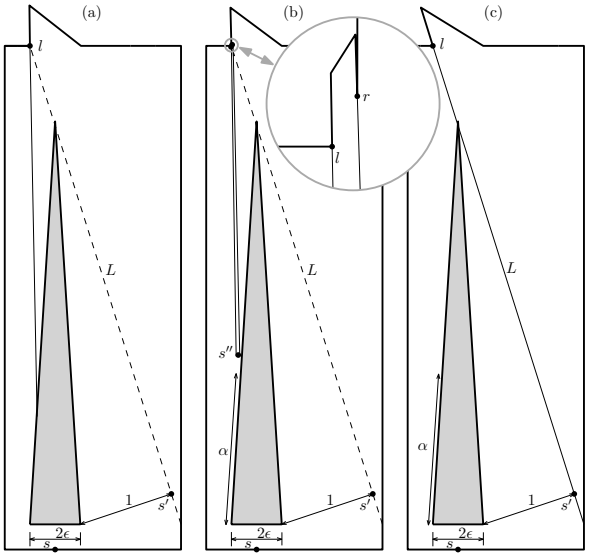


Figure 2: Lower bound example, general case

discovered reflex vertex l. Now, $\mathcal{S}_1$ knows line L and point s' on L, which is the closest point (say in distance 1) on the right side of the hole where $\mathcal{S}_1$ could learn the hidden edge behind vertex l. In fact, after reaching s' the strategy perhaps sees everything and returns to s, Fig.2(c). But directly moving to s' is a fatal decision given the slightly modified situation in Fig.2(b). Here it suffices to move distance α on the left side of the hole to point s''. There $\mathcal{S}_1$ learns both vertex l and some reflex vertex r that is hidden behind l. (Hint: r is very close to l and can be learned only from the left side in a competitive way.)

Any competitive strategy $\mathcal{S}_1$ must be able to handle both possibilities, the optimal strategy chooses the correct side. Therefore, $\mathcal{S}_1$ must try to explore corner l on the left side first, travelling some distance α. Because of the malicious adversary, it still misses the cut of l by a very small distance. Then it will return and explore the cut of l from the right side. If the task is completed in s', the strategy travels a total distance of $2\alpha + 2$. But close to s', it could also learn the existence of vertex r and $\mathcal{S}_1$ has to return once again to the left side of the hole. This yields a total path length of $4\alpha + 2$.

In both cases the quotient of the tour length generated by $\mathcal{S}_1$ and the optimal tour length is a function in α. The monotonically decreasing function $f(\alpha) = \frac{4\alpha+2}{2\alpha}$ describes the competitive ratio, if the cut points to the left side of the hole, Fig.2(b). If the cut of l is learned in s' we have the monotonically increasing function $g(\alpha) = \frac{2\alpha+2}{2}$, Fig.2(c). Comparing both functions to determine the optimal value for α results in $\alpha = \frac{1+\sqrt{5}}{2}$, the golden ratio. We obtain $\frac{3+\sqrt{5}}{2} \approx 2.618$ as lower bound for the competitive ratio. □

In contrast to the orthogonal case a colored hole would make no difference. The problem of learning a reflex vertex in presence of a vision-blocking hole turns out to be a fundamental issue for any strategy that wants to explore the polygon.

3 A competitive strategy for polygons with one colored hole

Since the complete description of the strategy (and even more of the analysis) is very complex, we will focus on the presentation of the key ideas and the global structure of the algorithm. As usual, we assume that the starting point s is on the outer boundary of $\mathcal{P}$ and $\mathcal{T}_{opt}$ denotes a shortest closed watchman tour. The design of our strategy follows the basic principle that in each phase and subphase the generated path should compare to $\mathcal{T}_{opt}$ in a competitive way.

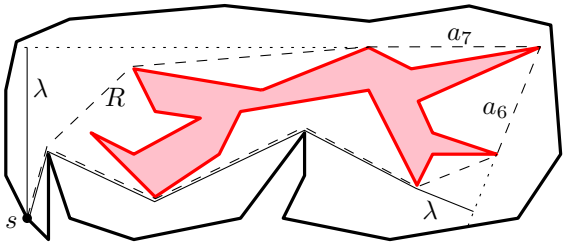


Figure 3: Learning the relative convex hull R

3.1 Phase 1: Learning the relative convex hull

If no point of the (colored) hole is visible from s we start the 26.5-competitive HIKK-strategy until H becomes visible. The next goal will be to look once around the hole, more precisely to learn the boundary R of the relative convex hull of $H \cup \{s\}$. A set M is relatively convex in $\mathcal{P}$ if for each pair of points in M the geodesics (shortest paths) connecting them are included in M. Thus, one can imagine $R = \partial\left(\mathcal{RCH}\left(H \cup \{s\}\right)\right)$ as the shape of a rubber band spanned around H and the starting point inside $\mathcal{P}$ (Fig.3).

Let $a_1, a_2, \ldots, a_n$ be the chain of line segments that represents R in ccw–orientation starting from s. Learning R means to see each a_i (possibly by touching an extension of a_i only). Any strategy that tries to learn R circling H in a fixed orientation will fail to be competitive. Consider e.g. the situation in Figure 2. A strategy that explores R in cw–orientation has to walk up to the top vertex of H on the left side and down again on the right side of H. This can exceed $c \cdot |\mathcal{T}_{opt}|$ for any constant c. Thus, we explore R in rounds with a doubling approach [7]: In an odd/even round k we move in cw–/ccw–orientation going 2^{k-1} length units. In each round there is a last known segment of R ending at a reflex vertex that hides the next segment. Our strategy explores this next segment on a semicircle, see [6].

Remark that $\mathcal{T}_{opt}$ might not circle H completely, but it must visit extensions of all segments a_i. Consider two segments a_i, a_{i+1} incident to a common vertex of H and compute the sum of the shortest path lengths from s to an extension of a_i ccw. around H and to an extension of a_{i+1} cw. around H. Minimizing this sum over all pairs (a_i, a_{i+1}), we get a lower bound λ on $|\mathcal{T}_{opt}|$. In Figure 3 the lower bound λ for learning R and, consequently, for the length of $\mathcal{T}_{opt}$ is realized by (a_6, a_7). Combining this with the factor 2 of the semicircle strategy and the factor 9 of the doubling approach we can show that our strategy learns R with total path length $\leq 36\,|\mathcal{T}_{opt}|$.

3.2 Phase 2: The hybrid approach

H is called a *c-save* hole (for a fixed constant c), if

$$|R| \leq c\,|\mathcal{T}_{opt}| \tag{1}$$

holds. Note that $|R|$ depends on both the shape of H and the position of s. After learning R and computing $\lambda \leq |\mathcal{T}_{opt}|$ we can determine whether $|R| \leq c\lambda$ and decide if the hole is safe.

Theorem 3 *Let $\mathcal{S}_0$ be a $\mathcal{C}$-competitive strategy for exploring simple polygons. After Phase 1 any polygon $\mathcal{P}$ with a c-safe hole H can be explored with total path length $\leq (4c+2)\mathcal{C}\,|\mathcal{T}_{opt}|$.*

Proof. Consider a shortest path p from s to H. If it hits some reflex vertices of $\mathcal{P}$, we translate p into the polygon's interior. This way we introduce an additional barrier that transforms $\mathcal{P}$ into a simple polygon $\mathcal{P}'$ (Fig.4). We show that the strategy $\mathcal{S}_0$ applied to $\mathcal{P}'$ will fulfill the required condition. Therefore it is sufficient to show that there is a closed watchman tour of length $\leq (4c+2)\,|\mathcal{T}_{opt}|$ in the simple polygon $\mathcal{P}'$.

Since the original $\mathcal{T}_{opt}$ might get sliced by the barrier p into several (left and right) pieces and p might limit the visibility too, we use R plus two copies of p (one on the left, the other one on the right side of the barrier) to link together all pieces of $\mathcal{T}_{opt}$ and to restore full vision. Since the construction of a closed tour is based on an Eulerian graph, we ensure that all vertices have even degree using two copies of our structure. Finally, the length of this tour is at most $2\,(|R| + 2|p| + |\mathcal{T}_{opt}|) \leq 4|R| + 2\,|\mathcal{T}_{opt}| \leq (4c+2)\,|\mathcal{T}_{opt}|$. □

The hole H is called *c-critical* as long as we don't know whether condition (1) holds. The hybrid approach consists in implementing the following rule: As soon as we learn that (1) is fulfilled our strategy will switch to the simple polygon mode using Theorem 3.

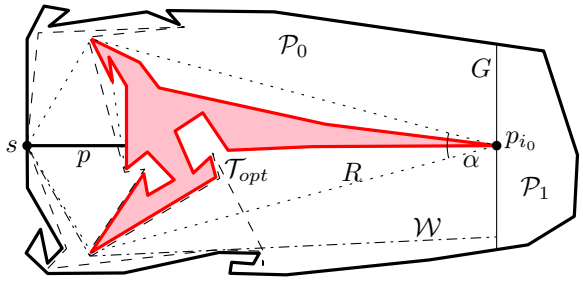


Figure 4: Exploring a polygon with a c-save hole.

Next we discuss how to proceed if the hole is (still) c-critical. Assume that the lower bound λ was established by the edge pair (a_{i_0}, a_{i_0+1}) with the common polygon vertex p_{i_0}. Using some trigonometric reasoning one can show that the angle α between a_{i_0} and a_{i_0+1} is very small. Let G be the line segment perpendicular to the angular bisector of α through p_{i_0} in $\mathcal{P}$. Then G subdivides $\mathcal{P}$ into two simple polygons $\mathcal{P}_0$ and $\mathcal{P}_1$, where $\mathcal{P}_0$ is the one including s and the hole.

Theorem 4 *Let $\mathcal{S}_0$ be a $\mathcal{C}$-competitive strategy for simple polygons, $\mathcal{P}$ a polygon with a c-critical hole, and $\mathcal{W}$ the shortest path from s to the line segment G. Assume that $\mathcal{S}_0$ starts in s to explore the simple polygon $\mathcal{P}_0$ until it finishes the exploration or the total path length exceeds $2 \cdot \mathcal{C} \cdot |\mathcal{W}|$. If $\mathcal{S}_0$ stops with completed exploration of $\mathcal{P}_0$, then the exploration was $\mathcal{C}$-competitive. If $\mathcal{S}_0$ stops with total path length $2 \cdot \mathcal{C} \cdot |\mathcal{W}|$, then $|\mathcal{T}_{opt}| \geq 2 \cdot |\mathcal{W}|$ holds.*

Proof. If $\mathcal{T}_{opt}$ reaches G, then the claim is obvious. Otherwise $\mathcal{T}_{opt}$ is a tour inside of $\mathcal{P}_0$ and the claim follows from the competitiveness of $\mathcal{S}_0$. □

We remark that in the second case the hole becomes 3.1-safe and thus we can apply Theorem 3. Only in the first case the hole might persist critical. Then it remains to explore the polygon $\mathcal{P}_1$ from remote (in $\mathcal{P}_0$). Note that this is just the situation known from the lower bound example, see Theorem 2. To accomplish this goal we use the procedures for exploring groups of left and right vertices from [6]. The point is that it is not clear from which side of the hole an extension of a hidden polygon edge in $\mathcal{P}_1$ has to be explored. Thus, we will apply a doubling approach again. Due to space limitations we have to omit the quite complex analysis, see [3] for details.

In summary, we get the following main result.

Theorem 5 *There is a competitive strategy for exploring polygons with at most one colored hole and given starting point on the outer boundary.*

The competitive factor that follows from the proof is huge (≈ 610) and the corresponding constant c is chosen to be 5.

Finally, an interesting feature of our solution is the qualitative dependency of the proven competitive ratio on geometric properties of the polygon. For example, exploiting the relation between constant c (c–save holes) and the value of angle α leads to an improved competitive factor for polygons without small angles.

4 Future work

We conjecture that our main result holds for the case of an uncolored hole. Moreover, we are sure that the competitive factor can be considerably improved. On the other hand, one could try to generalize the result to polygons with several differently colored holes. We remark that for $h \geq 2$ holes the task of exploring the polygon is no longer equivalent to exploring all edges, compare Fig.5. However, this is true for the case of at most one hole.

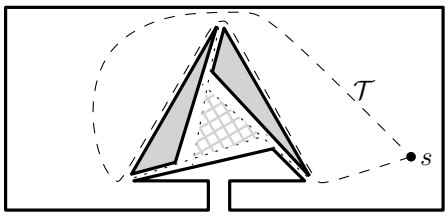


Figure 5: Seeing all boundary edges does not guarantee the exploration of the hole polygon

Acknowledgment. We would like to thank the anonymous referees for their valuable comments.

References

[1] S. Albers, K. Kursawe, and S. Schuierer. Exploring Unknown Environments with Obstacles. In *Proc. SODA*, pages 842–843, Philadelphia, USA, 1999.

[2] X. Deng, T. Kameda, and C. Papadimitiou. How to Learn an Unknown Environment I: The Rectilinear Case. *Journal of the ACM*, 45:215–245, March 1998.

[3] R. Georges. *Online-Erkundung von Polygonen.* Diploma thesis, February 2012, Fachbereich Mathematik und Informatik, Freie Universität Berlin, Germany.

[4] R. Hagius, C. Icking, and E. Langetepe. Lower Bounds for the Polygon Exploration Problem. *Abstracts 20th European Workshop Comput. Geom.*, 2004, Sevilla, Universidad de Sevilla.

[5] M. Hammar, B. J. Nilsson, and M. Persson. Competitive exploration of rectilinear polygons. *Theor. Comput. Sci.*, 354:367–378, April 2006.

[6] F. Hoffmann, C. Icking, R. Klein, and K. Kriegel. The Polygon Exploration Problem. *SIAM J. Comput.*, 31:577–600, February 2002.

[7] R. Klein. *Algorithmische Geometrie.* Springer-Verlag, 2005.

On the Rectilinear Convex Layers of a Planar Set

C. Peláez* A. Ramírez-Vigueras* C. Seara† J. Urrutia‡

Abstract

In this paper we give an optimal $O(n \log n)$ time and $O(n)$ space algorithm to compute the rectilinear convex layers of a set S of n points on the plane. We also compute the rotation of S that minimizes the number of rectilinear convex layers in $O(n^2 \log n)$ time and $O(n^2)$ space.

1 Introduction

Let $S = \{p_1, \ldots, p_n\}$ be a set of n points in the plane in general position. A *quadrant* Q of the plane is the intersection of two closed half-planes whose supporting lines are parallel to the x- and y-axes. We say that a quadrant is *S-free* if it does not contain any point of S in $int(Q)$, where $int(Q)$ denotes the interior of Q. The *rectilinear convex hull* $\mathcal{RH}(S)$ of S is defined as:

$$\mathcal{RH}(S) = \mathbb{R}^2 - \bigcup_{Q \text{ is } S\text{-free}} int(Q).$$

The rectilinear convex hull was introduced in [8] and further studied in [1] and [2]. The rectilinear convex layers of S are defined as follows:

1. The first rectilinear convex layer of S, denoted $\mathcal{L}_1$, is $\mathcal{RH}(S)$. Let S_1 denote the set of elements of S that belong to $\mathcal{RH}(S)$.
2. The i-th rectilinear convex layer $\mathcal{L}_i$ of S is the rectilinear convex hull $\mathcal{RH}(S_i)$, where $S_i = S \setminus \{S_1 \cup \cdots \cup S_{i-1}\}$, where S_j is the set of elements of S in $\mathcal{L}_j$. We stop when $S \setminus \{S_1 \cup \cdots \cup S_i\} = \emptyset$; the first i for which $S \setminus \{S_1 \cup \cdots \cup S_i\} = \emptyset$ is the number of rectilinear convex layers of S.

Figure 1 shows a point set and its rectilinear convex hull. Figure 2 shows an example of the rectilinear convex layers of a point set.

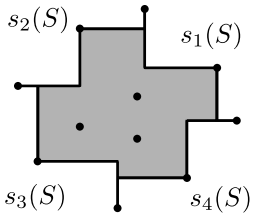


Figure 1: Rectilinear convex hull and its staircases.

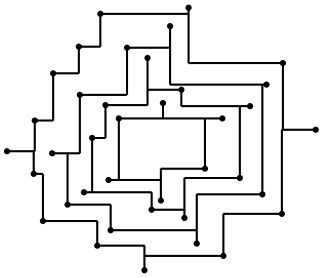
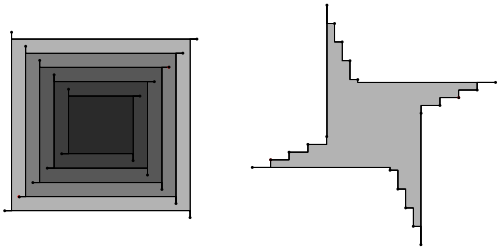


Figure 2: Rectilinear convex layers of a point set.

If we rotate S around the origin the rectilinear convex hull of S, as well as the number of rectilinear convex layers of S changes (Figure 3).



(a) A point set S with $O(n)$ rectilinear convex layers.

(b) One rectilinear convex layer of S by doing a rotation of $\alpha = \frac{\pi}{8}$.

Figure 3: As we rotate S, the number of rectilinear convex layers changes from one to $\frac{n}{4}$.

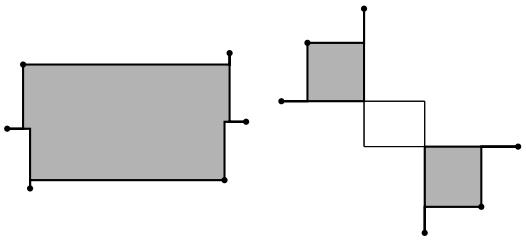
Notice that the rectilinear convex hull of S is not necessarily connected (Figure 4). It is not hard to show that if S contains at least four points in general position, there is always a rotation of S for which its rectilinear convex hull is connected, and its interior is nonempty.

Results. In this paper, we give an optimal $\Theta(n \log n)$ time and $O(n)$ space algorithm to calcu-

*Posgrado en Ciencia e Ingeniería de la Computación, Universidad Nacional Autónoma de México, {canek, adriana.rv}@ciencias.unam.mx. Partially supported by CONACYT of Mexico.

†Dept. de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, carlos.seara@upc.edu. Partially supported by projects MTM2009-07242, Gen Cat DGR2009GR1040, and the ESF EUROCORES programme EuroGIGA-ComPoSe IP04-MICINN Project EUI-EURC-2011-4306.

‡Instituto de Matemáticas, Universidad Nacional Autónoma de México, urrutia@matem.unam.mx. Partially supported by projects MTM2006-03909 (Spain) and SEP-CONACYT of México, Proyecto 80268.



(a) $\mathcal{RH}(S)$ with $\alpha = 0$. **(b)** $\mathcal{RH}(S)$ with $\alpha = \frac{\pi}{4}$.

Figure 4: Two different rectilinear convex hulls.

late the rectilinear convex layers of S. We also give an $O(n^2 \log n)$ time and $O(n^2)$ space algorithm to compute the rotation of S that minimizes (or maximizes) the number of rectilinear convex layers of S. We will omit some proofs due to the lack of space.

Related work. The problem of finding the convex hull of a point set S, as well as its convex layers has been studied in [9]. An optimal algorithm to find them in $O(n \log n)$ was obtained in [4].

The problem of finding the rotation of S that minimizes the area of the rectilinear convex hull was studied in [2] and in [1] it was proved that it can be found in $\Theta(n \log n)$ time and $O(n)$ space. The problem of finding the rectilinear convex hull of a point set S, is closely related to that of finding elements of S which are maximal under vector dominance. Optimal $O(n \log n)$ algorithms to find them in two and three dimensions were obtained in [6]. In [7] an $O(n^{2.688})$ algorithm for $d = n$ was given. An $O(n \log n)$ algorithm for computing the layers of maxima for point sets in three dimensions was presented in [3].

1.1 Preliminaries

For a planar point set S, we define four partial orders on S, $P(S, \preceq_i)$ using the following binary relations on pairs of elements $p_i = (x_i, y_i)$ and $p_j = (x_j, y_j)$ in S:

- $p_i \preceq_1 p_j$ iff $x_i \leq x_j$ and $y_i \leq y_j$.
- $p_i \preceq_2 p_j$ iff $x_i \geq x_j$ and $y_i \leq y_j$.
- $p_i \preceq_3 p_j$ iff $x_i \geq x_j$ and $y_i \geq y_j$.
- $p_i \preceq_4 p_j$ iff $x_i \leq x_j$ and $y_i \geq y_j$.

Let $C = \{q_i = (x_i, y_i); i = 1, \ldots, r\}$ be an antichain of $P(S, \preceq_1)$ such that its elements are sorted according to their x-coordinate. C determines a *staircase* formed by the union of a set of *elbows* obtained as follows: Join q_i to q_{i+1} by an *elbow* passing trough the point (x_i, y_{i+1}), $i = 1, \ldots, r-1$ (Figure 5a). In a similar way, the antichains of $P(S, \preceq_2)$, $P(S, \preceq_3)$, and $P(S, \preceq_4)$ determine staircases (Figure 5b).

We will denote the staircase determined by the maximal elements of $P(S, \preceq_i)$ with $s_i(S)$ (Figure 1). Notice that $s_4(S)$ and $s_1(S)$, and $s_i(S)$ and $s_{i+1}(S)$ share an element of S, $i = 1, 2, 3$. The boundary of the rectilinear convex hull of S is contained in $s_1(S) \cup s_2(S) \cup s_3(S) \cup s_4(S)$, and the set of elements of S that belong to the boundary of its rectilinear convex hull, is the union of the sets of maximal elements of $P(S, \preceq_1)$, $P(S, \preceq_2)$, $P(S, \preceq_3)$, and $P(S, \preceq_4)$.

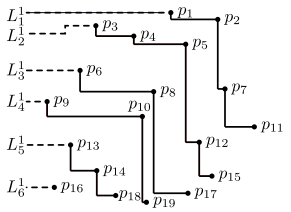
Since each of $s_i(S)$ can be obtained in $O(n \log n)$, the rectilinear convex hull of S can be obtained in the $O(n \log n)$. We will present an algorithm to compute the rectilinear convex layers of S in optimal $\Theta(n \log n)$ time. We will use the layers of maxima points as defined by Buchsbaum and Goodrich [3].

2 The k-level staircases

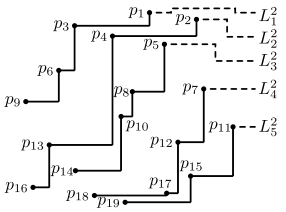
We define the k-level staircases of $P(S, \preceq_1)$ as follows: The 1-level staircase of $P(S, \preceq_1)$, denoted L_1^1, is $s_1(S)$. The i-level staircase of $P(S, \preceq_1)$, denoted L_i^1, is:

$$L_i^1 = s_1\left(S \setminus \bigcup_{j=1}^{i-1} L_j^1\right).$$

The k-level staircases of $P(S, \preceq_2)$, $P(S, \preceq_3)$, and $P(S, \preceq_4)$ are defined in a similar way (Figure 5).



(a) k-level staircases of $P(S, \preceq_1)$.



(b) k-level staircases of $P(S, \preceq_2)$.

Figure 5: k-level staircases of $P(S, \preceq_1)$ and $P(S, \preceq_2)$.

We show now how to compute all the k-level staircases of $P(S, \preceq_1)$ in $O(n \log n)$ time. This problem has also been solved by Buchsbaum and Goodrich [3] with the same complexity. We present a slightly different solution using what we call a *domination tree*. This tree will allow us to solve the problem of finding the rotation of S that minimizes the number of rectilinear convex layers.

2.1 The domination tree

The *domination tree* is a rooted tree T that satisfies the following properties: The elements at depth i in T are the elements of L_i^1, ordered by the x-coordinate. The *parent* of a point $q \in L_{i+1}^1$ is a point $p \in L_i^1$, with smallest x-coordinate such that $p(x) > q(x)$ (Figure 6). To construct the domination tree, we will use an algorithm that we call the BuildTree. We present only an outline of how BuildTree works.

Assume that the elements of S are labelled $p_1, \ldots, p_n$ such that if $i < j$, then the y-coordinate

of p_i is greater than the y-coordinate of p_j. We process the elements of S in this order, that is from top to bottom. Suppose that we have processed $p_1, \ldots, p_{i-1}$, and that up to this point, we have detected k layers of $S_{i-1} = \{p_1, \ldots, p_{i-1}\}$. Suppose that we have an auxiliary array A such that for any $j \leq k$, $A[j]$ contains the rightmost element of the j-th layer of S_{i-1}. The following assertion is not hard to prove: If p belongs to the r-th layer L_r^1 of S, then $A[r-1]$ dominates p_i, and $A[r]$ does not dominate p_i. Using the array A, we can find r in logarithmic time using binary search. It is not hard to see that the parent of p_i in L_{r-1}^1 can be also determined in logarithmic time.

For each point p_i, we will see if it is to the right of the rightmost point in the deepest k-level we have found: If it is so, we will add it to that k-level. Otherwise, we will find the point in the deepest level with the smallest x-coordinate greater than $p_i(x)$. If there is no such point, then p_i will go into a new k-level. Using the auxiliary sorted array A with the rightmost element of every k-level found, we will be able to do this in $O(\log n)$ time for every point, which will yield a total complexity of $O(n \log n)$ time. Thus we have the following results:

Theorem 1 *The complexity of* BuildTree *is* $O(n \log n)$.

Theorem 2 *All of the i-level staircases of $P(S, \preceq_1)$ can be constructed in $O(n \log n)$ time and $O(n)$ space.*

An example of the output of BuildTree is shown in Figure 6.

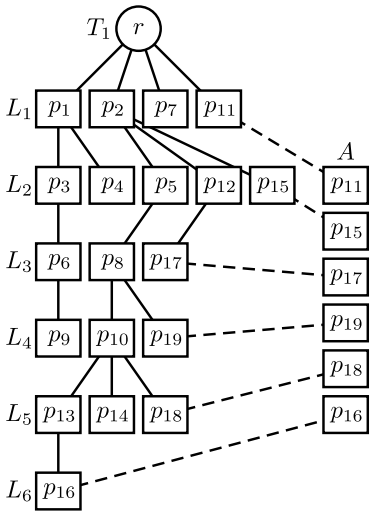


Figure 6: The tree T_1 resulting of running BuildTree on the point set shown in Figure 5.

3 Rectilinear Convex Layers

Clearly BuildTree can also be used to build all the k-level staircases of $P(S, \preceq_1)$, $P(S, \preceq_2)$, $P(S, \preceq_3)$, and $P(S, \preceq_4)$. We now show how to use them to build the rectilinear convex layers of S.

Lemma 3 *Let $S_i = S \setminus \bigcup_{k=1}^{i-1} \mathcal{L}_k$. If $p \in s_j(S_i)$ for some $j \in \{1, 2, 3, 4\}$, then p is in L_i^j.*

Proof. The result is obvious for $i = 1$. Suppose then that for some $i > 1$, $p \in \mathcal{L}_i$. Then $p \in S_i$, and thus $p \in s_j(S_i)$ for some $j \in \{1, 2, 3, 4\}$. Without loss of generality suppose that $p \in s_1(S_i)$. Then it is easy to see that there is a point $q \in s_1(S_{i-1})$ such that $p \preceq_1 q$. Therefore, p is in the i-level staircase of $P(S, \preceq_1)$. □

Corollary 4 *If $p \in \mathcal{L}_i$, then either $p \in L_i^1$, $p \in L_i^2$, $p \in L_i^3$, or $p \in L_i^4$.*

Lemma 5 *Let $p \in S$ be such that $p \in L_{j_i}^i$ for $i = 1, 2, 3, 4$. If $j = \min\{j_1, j_2, j_3, j_4\}$, then p is in the j-th rectilinear convex layer $\mathcal{L}_j$ of S.*

Theorem 6 *Let S a planar point set in general position. Computing the rectilinear convex layers of S can be done in optimal $\Theta(n \log n)$ time and $O(n)$ space.*

The time optimality follows from the fact that computing the rectilinear convex hull of a point set requires $\Omega(n \log n)$ time [1].

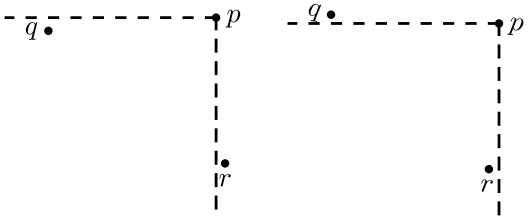
3.1 The rotation of S with the minimum number of layers

The convex hull and the convex layers of a planar point set are invariant under rotations of the point set. This is not the case when we consider the rectilinear convex hull and the rectilinear convex layers. Thus, we now study the following problem: Given a planar point set S, find a rotation of S around the origin that minimizes (or maximizes) the number of rectilinear convex layers of S.

By using the four domination trees obtained by BuildTree to compute the rectilinear convex layers of S we can produce this minimum (or maximum) in $O(n^2 \log n)$ time and $O(n^2)$ space: We only need to find the critical directions where the k-level staircases in our domination tree, and therefore the rectilinear convex layers of S, may change.

These critical directions are defined when the line joining two points in S becomes horizontal, or vertical. When this happens, the partial order $P(\preceq_i, S)$ can change, and this can cause the rectilinear convex layers and the rectilinear convex hull to change (Figure 7).

We compute first T_1, and the rectilinear convex layers of S. Next, we will generate the set $\mathcal{D}$ of critical



(a) Rotation $\alpha = 0$. **(b)** Rotation $\alpha = \frac{\pi}{24}$.

Figure 7: In the rotation $\alpha = 0$, $q \preceq_1 p$, and p and r are not comparable. In the rotation $\alpha = \frac{\pi}{24}$, q and p are not comparable, and $r \preceq_1 p$.

directions in $O(n^2)$ time and space, and sort them in $O(n^2)$ time using standard techniques. We do the sorting in such a way that when we pass through a critical direction $\gamma_i \in \mathcal{D}$ we know whether the two points p and q defining $\gamma_i \in \mathcal{D}$ become horizontal or vertical aligned according to the current orientation of the axes. Let p and q be the two points defining $\gamma_i \in \mathcal{D}$ and assume that the line joining p and q becomes horizontal (Figure 7). If p is not the parent of q or the other way around, we will continue to the next critical direction. Otherwise, q jumps from the j-level staircase to the $(j-1)$-level staircase, so we remove q from its staircase it belongs to, and add it to that one level above it. We repeat this recursively for all the descendants of q.

If the line joining p to q becomes vertical, we proceed as follows: If both p and q are not in the same j-level staircase and next to each other in that staircase, continue to the next critical direction. Otherwise, suppose that p is before r in their j-level staircase. Then r jumps down from the j-level staircase to the $(j+1)$-level staircase: We create the $(j+1)$-level staircase if it does not exists.

Lemma 7 *When going from γ_i to γ_{i+1}, a point $p \in S$ either preserves its depth, or it only changes by* 1.

Lemma 8 *When a point q jumps up from the j-level staircase to the $(j-1)$-level staircase, all its descendants in the T_1 tree also jump up one staircase level, and not other point in S does.*

The result is similar for the other three trees.

Theorem 9 *At the end of the i-th cycle of the loop, the L_j array will be the j-level staircase L_j^1 of S in the rotation γ_i.*

To prove the time complexity of the algorithm we need the following non trivial lemma, given without proof.

Lemma 10 *The number of times $p \in S$ changes its rectilinear depth as S rotates is at most $O(n)$.*

Theorem 11 *Let S a planar point set in general position. Computing the rotation of S that minimizes the number of layers can be done in $O(n^2 \log n)$ time and $O(n^2)$ space.*

This algorithm can be modified to solve other problems such as computing the rotation that: (1) maximizes the number of rectilinear convex layers of S, (2) produces the rectilinear convex hull with the largest (smallest) number of vertices.

References

[1] C. Alegría-Galicia, T. Garduño, A. Rosas-Navarrete, C. Seara, and J. Urrutia. Rectilinear convex hull with minimum area. *The XIV Spanish Meeting on Computational Geometry*, 2011.

[2] S. W. Bae, C. Lee, H. K. Ahn, S. Choi, K. Y. Chwa. Computing minimum-area rectilinear convex hull and L-shape. *Computational Geometry: Theory and Applications*, 42(9), 903–912, 2009.

[3] A. L. Buchsbaum, and M. T. Goodrich. Three-dimensional layers of maxima. *Algorithmica*, 39(4), 275–286, 2004.

[4] B. Chazelle. On the convex layers of a planar set. *IEEE Transactions on Information Theory*, 31, 509–517, 1985.

[5] B. Chazelle, L. Guibas, and D. T. Lee. The power of geometric duality. *Proc 24th IEEE Ann. Symp. Foundations of Computer Science*, 217–225, 1983.

[6] H. T. Kung, F. Luccio, and F. P. Preparata. On finding the maxima of a set of vectors. *J. ACM*, 22(4), 469–476, 1975.

[7] J. Matoušek. Computing dominances in E^n. *Information Processing Letters*, 38(5), 277–278, 1991.

[8] T. Ottmann, E. Soisalon-Soininen, and D. Wood. On the definition and computation of rectilinear convex hulls. *Information Sciences*, 33(3), 157–171, 1984.

[9] M. H. Overmars, and J. van Leeuwen. Maintenance of configurations in the plane. *Journal of Computer and System Sciences*, 23(2), 166–204, 1981.

[10] C. Peláez, A. Ramírez-Vigueras, C. Seara, and J. Urrutia. Weak separators, vector dominance, and the dual space. *XIV Spanish Meeting on Computational Geometry*, 2011.

[11] M. I. Shamos. Computational Geometry. *Ph. D. dissertation, Yale Univ., New Haven, CT, 1978.*

On balanced 4-holes in bichromatic point sets

S. Bereg * J.M. Díaz-Báñez † R. Fabila-Monroy ‡ P. Pérez-Lantero § A. Ramírez-Vigueras ¶
T. Sakai ‖ J. Urrutia ** I. Ventura †

1 Introduction

Let $S = R \cup B$ be a set of n points in general position in the plane. The elements of R and B will be called, respectively, the *red* and *blue* elements of S. A k-hole of S is a simple polygon with k vertices, all in S, and containing no element of S in its interior. A 4-hole of S is balanced if it has two blue and two red vertices. In this paper, we characterize the set of bicolored points $S = R \cup B$ that have balanced *convex* 4-holes. We also show that if the 4-holes of S are allowed to be non-convex, and $|R| = |B|$, then S always has a quadratic number of balanced 4-holes.

The study of k-holes in colored point sets was introduced by Devillers et al. [2]. They obtained a bichromatic point set $S = R \cup B$ with 18 points that contains no convex monochromatic 4-hole. Recently, Hummer and Seara obtained a bichromatic point set with 36 points that does not contain monochromatic 4-holes [3]. This result was improved by Koshelev [4] to 46. Devillers et al. [2] also proved that every 2-colored Horton set with at least 64 elements contains an empty monochromatic 4-hole. In the same paper the following conjecture is posed: Every sufficiently large bichromatic point set contains a monochromatic convex 4-hole. This conjecture remains open. On the other hand, Aichholzer et al [1] proved that any sufficiently large bichromatic point set always contains a *not necessarily convex* monochromatic 4-hole.

*University of Texas at Dallas, besp@utdallas.edu.

†Departamento de Matemática Aplicada II, Universidad de Sevilla, España, {dbanez, iventura}@us.es. Partially supported by Spanish MEC grant MTM2009-08625 and ESF 279 EUROCORES programme EuroGIGA, CRP ComPoSe: grant EUI-EURC-2011-4306.

‡Departamento de Matemáticas, Cinvestav, ruyfabila@math.cinvestav.edu.mx.

§Escuela de Ingeniería Civil e Informática, Universidad de Valparaíso, Chile, pablo.perez@uv.cl. Partially supported by Spanish MS grant MTM2009-08625 and grand FONDECYT 11110069.

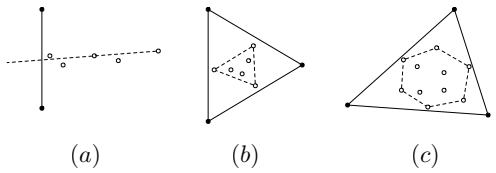
¶Posgrado en Ciencia e Ingeniería de la Computación, Universidad Nacional Autónoma de México, adriana.rv@ciencias.unam.mx. Partially supported by CONACYT of México, Proyecto 80268.

‖Research Institute of Educational Development, Tokai University, 2-28-4 Tomigaya, Shibuyaku, Tokyo 151-8677, Japan, sakai@tokia-u.jp.

**Instituto de Matemáticas, Universidad Nacional Autónoma de México, México D.F. México, urrutia@matem.unam.mx. Partially supported by projects MTM2009-08625(Spain) and SEP-CONACYT of México, Proyecto 80268.

2 Balanced Convex 4-holes

For any two points p and q on the plane, $\overline{pq}$ and $\ell(p, q)$ will denote, respectively, the line segment joining p and q, and the line determined by them. If p and q are both blue points (resp. red points), $\overline{pq}$ and $\ell(p, q)$ will be called a *blue edge*, and a *blue line* of S (resp. *red edge*, and *red line* of S). Given a set X, $CH(X)$ will denote the convex hull of X. In all of our figures, blue points are represented with non-solid small circles, whereas red points are represented with solid small circles. Blue edges will be drawn with dotted line segments, and red edges with solid line segments.



(a) (b) (c)

Figure 1: Point sets with no 4-balanced convex holes.

Clearly not all bicolored point sets have a balanced convex 4-hole, see Figure 1(a), (b), and (c). The number of blue points within the blue triangle and hexagon in Figure 1(b), and (c) can be arbitrarily large. We now proceed to characterize bicolored point sets which contain balanced convex 4-holes. We assume that $|R|, |B| \geq 2$.

Lemma 1 *If S contains red and a blue edges that intersect, then S contains a balanced convex 4-hole.*

Proof. Suppose that S has red and blue edges that intersect. Choose a red edge $\overline{ab}$ and a blue edge $\overline{cd}$ such that the convex quadrilateral Q with vertex set $\{a, b, c, d\}$ is of minimum area.

If Q is not a balanced 4-hole of S, then there is a point in S contained in the interior of Q. Suppose w.l.o.g. that Q contains a red point $e \in S$, and that $\overline{ae}$ intersects $\overline{cd}$. Then $\{a, e, c, d\}$ is the set of vertices of a balanced convex quadrilateral with area smaller than that of Q, a contradiction, see Figure 2. □

Two cases arise: R and B are linearly separable, or their are not. We analyze first the case when R and B are not linearly separable.

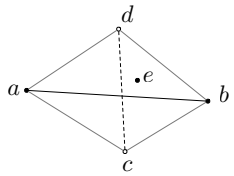


Figure 2: Crossing segments. Segment $\overline{ab}$ can be replaced by segment $\overline{ae}$.

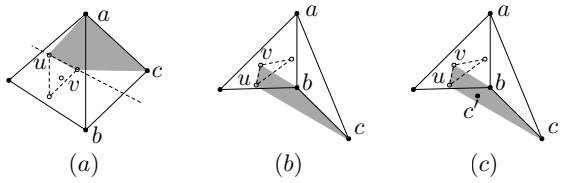


Figure 3: Two possible quadrilaterals including the blue points: convex and non-convex.

2.1 R and B are not linearly separable

Suppose that R and B are not linearly separable. We prove now the next result:

Theorem 2 *Let $S = R \cup B$, such that R and B are not linearly separable. Then S has a balanced 4-hole if and only if either of the following three conditions holds:*

1. *$CH(B) \subset CH(R)$, $|R| \geq 4$, $|B| \geq 2$,*
2. *$CH(R) \subset CH(B)$, $|B| \geq 4$, $|R| \geq 2$,*
3. *$CH(B)$ and $CH(R)$ overlap.*

Proof. Suppose first that $CH(B) \subset CH(R)$, and R and B have at least four and two elements, respectively. Consider a triangulation T of R. By Lemma 1, all blue points have to be in one triangle t of T, for otherwise there would be a red and a blue edge of S that cross each other. If B has exactly two elements, then it is easy to see that two vertices of t, together with the elements of B, form a balanced 4-hole. Suppose then that B has at least three elements.

Since R has at least four points, T has at least two triangles, one of which, call it t', shares an edge $\overline{ab}$ with t. Let $Q = t \cup t'$, and suppose first that Q is convex, see Figure 3(a). If the line determined by two consecutive elements of $CH(B)$, say u and v, intersects $\overline{ab}$, then two vertices of t' together with u and v form a balanced 4-hole, see Figure 3(a).

Suppose then that Q is not convex, and let c be the third vertex of t', see Figure 3(b). Since B has at least three vertices, there are at least two blue vertices on one of the half planes determined by the line $\ell(b, c)$. It is easy to see that we can always choose two of them, call them u and v, such that the quadrilateral Q' with vertices u, v, b, c is convex, and contains no blue point in its interior, see Figure 3(b). It might happen that Q' contains a red point in its interior. In such a case, we can always choose a red point c' in the interior of Q' such that the quadrilateral with vertices u, v, b, c' is a balanced 4-hole, see Figure 3(c). Our result follows. It is worth pointing out that if $CH(B) \subset CH(R)$ but R has only three elements, our result is not true. Counterexamples of this are the point sets in Figure 1(b) and (c).

Finally observe that when the convex hulls of B and R overlap there is a red and a blue edge that intersect. By Lemma 1 we can conclude that S contains a balance 4-hole. □

2.2 R and B are linearly separable.

Suppose that R and B are in convex position, that they are linearly separable, and that there is a line ℓ that separates R and B such that the elements in R are to the left of ℓ, and the elements in B to its right. We will assume w.l.o.g. that ℓ is vertical. Assume also w.l.o.g. that there is a horizontal line ℓ' that is a supporting line of $CH(B)$ and $CH(R)$ that intersects them at a single point, and that R and B are contained in the closed half-plane determined by ℓ' and above it. Let r_1 and b_1 be the lowest elements of R and B respectively. Label the elements of R as $r_1, \ldots, r_m$ in the anti-clockwise direction around $CH(R)$, starting at r_1. In a similar way, label the elements of B as $b_1, \ldots, b_n$ in the clockwise direction around $CH(B)$ staring at b_1.

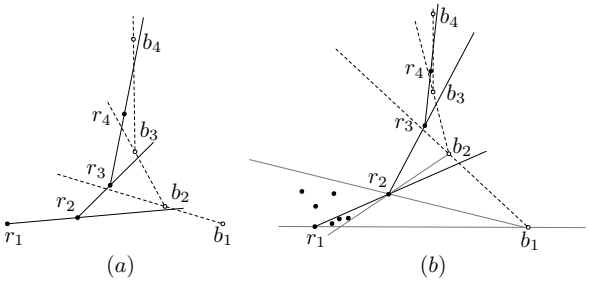


Figure 4: A funnel, and a funnel with a red tail.

We say that $S = R \cup B$ is a *funnel* if the following conditions hold: $\ell(r_1, r_2)$ intersects the segment $\overline{b_1 b_2}$, or $\ell(b_1, b_2)$ intersects the segment $\overline{r_1 r_2}$. In the first case, the following conditions must hold:

a) For each i such that the line $\ell(b_i, b_{i+1})$ and the segment $\overline{r_{i+1} r_{i+2}}$ exist, $\ell(b_i, b_{i+1})$ intersects $\overline{r_{i+1} r_{i+2}}$.

b) For each i such that $\ell(r_i, r_{i+1})$ and $\overline{b_i b_{i+1}}$ exist, $\ell(r_i, r_{i+1})$ intersects $\overline{b_i b_{i+1}}$.

If $\ell(b_1, b_2)$ intersects $\overline{r_1 r_2}$, then $\ell(r_i, r_{i+1})$ intersects $\overline{b_{i+1} b_{i+2}}$, and $\ell(b_i, b_{i+1})$ intersects $\overline{r_i r_{i+1}}$.

The point set in Figure 4(a) is a funnel in which each of R and B contains four elements. It is easy

to see that if $R \cup B$ is a funnel, then $||R| - |B|| \leq 1$, and that if we choose a red edge and a blue edge of S, the convex hull of their vertices is a triangle, or it is a convex quadrilateral that contains at least a point of S in its interior.

Suppose that S is a funnel such that $\ell(r_1, r_2)$ intersects $\overline{b_1 b_2}$. Let $\mathcal{T}$ be the unbounded region bounded by $\ell(b_1, r_1)$, $\ell(b_1, r_2)$, and $\ell(b_2, r_2)$. A set of red points R' is called a *red tail* of S if all the elements of R' belong to the interior of $\mathcal{T}$, see Figure 4(b). A blue tail is defined in a similar way when $\ell(b_1, b_2)$ intersects $\overline{r_1 r_2}$.

A *double funnel* is defined as follows: Let R and B be separable point sets in convex position, and let ℓ and ℓ' be as above. Suppose also that there is exactly one edge in the convex hull of R or B such that the line containing this edge separates R and B. Without loss of generality, assume that this edge is red, and that the blue points lie to the right of this line, see Figure 5.

Label the elements of R as $r_1, \ldots, r_n$ in the anti-clockwise direction around $CH(R)$, starting at the lowest element of R. Label the elements of B as $b_1, \ldots, b_m$ in the clockwise direction around $CH(B)$, starting again at the lowest element of B. Suppose that the edge of $CH(R)$ such that the line containing it that separates R and B joins r_i and r_{i+1} for some i.

Let b_i be the closest element of B to $\overline{r_i r_{i+1}}$. Consider the following sets of points: $R_1 = \{r_1, \ldots, r_{i+1}\}$, $R_2 = \{r_i, \ldots, r_n\}$, $B_1 = \{b_1, \ldots, b_i\}$, $B_2 = \{b_i, \ldots, b_m\}$.

We say that $R \cup B$ is a *double funnel* if the sets $B_1 \cup R_1$ and $B_2 \cup R_2$ are funnels, see Figure 5. Note that one of them, is an upside down funnel!

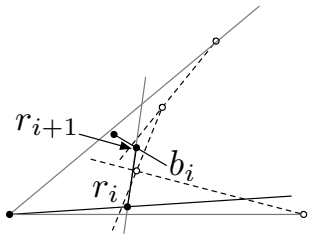


Figure 5: A double funnel.

In a similar way as we defined a tail for a funnel, we can define a tail of a double funnel. A tail of $R \cup B$ is a tail of $B_1 \cup R_1$, or a tail of $B_2 \cup R_2$. The following result is given without proof:

Theorem 3 *If R and B are linearly separable, then $S = R \cup B$ contains no convex balanced 4-holes if and only if S is:*

- *a funnel with or without tail*
- *a double funnel, with one or two tails such that:*
  1. *The two tails have the same color.*
  2. *The double funnel can be splitted into two funnels f_1 and f_2 such that one of them, say f_2 has at most two points of each color.*

3 Non-convex balanced 4-holes

It is not hard to see that if R and B have at least two elements each, then S always contains 4-balanced holes which are not necessarily convex. In fact, it is easy to see that a double funnel contains $O(n^2)$ balanced non-convex 4-holes. In this section we will prove that if $|R| = |B| = n$, then $S = R \cup B$ always contains a quadratic number of balanced, not necessarily convex 4-holes. In the rest of this section, we will assume that $|R| = |B| = n$.

We give our proof only for the case when R and B are linearly separable. Our proof can be modified to prove that for any two points sets R and B, $R \cup B$ always has a quadratic number of balanced holes. We omit these not so trivial changes.

Suppose that there is a horizontal line ℓ that separates R from B, and that the elements in R are above ℓ, and the elements of B below it. We further assume that the y-coordinates of all of the elements of S are different.

Given two points p and q, $p \to q$ will denote the ray staring at p, and passing through q. We now color edges joining blue and red points as follows:

An edge $\overline{rb}$ joining a red point r to a blue point b is colored *green* if it is an edge, or a diagonal of a balanced 4-hole of S. An edge $\overline{rb}$, not colored green, is colored *red* if when we rotate the ray $r \to b$ around r in the clockwise direction, it hits a red point before it hits a blue point. It is not hard to see that in this case, if we rotate $r \to b$ in the anti-clockwise direction, it also hits a red point before it hits a blue point, for otherwise $\overline{rb}$ would be green. Moreover if we rotate $b \to r$ in the clockwise, or anti-clockwise direction, it will also hit red points before it hits blue points. See Figure 6.

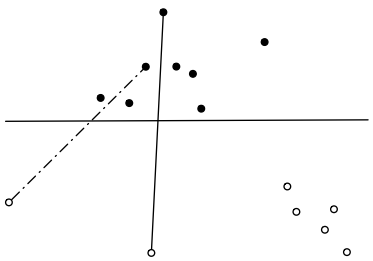


Figure 6: A red and a green edge.

In a similar way we color $\overline{rb}$ *blue* if when we rotate $r \to b$ around b it hits a blue point before it hits a red

point. We will prove that the number of green edges is quadratic. Theorem 9 follows easily from this.

Let r be a red point. Sort and relabel the blue points from left to right in the anti-clockwise direction around r as $\{b_1, \ldots, b_n\}$. The following lemmas, are given without proof:

Lemma 4 *There are no consecutive edges $\overline{rb_i}$ and $\overline{rb_{i+1}}$ such that one is blue, and the other is red.*

In other words between a red and a blue edge there is a green edge.

Let i such that $\overline{rb_i}$ and $\overline{rb_{i+1}}$ are both red, and let Δ be the triangle bounded by ℓ, $\overline{rb_i}$, and $\overline{rb_{i+1}}$.

Lemma 5 *Δ contains at least three red points of S, see Figure 7.*

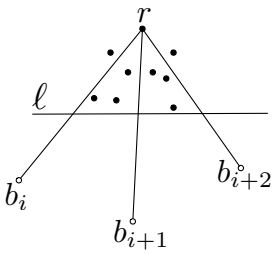


Figure 7

Observe that this Lemma implies that the set $\{\overline{rb_i}; i = 1, \ldots n\}$ contains at most $\frac{n}{3}$ consecutive red edges.

Suppose next that we have a block $\mathcal{B}$ of k consecutive red edges $\overline{rb_i}$, $\ldots$, $\overline{rb_{i+k-1}}$ incident to r, such that $\overline{rb_{i-1}}$ and $\overline{rb_{i+k}}$ are green. Then we can associate to $\mathcal{B}$ at least $3k-1$ red points that lie in the triangle bounded by ℓ, $\overline{rb_i}$, and $\overline{rb_{i+k-1}}$, plus the red points hit by $\overline{rb_i}$ and $\overline{rb_{i+k-1}}$ when we rotate them in the clockwise, and anti-clockwise direction respectively.

It is easy to see that the sets of red points associated to different blocks of consecutive red edges incident to r are disjoint.

Suppose now that the red edges incident to r are grouped into s blocks of consecutive red edges with cardinalities $t_1, \ldots, t_s$.

Label the red points $r_1, \ldots, r_n$ from bottom to top according to their y-coordinate. The next result follows:

Lemma 6 *The number of red edges incident to r_i is at most $(3t_1 - 1) + \cdots + (3t_s - 1) - 2$.*

The worst case is when each $t_i = 1$, in which case the previous Lemma yields $2s - 2$. From here we get:

Lemma 7 *The number of red edges incident to r_i, is at most $\lfloor \frac{i-1}{2} \rfloor + 1$.*

Thus we have:

Theorem 8 *There are at most $2(1 + \cdots \lfloor \frac{n-1}{2} \rfloor) \sim \frac{n^2}{4}$ red edges.*

A simmetric argument shows that the number of blue edges is $\sim \frac{n^2}{4}$.

Since there are exactly n^2 edges joining red and blue points, it follows that at least approximately half of them are green, which proves:

Theorem 9 *The number of balanced bichromatic 4-holes of S is at least $\frac{t}{6}$, were $t \sim \frac{n^2}{2}$.*

Our argument can be modified to prove:

Theorem 10 *Let $S = R \cup B$ be a set with $2n$ points, n red, and n blue. Then S always has at least a quadratic number of balanced bichromatic 4-holes.*

We observe that our bound is asymptotically tight, as in examples as that shown in Figure 8, any balanced 4-hole has to be convex, and there are only a quadratic number of these 4-holes.

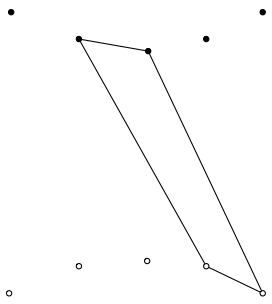


Figure 8: Any balanced 4-hole uses two consecutive red, and two consecutive blue points.

References

[1] O. Aichholzer, T. Hackl, C. Huemer, F. Hurtado and B. Vogtenhuber. *Large bichromatic point sets admit empty monochromatic 4-gons.* SIAM Journal on Discrete Mathematics (SIDMA), Volume 23, Issue 4, pp. 2147-2155, 2010.

[2] O. Devillers, F. Hurtado, G. Kàrolyi and C. Seara. *Chromatic variants of the Erdös-Szekeres Theorem.* Computational Geometry Theory and Applications, Volume 26, Issue 3, pp. 193-208, 2003.

[3] C. Huemer and C. Seara. *36 Two-Colored Points with no Empty Monochromatic Convex Fourgons.* Geombinatorics, Volume XIX(1), pp. 5–6, July 2009.

[4] V. Koshelev. *On Erdös-Szekeres problem and related problems.* ArXiv e-prints, 2009.

One-to-one Point Set Matchings for Grid Map Layout*

David Eppstein† Marc van Kreveld‡ Bettina Speckmann§ Frank Staals‡

Abstract

We study several one-to-one point set matching problems which are motivated by layout problems for grid maps. We are given two sets A and B of n points in the plane, and we wish to compute an optimal one-to-one matching between A and B. We consider two optimisation criteria: minimising the sum of the L_1-distances between matched points, and maximising the number of pairs of points in A for which the matching preserves the directional relation. We show how to minimise the total L_1-distance under translation or scaling in $O(n^6 \log^3 n)$ time, and under both translation and scaling in $O(n^{10} \log^3 n)$ time. We further give a 4-approximation for preserving directional relations by computing a minimum L_1-distance matching in $O(n^2 \log^3 n)$ time.

1 Introduction

We study two point set matching problems motivated by geographic information visualization, specifically by the layout of *grid maps*. Grid maps are a special type of single-level spatial treemap [11]. The input is a set of geographic locations or regions (represented by their centroids) which are mapped one-to-one to a grid of equal-sized rectangles (see Fig. 1). Within the rectangle corresponding to each region or location, additional information can be displayed, see for example the London *BikeGrid* (`gicentre.org/bikegrid`). To aid the user in identifying the regions in the grid map it is essential that the (relative) positions of the input are preserved as much as possible.

We are now given a set of n points A (the geographic locations or region centroids) and a set of n points B (the centers of the grid cells) and we want to compute an optimal one-to-one matching ϕ between A and B. We consider two optimisation criteria: (i) minimising the sum of the L_1-distances between matched points under translation, scaling, and both scaling and translation of point set A, and (ii) maximising the number of pairs of points in A for which the matching preserves the *directional relation*. That is, if a point a_2 lies northwest of point a_1, then we would like $\phi(a_2)$ to lie northwest of $\phi(a_1)$ as well.

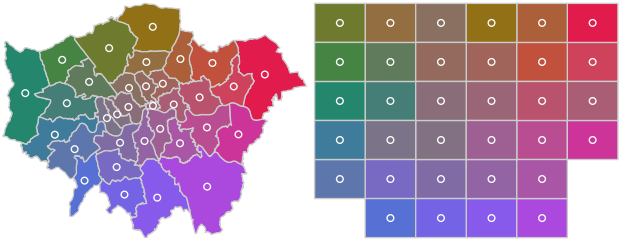


Figure 1: A grid map of the London boroughs.

Related Work. Point set matching problems have been studied extensively. We review some of the minimum distance matching methods that are most related to our work. A more extensive survey of existing methods is presented by Alt and Guibas [1], or more recently by Veltkamp and Hagedoorn [10].

Atkinson [4] presents an algorithm for computing a one-to-one matching in case the input point sets are assumed to be copies the same point set, possibly with affine transformations applied on them. His algorithm runs in $O(n \log n)$ time.

Hong and Tan [7] present a similar approach which can also be used when the points sets are not exact copies of each other. They allow a point p to be matched to q if q lies in the *error area* $E(p)$ of p: a convex polygon for which the distance between p and the closest point on the boundary of $E(p)$ and the distance between p and the furthest point on the boundary differs by at most a constant factor. It is assumed that p is contained in $E(p)$, and that for any point q we can check if $E(p)$ contains q in constant time. Furthermore, all error regions are pairwise disjoint. Sprinzak and Werman [8] extend Hong and Tan's method for point sets in arbitrary dimensions.

Alt et al. [2] present algorithms for several types of point set matching problems. Their algorithms can handle both the L_2-metric, and the L_∞-metric. In case all points have disjoint error areas with radius ϵ (a disk with radius ϵ in the Euclidean case and a square with side lengths 2ϵ in case of the L_∞-metric) they present an $O(n \log n)$ time algorithm to compute the minimum distance matching, and the translation that yields this matching.

*B. Speckmann and F. Staals were supported by the Netherlands' Organisation for Scientific Research (NWO) under project no. 639.022.707 and 612.001.022, respectively.

†Department of Computer Science, University of California, Irvine, `eppstein@ics.uci.edu`.

‡Department of Information and Computing Sciences, Utrecht University, `m.j.vankreveld@uu.nl`, and `f.staals@uu.nl`.

§Department of Mathematics and Computer Science, TU Eindhoven, `speckman@win.tue.nl`.

In case the error regions are not given, but instead we should decide whether or not there is a translation with corresponding matching such that the distance between a pair of matched points is at most ϵ, Alt et al. [2] present an $O(n^6)$ time algorithm. When we actually want to find the smallest such ϵ the running time increases to $O(n^6 \log n)$. Finally, Alt et al. [2] show that if we want to allow rotation and reflection as well we can solve the decision version of the problem in $O(n^8)$ time.

Vaidya [9] studies computing a minimum weight complete matching in a graph G of $2n$ points in which the weights are given by the distance between the points. He shows that for the L_1-, L_2-, and L_∞-distance an optimal matching can be computed in $O(n^{2.5} \log^4 n)$ time, or $O(n^{2.5} \log n)$ in case G is bipartite. For the L_1- and L_∞-metrics this can be improved to $O(n^2 \log^3 n)$. This is the algorithm we will use to compute basic one-to-one matchings without translation or scaling.

Efrat and Itai [6] investigate *bottleneck matching*. They show how to find a one-to-one matching that minimises the L_∞-distance in $O(n^5 \log^2 n)$ time. This improves the results of Alt et al. [2] by almost a factor n. For the decision version of the problem their algorithm runs in $O(n^5 \log n)$ time.

There is also a point set matching approach by Cohen and Guibas [5] which uses the *Earth Mover's Distance*. In this setting each point has a certain weight. The amount of work to match a point $a \in A$ with a point $b \in B$ is determined by the distance between a and b and the weight of a that is matched to b. The earth mover's distance expresses the minimum work required to match A and B. When using the L_2^2-distance Cohen and Guibas [5] present an algorithm that computes an optimal transformation of A and a minimum distance matching. For other distances their method yields only a locally optimal transformation and matching.

Alt et al. [3] propose a probabilistic method for matching planar regions. Their algorithm picks a random set of points A in one region and a random set of points B in the other. It then computes a transformation (consisting of a translation and a rotation) such that the area of overlap between the planar regions is close to maximal. They argue that it may be possible to extend their approach to compute a minimum distance matching under affine transformations.

Organisation. In the next section we consider minimum L_1-distance matchings. We first compute an optimal matching under translation, and then adapt our approach to compute a minimum distance matching under scaling, and both translation and scaling. In Section 3 we then consider matchings that preserve directional relations. Interestingly, we use an algorithm that computes a minimum L_1-distance matching to obtain a 4-approximation for this problem.

2 Minimising L_1-distance

We first define some notation. For a point $a = (a_x, a_y)$ and a translation $t = (t_x, t_y)$ we write $a + t = (a_x + t_x, a_y + t_y)$. We also use this notation for a set of points: $A + t = \{a + t \mid a \in A\}$. Similarly, for a scaling $\lambda = (\lambda_x, \lambda_y)$ we write $\lambda a = (\lambda_x \cdot a_x, \lambda_y \cdot a_y)$. A transformation (either translation or scaling) in which both components have the same value c we denote by $\overline{c} = (c, c)$.

Let $\phi : A \to B$ be a one-to-one matching for the point sets A and B, let t be a translation and let λ be a scaling. Then we define the *total distance* of matching ϕ with translation t and scaling λ as

$$D(\phi, t, \lambda) = \sum_{a \in A} d(\lambda a + t, \phi(a))$$

where $d(a, b)$ denotes the L_1-distance between a and b. We can decompose d into a horizontal and a vertical component: $d(a, b) = x(a, b) + y(a, b)$ with $x(a, b) = |a_x - b_x|$ and $y(a, b) = |a_y - b_y|$. We generalise this notion to D, which gives us $D(\phi, t, \lambda) = X(\phi, t, \lambda) + Y(\phi, t, \lambda)$.

Additionally, we define $D_{\mathcal{T}}(\phi, t) = D(\phi, t, \overline{1})$, $D_\Lambda(\phi, \lambda) = D(\phi, \overline{0}, \lambda)$ and $D_I(\phi) = D(\phi, \overline{0}, \overline{1})$. The functions $X_{\mathcal{T}}, Y_{\mathcal{T}}, X_\Lambda$, etc. are defined accordingly.

We now want to find a matching together with a translation and/or scaling that minimises the total distance. More formally, let Φ be the collection of all one-to-one matchings between A and B, let $\mathcal{T}$ be the collection of all translations, and let Λ be the collection of all scalings, then we try to find a matching $\phi^* \in \Phi$, a translation $t^* \in \mathcal{T}$, and a scaling $\lambda^* \in \Lambda$ such that

$$D(\phi^*, t^*, \lambda^*) = \min_{\phi \in \Phi, t \in \mathcal{T}, \lambda \in \Lambda} D(\phi, t, \lambda).$$

Minimising L_1 under translation. To find a minimum distance matching under translation, i.e. a matching that minimises $D_{\mathcal{T}}$, we identify a (finite) set of translations $T \subset \mathcal{T}$ that contains an optimal translation. We then use one of the existing one-to-one matching algorithms for each translation in T to compute an optimal matching.

We say a translation t is a *horizontal translation* if and only if $t = (c, 0)$ for some $c \in \mathbb{R}$. Two point sets A and B are *x-aligned* if (and only if) there is a point $a \in A$ and a point $b \in B$ with $a_x = b_x$. We define *vertical translation* and *y-aligned* symmetrically.

We now observe that for any matching ϕ between point sets A and B that are not x-aligned we can decrease $D_{\mathcal{T}}(\phi)$ by x-aligning A and B (Fig. 2). Hence:

Lemma 1 *Let A and B be two non x-aligned sets of n points in the plane, and let ϕ be any one-to-one*

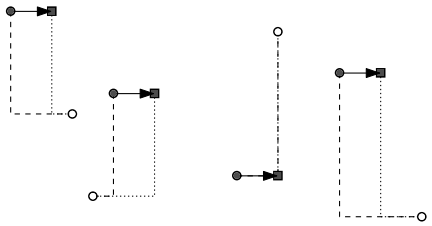


Figure 2: We can improve a matching between A (grey) and B (white) indicated by the dashed lines by x-aligning the point sets (the dotted lines).

matching between A and B. Then there is a horizontal translation $t^* \neq \overline{0}$ such that $A^* = A + t^*$ and B are x-aligned and $D_{\mathcal{T}}(\phi, t^*) \leq D_I(\phi)$.

Due to the lack of space, we omit the details of the proof. The crucial observation is that for a given ϕ, $X'(t) = X_{\mathcal{T}}(\phi, t)$ is a piecewise linear function in t that has its minimum at a breakpoint. Such a breakpoint corresponds to aligning the point sets. An analogous argument gives us that we can decrease $Y_{\mathcal{T}}$ by y-aligning two non y-aligned sets of points.

Consider the set T of translations that both x-align and y-align A and B. A translation $t \in T$ x-aligns a pair of points (a, b), and independently y-aligns a pair of points (a', b'). This means T contains at most n^4 translations.

From Lemma 1 (and its counterpart for y-aligning the point sets) it follows that T contains an optimal translation t^*. Hence, we can find ϕ^* by computing a minimum distance matching for all translations in T. If we use the algorithm of Vaidya [9] to compute these point set matchings we obtain the following result:

Theorem 2 *Given two sets A and B of n points in the plane, a one-to-one matching ϕ^* and a translation t^* that minimise $D_{\mathcal{T}}$ can be computed in $O(n^4 \cdot n^2 \log^3 n) = O(n^6 \log^3 n)$ time.*

The main difficulty in improving this result is that $X^*(t) = X(\phi_t^*, t)$, where ϕ_t^* denotes an optimal matching for horizontal translation t, is not unimodal. Therefore X^* may have several local minima, which means we cannot use a binary search to find an optimal translation t^*. Instead, we have to compute a matching for all translations in T.

Minimising L_1 under scaling. For scaling we can use the same procedure as for translation: we prove that there is an optimal scaling that x-aligns and y-aligns A and B and does not increase the total distance. We again have a set of at most n^4 scalings that is guaranteed to contain an optimal scaling. Hence:

Theorem 3 *Given two sets A and B of n points in the plane, a one-to-one matching ϕ^*, and a scaling λ^* that minimise D_Λ can be computed in $O(n^6 \log^3 n)$ time.*

Minimising L_1 under both translation and scaling. We can use same the approach, but now we x-align (y-align) two distinct pairs of points. We obtain:

Theorem 4 *Given two sets A and B of n points in the plane, a one-to-one matching ϕ^*, a translation t^*, and a scaling λ^* that minimise D can be computed in $O(n^{10} \log^3 n)$ time.*

3 Preserving directional relations

The second criterion that we consider is preserving directional relations. Let A and B be two sets of n points in which no two points have the same x- or y-coordinate, and let $dir(p, q)$ denote the directional relation of q with respect to p (see Fig. 3 (a)). The goal is now to find a matching $\phi^* : A \to B$ that maximises the number of pairs $(a_1, a_2) \in A \times A$ for which $dir(a_1, a_2) = dir(\phi^*(a_1), \phi^*(a_2))$. Stated differently, we are looking for a matching ϕ^* that minimises the number of *out-of-order* pairs W defined as

$$W(\phi) = |\{(a_1, a_2) \mid (a_1, a_2) \in A \times A \wedge dir(a_1, a_2) \neq dir(\phi(a_1), \phi(a_2))\}|.$$

To avoid many nested brackets we will write $a' = \phi(a)$ from now on. Furthermore, we observe that translations and scalings do not influence W.

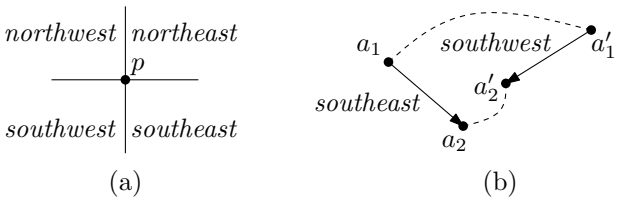


Figure 3: (a) The areas in the plane corresponding to each direction. (b) The directional relation between a_1 and a_2 is not preserved, (a_1, a_2) is an x-inversion.

A 4-approximation algorithm for minimising W. We now describe an algorithm to compute a matching that approximately minimises W. Let $x\text{-}rank_P(p)$ denote the x-*rank* of point $p \in P$: that is, the number of points in P to the left of p. For points $p \in P$ and $q \in Q$ we write $p \prec_x q$ for $x\text{-}rank_P(p) < x\text{-}rank_Q(q)$. The y-rank and $\prec_y$ are defined analogously.

For a given matching, $(a_1, a_2) \in A \times A$ is an x-*inversion* if (and only if) $a_1 \prec_x a_2$ and $a'_1 \succ_x a'_2$, or $a_2 \prec_x a_1$ and $a'_2 \succ_x a'_1$. See Fig. 3 (b). Similarly we define a y-*inversion*. An inversion is an x-inversion,

a y-inversion or both. We denote the number of x-inversions and the number of y-inversions of matching ϕ by $I_x(\phi)$ and $I_y(\phi)$, respectively. It is easy to see that there is a one-to-one correspondence between the number of out-of-order pairs $W(\phi)$ of matching ϕ and the number of inversions $I(\phi)$, i.e. $W(\phi) = I(\phi)$. Furthermore, we have $\max(I_x(\phi), I_y(\phi)) \leq I(\phi)$ and $I(\phi) \leq I_x(\phi) + I_y(\phi)$.

We define a distance measure w between points $a \in A$ and $b \in B$:

$$w(a,b) = |x\text{-}rank_A(a) - x\text{-}rank_B(b)| + |y\text{-}rank_A(a) - y\text{-}rank_B(b)|.$$

We now compute a minimum distance matching ϕ with w as distance measure. The distance measure w is simply the L_1-distance on the ranks of the points, which means we can use Vaidya's algorithm [9] to compute ϕ. We again denote the total distance of ϕ by $D(\phi)$, and decompose it into a separate x-component $X(\phi)$ and a y-component $Y(\phi)$.

The intuition behind our approach is as follows. Consider matching a point a_r with $x\text{-}rank_A(a_r) = i$ to a point b_r with $x\text{-}rank_B(b_r) = j > i$. By the pigeonhole principle there are at least $j - i$ points $\hat{a}$ with $x\text{-}rank_A(\hat{a}) > i$ that are matched to a point $\hat{b}$ with $x\text{-}rank_B(\hat{b}) < j$. Hence, matching a_r to b_r will result in at least $j - i = |x\text{-}rank_A(a_r) - x\text{-}rank_B(b_r)|$ x-inversions.

To prove this we order the points $a \in A$ by the rank of $\phi(a)$. For point a_r we consider the number of points that have to "overtake" a_r from left to right when sorting them by rank in A (denoted $\hat{a}$ above). If there are m such points, a_r moves a distance of $2m + (j - i)$ in the sorting process. It follows that $I_x(\phi) = M + X(\phi)/2$, where $M = \sum_{a \in A} m_a$.

Additionally, we can show that there are also at least m points that overtake a_r from right to left, which results in $M \leq X(\phi)/2$. Hence $I_x(\phi) \leq X(\phi)$.

An easy argument as used in the analysis of bubblesort gives us an upper bound of $X(\phi) \leq 2I_x(\phi)$ for the number of inversions in terms of X. We conclude:

Lemma 5 $I_x(\phi) \leq X(\phi) \leq 2I_x(\phi)$.

A symmetric argument holds for the y-rank and the number of y-inversions. This allows us to prove $W(\phi) = I(\phi) \leq D(\phi) \leq 4I(\phi) = 4W(\phi)$. The following theorem follows:

Theorem 6 *Given two sets A and B of n points in the plane, a one-to-one matching ϕ such that $W(\phi) \leq 4 \cdot \min_{\phi^* \in \Phi} W(\phi^*)$ can be computed in $O(n^2 \log^3 n)$ time.*

4 Concluding Remarks

We have seen how to compute a minimum distance matching with respect to the total L_1-distance under translation, scaling, and both translation and scaling. An implementation of our method shows that for small values of n we can compute a minimum distance matching under translation or scaling. However, optimising under both is unfeasible in practice.

Furthermore, we can use an algorithm for a minimum L_1-distance matching to obtain a matching that approximately maximises the number of correct directional relations. An interesting remaining question is whether it is possible to devise an exact algorithm for this problem that runs in polynomial time.

References

[1] H. Alt and L. Guibas. Discrete geometric shapes: Matching, interpolation, and approximation. In *Handbook of Computational Geometry*, chapter 3, pages 121–153. Elsevier, 1996.

[2] H. Alt, K. Mehlhorn, H. Wagener, and E. Welzl. Congruence, similarity, and symmetries of geometric objects. *Discrete and Computational Geometry*, 3(1): 237–256, 1988.

[3] H. Alt, L. Scharf, and D. Schymura. Probabilistic matching of planar regions. *Computational Geometry*, 43(2):99–114, 2010.

[4] M. Atkinson. An optimal algorithm for geometrical congruence. *Journal of Algorithms*, 8(2):159–172, 1987.

[5] S. Cohen and L. Guibas. The earth mover's distance under transformation sets. In *Proc. 7th IEEE International Conference on Computer Vision*, volume 2, pages 1076–1083, 1999.

[6] A. Efrat and A. Itai. Improvements on bottleneck matching and related problems using geometry. In *Proc. 12th annual ACM Symposium on Computational Geometry*, pages 301–310, 1996.

[7] J. Hong and X. Tan. A new approach to point pattern matching. In *Proc. 9th International Conference on Pattern Recognition*, volume 1, pages 82–84, 1988.

[8] J. Sprinzak and M. Werman. Affine point matching. *Pattern Recognition Letters*, 15(4):337–339, 1994.

[9] P. Vaidya. Geometry helps in matching. *SIAM Journal on Computing*, 18(6):1201–1225, 1989.

[10] R. Veltkamp and M. Hagedoorn. State of the Art in Shape Matching. In *Principles of Visual Information Retrieval*, chapter 4, pages 87–115. Springer-Verlag, 2001.

[11] J. Wood and J. Dykes. Spatially ordered treemaps. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1348–1355, 2008.

Coloring Dynamic Point Sets on a Line

Jean Cardinal* Nathann Cohen* Sébastien Collette* Michael Hoffmann† Stefan Langerman*
Günter Rote‡

Abstract

We consider a coloring problem on dynamic, one-dimensional point sets: points appearing and disappearing on a line at given times. We wish to color them with k colors so that at any time, any sequence of $p(k)$ consecutive points, for some function p, contains at least one point of each color.

We prove that no such function $p(k)$ exists in general. However, in the restricted case in which points appear gradually, but never disappear, we give a coloring algorithm guaranteeing the property at any time with $p(k) = 8k - 5$.

This can be interpreted as coloring point sets in $\mathbb{R}^2$ with k colors such that any bottomless rectangle containing at least $8k - 5$ points contains at least one point of each color. Chen *et al.* (2009) proved that such colorings do not always exist in the case of general axis-aligned rectangles. Our result also complements recent results from Keszegh and Pálvölgyi (2011).

1 Introduction

It is straightforward to notice that n points lying on a line can be colored with k colors in such a way that any set of k consecutive points receive different colors: it is sufficient to color them cyclically with the colors $1, 2, \ldots, k, 1, \ldots$. What can we do if points can appear and disappear on the line, and we wish a similar property to hold at any time? More precisely, we fix the number k of colors, and wish to maintain the property that at any given time, any sequence of $p(k)$ consecutive points, for some function p, contains at least one point of each color.

We show that in general, such a function does not exist: there are dynamic point sets on a line that are impossible to color with two colors so that monochromatic subsequences have bounded length. This holds even if the whole schedule of appearances and disappearances is known in advance. This family of point sets is described in Section 2.

*Université libre de Bruxelles (ULB), Belgium. jcardin@ulb.ac.be, secollet@ulb.ac.be, slanger@ulb.ac.be, nathann.cohen@gmail.com

†ETH Zürich, Switzerland. hoffmann@inf.ethz.ch

‡Freie Universität Berlin, Germany. rote@inf.fu-berlin.de

We prove, however, that there exists a linear function p in the case where points can appear on the line at any time, but *never disappear*. Furthermore, this is achieved in a constructive, *semi-online* fashion: the coloring decision for a point can be delayed, but at any time the currently colored points yield a suitable coloring of the set. The algorithm is described in Section 3.

In Section 4, we restate the result in terms of a coloring problem in $\mathbb{R}^2$: there exists a constant c such that for any integer $k \geq 1$, every point set in $\mathbb{R}^2$ can be colored with k colors so that any *bottomless* rectangle containing at least ck points contains one point of each color. Here, an axis-aligned rectangle is said to be bottomless whenever the y-coordinate of its bottom edge is $-\infty$.

Motivations and previous works. The problem is motivated by previous intriguing results in the field of geometric hypergraph coloring. Here, a geometric hypergraph is a set system defined by a set of points and a set of geometric ranges, typically polygons, disks, or pseudodisks. Every hyperedge of the hypergraph is the intersection of the point set with a range.

It was shown recently [12, 1, 4] that for every convex polygon P, there exists a constant c, such that any point set in $\mathbb{R}^2$ can be colored with k colors in such a way that any translation of P containing at least $p(k) = ck$ points contains at least one point of each color.

For the range spaces defined by translates of a given convex polygon, this corresponds to partitioning a given point set into k subsets, each subset being an ε-net for $\varepsilon = ck/n$. More on the relation between this coloring problem and ε-nets can be found in the recent papers of Varadarajan [13], and Pach and Tardos [9].

The problem for translates of polygons can be cast in its dual form as a covering decomposition problem: given a set of translates of a polygon P, we wish to color them with k colors so that any point covered by at least $p(k)$ of them is covered by at least one of each color. The two problems can be seen to be equivalent by replacing the points by translates of a symmetric image of P centered on these points. The covering decomposition problem has a long history that dates back to conjectures by János Pach in the early 80s (see for instance [7, 2], and references therein). The decomposability of coverings by unit disks was con-

This is an extended abstract of a presentation given at EuroCG 2012. 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 in a conference with formal proceedings and/or in a journal.

sidered in a seemingly lost unpublished manuscript by Mani and Pach in 1986. Up to recently, however, surprisingly little was known about this problem.

For other classes of ranges, such as axis-aligned rectangles, disks, or translates of some concave polygons [3, 8, 10, 11], such a coloring does not always exists, even when we restrict ourselves to two colors. For instance, the following result holds: for any integer $p \geq 2$, there exists a set of points in $\mathbb{R}^2$, every 2-coloring of which is such that there exists an open disk containing p monochromatic points.

Keszegh [5] showed in 2007 that every point set could be 2-colored so that any bottomless rectangle containing at least 4 points contains both colors. More recently, Keszegh and Pálvölgyi [6] proved the cover-decomposability of octants in $\mathbb{R}^3$: every collection of translates of the first octant can be 2-colored so that any point of $\mathbb{R}^3$ that is covered by at least 12 octants is covered by at least one of each color. This result generalizes the previous one (with a loser constant), as incidence systems of bottomless rectangles in the plane can be produced by restricted systems of octants in $\mathbb{R}^3$. It also implies similar covering decomposition results for homotopic copies of triangles. The generalization to k-colorings, however, still seems elusive.

Our positive result on bottomless rectangles (Corollary 4) is a generalization of Keszegh's results [5] to k-colorings. To our knowledge, this is the first example of a k-coloring achieving a linear bound $p(k)$ for ranges that are not translates of a given convex body. We see this as a first step towards similar results for more general range spaces, such as homothetic copies of convex polygons.

2 Coloring dynamic point sets

A *dynamic point set* S in $\mathbb{R}$ is a collection of triples $(v_i, a_i, d_i) \in \mathbb{R}^3$, with $d_i \geq a_i$, that is interpreted as follows: the point $v_i \in \mathbb{R}$ appears on the real line at time a_i and disappears at time d_i. Hence the set $S(t)$ of points that are present at time t are the points v_i with $t \in [a_i, d_i]$. A k-coloring of a dynamic point set assigns one of k colors to each such triple.

We now show that it is not possible to find a 2-coloring of such a point set while avoiding long monochromatic subsequences at any time.

Theorem 1 *For every $p \in \mathbb{N}$, there exists a dynamic point set S with the following property: for every 2-coloring of S, there exists a time t such that $S(t)$ contains p consecutive points of the same color.*

Proof. In order to prove this result, we work on an equivalent two-dimensional version of the problem. From a dynamic point set, we can build n horizontal segments in the plane, where the ith segment goes from (a_i, v_i) to (d_i, v_i). At any time t the visible points $S(t)$ correspond to the intervals that intersect the line $x = t$. It is therefore equivalent, in order to obtain our result, to build a collection of horizontal segments in the plane that cannot be 2-colored in such a way that any set of p segments intersecting some vertical segment contains one element of each color.

Our construction borrows a technique from Pach, Tardos, and Tóth [10]. In this paper, the authors provide an example of a set system whose ground set cannot be 2-colored without leaving some set monochromatic. This set system $\mathcal{S}$ is built on top of the $1 + p + \cdots + p^{p-1} = \frac{1-p^p}{1-p}$ vertices of a p-regular tree $\mathcal{T}^p$ of depth p, and contains two kinds of sets :

- the $1 + p + \cdots + p^{p-2}$ sets of *siblings*: the sets of p vertices having the same father,
- the p^{p-1} sets of p vertices corresponding to a path from the root vertex to one of the leaves in $\mathcal{T}^p$.

It is not difficult to realize that this set system is not 2-colorable: by contradiction, if every set of siblings is non-monochromatic, we can greedily construct a monochromatic path from the root to a leaf.

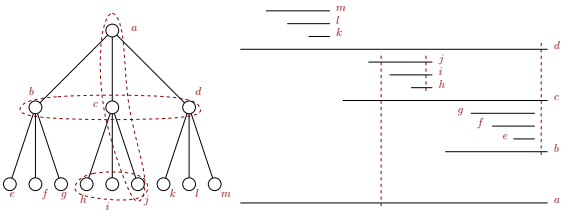
We now build a collection of horizontal segments corresponding to the vertices of $\mathcal{T}^p$, in such a way that for any set $E \in \mathcal{S}$ there exists a time t at which the elements of E are consecutive among those that intersect the line $x = t$. For any p (see Fig. 1), the construction starts with a building block B_p^1 of p horizontal segments, the ith segment going from $(-\frac{i}{p}, i)$ to $(0, i)$. Because these p segments represent *siblings* in $\mathcal{T}^p$, they are consecutive on the vertical line that goes through their rightmost endpoint, and hence cannot all receive the same color.

Block B_p^{j+1} is built from a copy of B_p^1 to which are added p resized and translated copies of B_p^j : the ith copy lies in the rectangle with top-right corner $(-\frac{i-1}{p}, i + 1)$ and bottom-left corner $(-\frac{i}{p}, i)$. With this construction, the ancestors of a segment are precisely those that are below it on the vertical line that goes through its leftmost point. When such sets of ancestors are of cardinality p, which only happens when one considers the set of ancestors of a leaf, the set is required to be non-monochromatic.

By adding to B_p^{p-1} a last horizontal segment below all others, corresponding to the root of $\mathcal{T}^p$, we ensure that a feasible 2-coloring of the segments would yield a proper 2-coloring of $\mathcal{S}$, which we know does not exist. □

The above result implies that no function $p(k)$ exists for any k that answers the original question. If it were the case, then we could simply merge color classes of a k-coloring into two groups and contradict the above statement.

Theorem 1 can also be interpreted as the indecomposability of coverings by a specific class of unbounded



(a) The tree $\mathcal{T}^3$.

(b) The corresponding set of horizontal segments B_3^2, with a root segment a.

Figure 1: The recursive construction of theorem 1, for $p = 3$.

polytopes in $\mathbb{R}^3$. We omit the details of this interpretation.

3 Coloring point sets under insertion

Since we cannot bound the function $p(k)$ in the general case, we now consider a simple restriction on our dynamic point sets: we let the deletion times d_i be infinite for every i. Hence points appear on the line, but never disappear.

A natural idea to tackle this problem is to consider an online coloring strategy, that would assign a color to each point in order of their arrival times a_i, without any knowledge of the points appearing later. However, we cannot guarantee any bound on $p(k)$ unless we delay some of the coloring decisions. To see this, consider the case $k = 2$, and call the two colors red and blue. An online algorithm must color each new point in red or blue as soon as it is presented. We can design an adversary such that the following invariant holds: at any time, the set of points is composed of a sequence of consecutive red points, followed by a sequence of consecutive blue points. The adversary simply chooses the new point to lie exactly between the two sequences at each step.

Our computation model will be *semi-online*: The algorithm considers the points in their order of the arrival time a_i. At any time, a point in the sequence either has one of the k colors, or is uncolored. Uncolored points can be colored later, but once a point is colored, it keeps its color for the rest of the procedure. At any time, the colors that are already assigned suffice to satisfy the property that any subsequence of $p(k)$ points have one point of each color, where $p(k) = O(k)$.

Theorem 2 *Every dynamic point set without disappearing points can be k-colored in the semi-online model such that at any time, every subsequence of at least $8k - 5$ consecutive points contains at least one point of each color.*

Proof. We define a *gap* for color i, or an i-gap, as a maximal interval containing no point of color i, that is, either between two successive occurrences of color i, or before the first occurrence, or after the last occurrence. At every step, we decide to assign a color to some of the uncolored points, in order to maintain the property that every gap contains at most $p(k) - 1$ points. This implies that any subsequence of $p(k)$ consecutive points contains at least one point of each color. We show that $p(k)$ can be linear in k. For this purpose, the algorithm will maintain another invariant:

(a) every gap contains at most $B = 4k - 2$ uncolored points.

At every step, we consider the new point, and leave it uncolored. This may cause some i-gap to contain $B+1$ uncolored points, and therefore violate invariant *(a)*. In that case, we pick one such gap and color its $(B/2 + 1)$th uncolored point with color i. This splits the i-gap into two parts, each containing $B/2$ uncolored points. If another gap was violating the invariant, say for color j, two cases can occur. Either the j-gap also contains the $(B/2 + 1)$th uncolored point of the i-gap, in which case the invariant is not violated anymore, or we can color with color j the $(B/2 + 1)$th uncolored point of the j-gap. In the latter case, note that the newly colored point cannot belong to the i-gap, as otherwise the first case would apply.

We now show that this algorithm also maintains the following invariant:

(b) every gap contains at least k uncolored points.

From the above steps, we know that when a point is colored with color i, its is separated from the two occurrences of color i on its left and right by at least $B/2$ uncolored points. Later, however, some of these uncolored points can be colored with a color distinct from i. Hence the number of uncolored points can decrease below $B/2$. However, from the time it is less than $B/2$, there cannot be more than one point of each color distinct from i appearing, since any new pair of colored points is separated by at least $B/2$ uncolored points. Hence there must still be at least $B/2 - (k - 1) = k$ uncolored points between two points of color i.

Now it remains to check that the two invariants *(a)* and *(b)* imply that any two successive points of the same color i cannot be separated by more than $O(k)$ other points. First, from invariant *(a)*, there can be at most B uncolored point in this i-gap. Then, every pair of occurrences of a color distinct from i is separated

by at least k uncolored points. This implies that there cannot be more than $1 + \left\lfloor \frac{B}{k} \right\rfloor$ occurrences of each of the $k-1$ other colors in the i-gap. Hence the total number of points cannot exceed

$$\begin{aligned} p(k) - 1 &= B + (k-1)\left(1 + \left\lfloor \frac{B}{k} \right\rfloor\right) \\ &= 4k - 2 + (k-1)\left(1 + \left\lfloor \frac{4k-2}{k} \right\rfloor\right) \\ &= 8k - 6. \end{aligned}$$

□

The function is not tight for every k, as witnessed by Keszegh's result for $k = 2$ [5]. For $k = 3$, we have the following result, the proof of which is omitted.

Theorem 3 *Every dynamic point set without disappearing points can be 3-colored in the semi-online model such that at any time, every subsequence of at least 8 consecutive points contains at least one point of each color.*

4 Coloring points with respect to bottomless rectangles

A *bottomless rectangle* is a set of the form $\{(x, y) \in \mathbb{R}^2 : a \leq x \leq b, y \leq c\}$, for a triple of real numbers (a, b, c) with $a \leq b$. We consider the following geometric coloring problem: given a set of points in the plane, we wish to color them with k colors so that any bottomless rectangle containing at least $p(k)$ points contains at least one point of each color. It is not difficult to realize that the problem is equivalent to that of the previous section.

Corollary 4 *Every point set $S \subset \mathbb{R}^2$ can be colored with k colors so that any bottomless rectangle containing at least $8k - 5$ points of S contains at least one point of each color.*

Proof. The algorithm proceeds by sweeping S vertically in increasing y-coordinate order. This defines a dynamic point set S' that contains at time t the x-coordinates of the points below the horizontal line of equation $y = t$. The set of points of S that are contained in a bottomless rectangle $\{(x, y) \in \mathbb{R}^2 : a \leq x \leq b, y \leq t\}$ correspond to the points in the interval $[a, b]$ in $S'(t)$. Hence the two coloring problems are equivalent, and Theorem 2 applies. □

The following is a corollary of Theorem 3.

Corollary 5 *Every point set $S \subset \mathbb{R}^2$ can be colored with 3 colors so that any bottomless rectangle containing at least 8 points of S contains at least one point of each color.*

Acknowledgments

This research is supported by the the ESF EUROCORES programme EuroGIGA, CRP ComPoSe (`http://www.eurogiga-compose.eu`). It was initiated at the ComPoSe kickoff meeting held at CIEM (International Centre for Mathematical meetings) in Castro de Urdiales (Spain) on May 23–27, 2011. The authors warmly thank the organizers of this meeting, Oswin Aichholzer, Ferran Hurtado, and Paco Santos, as well as all the other participants, for providing such a great working environment.

References

[1] Greg Aloupis, Jean Cardinal, Sébastien Collette, Stefan Langerman, David Orden, and Pedro Ramos. Decomposition of multiple coverings into more parts. *Discrete & Computational Geometry*, 44(3):706–723, 2010.

[2] Peter Brass, William O. J. Moser, and János Pach. *Research Problems in Discrete Geometry*. Springer, 2005.

[3] Xiaomin Chen, János Pach, Mario Szegedy, and Gábor Tardos. Delaunay graphs of point sets in the plane with respect to axis-parallel rectangles. *Random Struct. Algorithms*, 34(1):11–23, 2009.

[4] Matt Gibson and Kasturi R. Varadarajan. Optimally decomposing coverings with translates of a convex polygon. *Discrete & Computational Geometry*, 46(2):313–333, 2011.

[5] Balázs Keszegh. Weak conflict-free colorings of point sets and simple regions. In *CCCG*, pages 97–100, 2007.

[6] Balázs Keszegh and Dömötör Pálvölgyi. Octants are cover decomposable. *CoRR*, abs/1101.3773, 2011.

[7] János Pach. Covering the plane with convex polygons. *Discrete & Computational Geometry*, 1:73–81, 1986.

[8] János Pach and Gábor Tardos. Coloring axis-parallel rectangles. *J. Comb. Theory, Ser. A*, 117(6):776–782, 2010.

[9] János Pach and Gábor Tardos. Tight lower bounds for the size of epsilon-nets. In *Proceedings of the 27th annual ACM symposium on Computational Geometry*, SoCG '11, pages 458–463, 2011.

[10] János Pach, Gábor Tardos, and Géza Tóth. Indecomposable coverings. In *CJCDGCGT*, pages 135–148, 2005.

[11] Dömötör Pálvölgyi. Indecomposable coverings with concave polygons. *Discrete & Computational Geometry*, 44(3):577–588, 2010.

[12] Dömötör Pálvölgyi and Géza Tóth. Convex polygons are cover-decomposable. *Discrete & Computational Geometry*, 43(3):483–496, 2010.

[13] Kasturi R. Varadarajan. Weighted geometric set cover via quasi-uniform sampling. In *Proceedings of the 42nd ACM symposium on Theory of computing*, STOC '10, pages 641–648, 2010.

Order type invariant labeling and comparison of point sets

Greg Aloupis* Muriel Dulieu† John Iacono† Stefan Langerman* Özgür Özkan†
Suneeta Ramaswami‡ Stefanie Wuhrer§

Abstract

We consider the problem of computing an order type invariant labeling for a given set of n points. In $2D$, such a labeling can be constructed in $O(hn^2)$ time, where h is the size of the smallest convex layer. In $3D$ the time complexity is $O(n^3 \log n)$ if the point set is in general position[1].

This is useful to test if two point sets have the same order type within the same time bounds. It can also be used as preprocessing for any order type invariant algorithm, such as triangulation/tetrahedralization, or polygonization.

1 Introduction

When an algorithm computes a structure such as a triangulation or polygonization, and many solutions are possible, sometimes it is desirable that the same solution is always returned for two combinatorially equivalent point sets. With this in mind, we consider the problem of computing an order type invariant labeling for a point set S.

In the plane, the *order type* of S is determined by assigning to every ordered triple of points p_i, p_j, p_k an orientation depending on their relative positions: clockwise, counter-clockwise, or collinear. In 3D, order type labels are assigned to ordered subsets of four points, depending on their arrangement (collinear, coplanar, left/right-handed tetrahedra). An order type representation (OTR) of S is any (possibly implicit) encoding of this information.

Two unlabeled point sets S_1 and S_2 are *combinatorially equivalent* if and only if they have the same order type, or in other words if they have the same OTR for some labeling. If the point sets are labeled, then there must be a bijection between S_1 and S_2, such that any triple in S_1 has the same orientation as its corresponding triple in S_2. In other words, the OTR of S_1 will be identical to the OTR for some permutation (or relabeling) of S_2. Note that the bijection is not necessarily unique.

Our goal is to provide an order type invariant (re)-labeling of a given point set S (this labeling of S, $p'_1, p'_2, \ldots, p'_n$ is just a permutation of the given points). This means that for any combinatorially equivalent set S' labeled by the algorithm as $q'_1, q'_2, \ldots, q'_n$, we will have the property that $\text{OTR}(p'_1, p'_2, \ldots, p'_n) = \text{OTR}(q'_1, q'_2, \ldots, q'_n)$. The implication is that any algorithm relying only on combinatorial structure will produce the same output regardless of the initial labeling of S, if our relabeling is used as preprocessing.

We show that in $2D$ an order type invariant labeling can be computed in $O(hn^2)$ time, where h is the size of the smallest convex layer of the given point set. The factor h actually represents the size of the smallest subset of labelings (out of all $n!$ labelings of S) that can be formed "quickly" so that any labeling in the subset has a distinguishable OTR compared to any labeling left out. It seems that producing such a subset is not harder than producing a subset of "representative points" of S. So, equivalently one could think of our first phase as finding h "special" points among S. We first focus on how to report h labelings, in Section 3. These are further processed to report a unique labeling, by comparing all OTRs, in Section 3. The factor of $O(n^2)$ in the above term represents the smallest known encoding of an OTR in $2D$.

In $3D$, for general position, we can quickly reduce S to a subset consisting of a constant number of points (equivalently, we obtain a constant number of labelings). Our order type invariant labeling is done in $O(n^3 \log n)$ time; see Section 4.

Our labeling permits the comparison of two point sets, to see if they have the same order type. See Section 5.

2 Related work

Goodman and Pollack [7] showed that the number of order types on n points is at least $n^{4n+O(n/\log n)}$ and at most n^{6n}. They improved the upper bound later to $\left(\frac{n}{2}\right)^{4n(1+O(1/\log(n/2)))}$ [8]. Aichholzer et al. [1] enumerated all order types for up to 10 points, and thus established that any two such point sets in general position with the same number of hull points have isomorphic triangulations. Aichholzer and Krasser [2] extended this to sets of 11 points. Goodman et al. [10] showed that the coordinate representation of order type requires exponential storage.

*Département d'Informatique, Université Libre de Bruxelles, aloupis.greg@gmail.com, stefan.langerman@ulb.ac.be
†Department of Computer Science and Engineering, Polytechnic Institute of New York University, mdulieu@gmail.com, jiacono@poly.edu, ozgurozkan@gmail.com
‡Department of Computer Science, Rutgers University, rsuneeta@camden.rutgers.edu
§Cluster of Excellence MMCI, Saarland University, swuhrer@mmci.uni-saarland.de

[1]Our original accepted submission contained claims about higher dimensions and non-general position. After realizing that there was a flaw in our proof, we have since needed to revoke that portion of our work, as we work on a suitable correction.

The most related work is that of Goodman and Pollack [9], in which an efficient OTR was defined for any dimension d. Their OTR in 2D stores the number of points to the left of the vector through every ordered pair of points. In 3D, for each ordered triple of points, it stores the number of points in the halfspace bounded by the halfplane through the triple, with normal vector equal to the cross product of the triple. This extends easily to any dimension d, where the OTR has size $O(n^d)$. Note that there is no assumption on general position. Interestingly, they showed that this is sufficient to determine exactly which points are in any particular halfspace. The time to compute their OTR was given as $O(n^d \log n)$. Note that it has been established elsewhere that in 2D the time complexity is quadratic [6, 4].

Goodman and Pollack also defined *canonical orderings*, which are essentially a subset of all $n!$ orderings (labelings) that can be checked to determine if two sets are combinatorially equivalent. For 2D they gave h canonical orderings, one for each convex hull point p. For each p, the remaining points can be labeled by a clockwise sort around p, starting from any direction that does not intersect the convex hull. It is easy to see that if two point sets S_1,S_2 have the same order type, then the OTR of S_1 will match the OTR for at least one of the canonical orderings of S_2. This was a great improvement over the $n!$ OTR comparisons that would have to be made using brute force. Since two OTRs can be compared in quadratic time, two point sets can be compared in $O(hn^2)$ time. Note that one does not need the improvement of [6, 4] to avoid spending $\Theta(n^2 \log n)$ time computing each of the h OTRs. Since each of these OTRs represents the same point set, once one is computed the others can be obtained by a permutation, i.e. in quadratic time each. This is why quadratic time is spent for each of the h comparisons.

In 3D the canonical orderings were based on the directed edges of the convex hull (at most $6n - 12$ in total, which can be reported in $O(n \log n)$ time). For each such edge, a labeling of S could be obtained via a clockwise plane-sweep starting from a tangent plane containing the edge. The OTR corresponding to one of these labelings can be computed in $O(n^3 \log n)$ time, and each of the remaining takes cubic time, by a permutation on the first. So, two point sets can be compared, by in turn comparing an OTR of one to $\leq 6n-12$ OTRs of the other. Each such OTR comparison takes cubic time. Therefore two sets can be compared in $O(n^4)$ time.

Such sweeps can be extended to any fixed dimension d, according to [9]. There are $O(n^{\lfloor d/2 \rfloor})$ canonical orderings. Each corresponds to a "face flag", which can be thought of as a directed simplex. Each ordering can be computed in $O(n \log n)$ time once its face flag is known. Finding all face flags takes $O(n^d)$ time for general position; otherwise in $O(n^{(d(d+3)/2)})$ time.

For each ordering, an OTR is produced: the first in $O(n^d \log n)$ time, and the rest in $O(n^d)$ time. Therefore, for general position, it takes $O(n^{\lfloor 3d/2 \rfloor})$ time to compare two point sets (comparing $O(n^{\lfloor d/2 \rfloor})$ orderings, all of which are constructed in $O(n^{\lfloor 3d/2 \rfloor})$ time, and where each comparison takes $O(n^d)$ time). For non-general position the comparison is dominated by the time to compute all face flags: $O(n^{(d(d+3)/2)})$.

3 Order type invariant labeling in $2D$

We use "canonical" labelings, similarly to Goodman and Pollack [9]. Instead of using the convex hull and letting h represent its size, we set h to be the number of points on the convex layer c^* with smallest size. It takes $O(n \log n)$ time to compute all convex layers [3].

Assume for now that c^* contains more than one point. We select the starting vector for any point p on c^* to be the one through its clockwise neighbor on c^*. This is an order type invariant choice. Still, we can end up with $h = O(n)$ canonical labelings. For point sets with more than a constant number of convex layers, $h = o(n)$. For random point sets, h is sub-linear as well; see [5]. Note that the worst case is not the point set in convex position, for this is trivial to label (there are n possible distinct sequential labelings of a cycle, but they are all symmetric and combinatorially equivalent). It is worse to have a constant number of layers, each with a linear number of points.

Finally, if p is the only point on the smallest layer, then it is defined as the first point of our order type invariant labeling of S. We remove it from S and repeat. We will repeat at most once because even if we end up with a single point for a second time, the two points permit us to uniquely sort the rest.

The convex layer idea is simply a heuristic way of narrowing down the candidate labelings, from all permutations. The general idea is to apply any combinatorial test that easily ranks points, and keep only those points that have highest rank. If several points achieve the highest rank, we can apply another combinatorial test, and so on. So far, we have followed this idea by saying that the points of highest rank are those on a particular convex hull layer. Of course, we cannot apply an endless series of tests, for this will eventually become computationally expensive. The challenge is to select the most efficient tests possible. We have tried several combinations of heuristics (see full paper). Curiously, none so far have resulted in a worst-case sub-linear set of labelings, within $O(n^3)$ time.

By now we have chosen $O(h) = O(n)$ order type invariant labelings. We compute the OTR for each labeling $p'_1, \ldots, p'_n$, in quadratic time. Recall that this is a concatenation of $n(n-1)$ variables, each of which represents the number of points to the left of the vector from one point to another. The order of the variables is given by a lexicographic description of the vector index. This implicitly encodes any combinatorial differences missed by the simple geometric tests of the preceding section.

The rest is simple: we choose the largest lexicographic OTR as our solution. Since each OTR is a list of quadratic size, this takes $O(hn^2)$ time. If several lists tie for the

highest rank, we choose arbitrarily among them. Clearly any of these lists will provide the same combinatorial output if given to a combinatorial algorithm (for instance, for triangulation). Furthermore, if we are given two point sets, comparing their respective lexicographically selected OTRs will determine if they have the same order type. In 2D this is just a variant of the method of Goodman and Pollack, described in Section 2. Instead of comparing h OTR's from one set to one arbitrary OTR of the other, we compare h OTRs among themselves for each set, and then compare the winners.

4 Labeling in $\mathbb{R}^3$

Recall that in $3D$ the order type of S depends on the spatial relation among every quadruple of points.

In $2D$ we used only those points on the smallest convex layer, to determine a subset of labelings. The nice thing about $2D$ is that it is easy to obtain one labeling per point. In $3D$ it still makes sense (although only heuristically) to use only those points on the smallest convex layer, so we begin by discarding all other points. The objective is to obtain a labeled (ordered) triple of points, in an order type invariant way. From these, it is possible to label all remaining points with a sweep.

From the smallest convex layer, we will iteratively keep removing points, and in fact we will be satisfied if we can reduce S to any constant number of candidate points. However we want at least three non-collinear points, or possibly two if they happen to be convex hull neighbors. As in $2D$, if we remove too many points, we can store the few remaining and re-iterate on the removed set. Even if we end up with a collinear triple after three iterations of discarding, it is easy to ensure that the next iteration will give a non-collinear point; simply exclude all points collinear to the triple from that iteration. So this part of the algorithm does not rely on general position.

Once we have our desired constant number of points, for every ordered non-collinear triple among them, we can carry out a uniquely directed plane sweep to relabel S. By [9], the OTR for each such labeling can be computed in $O(n^3 \log n)$ time. Again, the overall solution will be the lexicographically largest OTR. For a constant number of candidate points, the number of triples, and thus the number of OTRs to compare, is constant. Each representation has size $O(n^3)$, so it takes cubic time to choose lexicographically. Therefore the overall run-time is also $O(n^3 \log n)$. What remains is to show how to remove all but a constant number of points, in an order type invariant way.

Let L be the points of the smallest convex layer. Let $CH(L)$ denote the convex hull of L. The following simple procedure does just what we need; it identifies a constant number of points from S. Let $P = L$. Repeat the following steps until no points are removed from P in step 2.

1. Compute $CH(P)$. Remove all interior points from P, including non-extreme collinear points.
2. Remove from P all points except those that have the lowest vertex degree on $CH(P)$.

If more than three points survive, $CH(P)$ must be a polyhedron with all vertex degrees being equal. If S is in general position then every face of $CH(P)$ must be a triangle. It can be shown, using Euler's formula, that $CH(P)$ must be combinatorially equivalent to a tetrahedron, octahedron, or icosahedron. Each has only a constant number of vertices.

Step 1 takes $O(n \log n)$ time. Step 2 takes linear time since the sum of vertex degrees is linear. There are fewer than n iterations since at least one point is removed each time. Thus this procedure takes $O(n^2 \log n)$ time, which does not affect the time complexity dominated by OTR computations.

5 Comparing the order type of two sets

As mentioned, the method of Goodman and Pollack [9] compares order types of three-dimensional point sets in general position in $O(n^4)$ time. The evaluation is done by comparing the OTR for each canonical labeling of S_1 to one arbitrary OTR from S_2.

We have ongoing research to improve OTR representation and computation. For now, in Section 4 we have shown that the number of canonical labelings can be reduced to a constant. It costs $O(n^3 \log n)$ time to compute the OTR for each, and with a lexicographic sort we can select one representative OTR for any point set. Then S_1 and S_2 are compared by matching their OTRs.

6 Final notes

Given an order-type invariant labeling of S, we can solve the following problems in an order-type invariant way. Consider any problem involving building a graph by adding edges to S, based on combinatorial comparisons, i.e. based on order type. Examples of such problems include triangulation/tetrahedralization, polygonization, finding a non-crossing matching, finding a non-crossing spanning tree, or computing a halving line or a ham-sandwich cut. An order type invariant labeling can be constructed as preprocessing, so that any combinatorially equivalent input will produce the same output.

We ask whether an OTR can be computed in $o(n^3 \log n)$ time in $3D$, and whether any worst-case sub-cubic algorithm exists for the planar case. For the latter, we specifically ask whether a sub-linear number of canonical labelings can be computed in sub-cubic time. Of course, the other possibility also exists (if the number of labelings remains linear): that of reporting an OTR in sub-quadratic

time. This is unlikely, as it implies the existence of an OTR that can be stored in sub-quadratic space. Whether this can be done was an open problem posed in [8].

Note that without general position, in $3D$ it is possible to follow the vertex-degree pruning step with another test for each vertex. That is, we can enumerate the size of surrounding faces, and keep only those vertices that qualify lexicographically. This would result in a polyhedron where every vertex appears identical, with respect to degree and surrounding face sizes. A topological proof of Archimedes' theorem (see [11, 12]) tells us that such a polyhedron is isomorphic to a semi-regular polyhedron (i.e., one of the Platonic or Archimedean solids, prisms, or antiprisms), or to the pseudorhombicuboctahedron. Among these, only the prisms and antiprisms may have more than constant size, and this is why general position is assumed. Our ongoing work involves resolving the case where pruning results in a prism of more than constant size (and where repetition of our algorithm on the discarded set of points recursively yields such prisms as well).

We thank the other participants of the 2011 Mid-Winter Workshop on Computational Geometry for providing a stimulating research environment.

References

[1] O. Aichholzer, F. Aurenhammer, and H. Krasser. Enumerating order types for small point sets with applications. *Order*, 19(3):265–281, 2002.

[2] O. Aichholzer and H. Krasser. Abstract order type extension and new results on the rectilinear crossing number. In *Symposium on Computational Geometry*, pages 91–98, 2005.

[3] B. Chazelle. On the convex layers of a planar point set. *IEEE Transactions on Information Theory*, 31(4):509–517, 1985.

[4] B. Chazelle, L. J. Guibas, and D. T. Lee. The power of geometric duality. In *IEEE Symposium on Foundations of Computer Science*, pages 217–225. IEEE, 1983.

[5] K. Dalal. Counting the onion. Master's thesis, McGill University, 2004.

[6] H. Edelsbrunner, J. O'Rourke, and R. Seidel. Constructing arrangements of lines and hyperplanes with applications. *SIAM Journal of Computing*, 15(2):341–363, 1986.

[7] J. Goodman and R. Pollack. Upper bounds for configurations and polytopes in R^d. *Discrete & Computational Geometry*, 1:219–227, 1986.

[8] J. Goodman and R. Pollack. The complexity of point configurations. *Discrete Applied Mathematics*, 31:167–180, 1991.

[9] J. E. Goodman and R. Pollack. Multidimensional sorting. *SIAM Journal on Computing*, 12(3):484–507, 1983.

[10] J. E. Goodman, R. Pollack, and B. Sturmfels. Coordinate representation of order types requires exponential storage. In *ACM Symposium on Theory of Computing*, 1989.

[11] M. Villarino. On the Archimedean or semiregular polyhedra. Technical Report arXiv:math/0505488v1, 2005.

[12] T. Walsh. Characterizing the vertex neighbourhoods of semi-regular polyhedra. *Geometriae Dedicata*, 1:117–123, 1972.

Proximity graphs: E, δ, Δ, χ and ω

Prosenjit Bose* Vida Dujmović* Ferran Hurtado† John Iacono‡ Stefan Langerman§
Henk Meijer¶ Vera Sacristán† Maria Saumell‖ David R. Wood**

Abstract

Graph-theoretic properties of certain proximity graphs defined on planar point sets are investigated. We first consider some of the most common proximity graphs of the family of the Delaunay graph, and study their number of edges, minimum and maximum degree, clique number, and chromatic number. Then we focus on the higher order versions of some of these graphs and give bounds on the same properties.

1 Introduction

Loosely speaking, a proximity graph has as its vertex set a set of points in the plane, and adjacency in the graph attempts to describe some of the proximity relations of the point set. Proximity graphs extract the relevant structure or shape of point sets, and thus find applications in areas where this structure is important, which include pattern recognition, computer vision, and cluster analysis.

This paper considers a family of proximity graphs comprising several graphs —and some of their variations— that are subgraphs of the Delaunay triangulation. We study classical graph-theoretic properties of these graphs: number of edges, minimum and maximum degree, chromatic number, and clique number. These parameters have been considered before in the literature; the existing results, though, leave some gaps and have been developed under distinct non-degeneracy assumptions for the point sets. In this paper we try to address these issues by fixing the same assumptions on the point sets for all graphs, and developing new bounds in order to close or narrow the existing gaps.

We first consider four order-0 proximity graphs, namely, the union of the minimum spanning trees, the relatively closest graph, the relative neighborhood graph, and the modified Gabriel graph (in the full version of this paper [3] we additionally look at the minimum spanning tree, the Gabriel graph, and the Delaunay graph). These graphs were quite popular in the eighties and early nineties, and many of their properties can be found in [5, 7, 8, 10, 11]. In this paper we focus on the basic properties mentioned earlier. When we study these graphs, we do not make any non-degeneracy assumption on the set of points on which the graphs are defined, since we believe that the analysis is more interesting in this case.

The rest of the paper is concerned with three higher order proximity graphs: the k-relative neighborhood graph, the k-Gabriel graph, and the k-Delaunay graph (in [3] we also consider the shared k-nearest neighbor graph and the k-nearest neighbor graph). Order-k graphs have been studied in [1, 2, 4, 9], but in general fewer of their properties are known. Here we assume that point sets S are in *general position* in an extended sense: no three points are collinear, no four points are concyclic and, for each $p \in S$, the kth nearest neighbor of p in S is unique, for any $k \geq 1$.

Definitions and results Let G be a graph. We denote by $V(G)$ (respectively, $E(G)$) the set of vertices (respectively, edges) of G. We denote by $d_G(v)$ the degree of v in G, where $v \in V(G)$. The *minimum degree* of G is $\delta(G) = \min\{d_G(v) : v \in V(G)\}$. The *maximum degree* of G is $\Delta(G) = \max\{d_G(v) : v \in V(G)\}$. A *clique* of G is a set of pairwise adjacent vertices. The *clique number* of G, denoted by $\omega(G)$, is the maximum number of vertices in a clique of G. A *k-coloring* of G is a mapping $f : V(G) \to \{1, 2, \ldots, k\}$ such that $f(v) \neq f(w)$ for every edge vw of G. The *chromatic number* of G, denoted by $\chi(G)$, is the minimum k such that G is k-colorable.

We denote by S a generic set of n points in the plane. All graphs considered in this paper are geometric graphs on S, that is, their edges consist of straight-line segments with endpoints in S. Points in S are usually denoted by $p_1, \ldots, p_i, \ldots, p_n$.

Definition 1 *A spanning tree of S with minimum sum of edge lengths is a* minimum spanning tree *of S.*

*School of Computer Science, Carleton University, jit@scs.carleton.ca, vida@cs.mcgill.ca
†Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, {ferran.hurtado,vera.sacristan}@upc.edu
‡Department of Computer Science and Engineering, Polytechnic Institute of New York University, jiacono@poly.edu
§Maître de recherches du F.R.S.-FNRS. Département d'Informatique, Université Libre de Bruxelles, stefan.langerman@ulb.ac.be
¶Science Department, Roosevelt Academy, h.meijer@roac.nl
‖Department of Applied Mathematics, Charles University, maria.saumell.m@gmail.com
**Department of Mathematics & Statistics , The University of Melbourne, woodd@unimelb.edu.au

This is an extended abstract of a presentation given at EuroCG 2012. 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 in a conference with formal proceedings and/or in a journal.

The set of minimum spanning trees of S is denoted by $\mathsf{MST}(S)$*, and the graph with vertex set S consisting of the union of all $T \in \mathsf{MST}(S)$ is denoted by* $\mathsf{U\text{-}MST}(S)$.

We denote by $\mathsf{O\text{-}LENS}(p_i, p_j)$ the region containing all points x such that $|p_i x| < |p_i p_j|$ and $|p_j x| < |p_i p_j|$, and by $\mathsf{C\text{-}LENS}(p_i, p_j)$ the regions containing the points x such that $|p_i x| \leq |p_i p_j|$ and $|p_j x| \leq |p_i p_j|$.

Definition 2 *The* relative neighborhood graph*, denoted by* $\mathsf{RNG}(S)$*, is the graph in which p_i, p_j are adjacent if* $\mathsf{O\text{-}LENS}(p_i, p_j) \cap S = \emptyset$*. The* relatively closest graph*, denoted by* $\mathsf{RCG}(S)$*, is the graph in which p_i, p_j are adjacent if* $\mathsf{C\text{-}LENS}(p_i, p_j) \cap S = \{p_i, p_j\}$.

We denote by $\mathsf{O\text{-}DISC}(p_i, p_j)$ and $\mathsf{C\text{-}DISC}(p_i, p_j)$ respectively the open and closed discs centered at the midpoint of $\overline{p_i p_j}$ with p_i and p_j on their boundary.

Definition 3 *The* modified Gabriel graph*, denoted by* $\mathsf{MGG}(S)$*, is the graph in which p_i, p_j are adjacent if* $\mathsf{O\text{-}DISC}(p_i, p_j) \cap S = \emptyset$.

The graphs defined so far satisfy that, for every point set S, $\mathsf{U\text{-}MST}(S) \subseteq \mathsf{RNG}(S) \subseteq \mathsf{MGG}(S)$ and $\mathsf{RCG}(S) \subseteq \mathsf{RNG}(S)$ (see [5, 10, 11]).

We next define some higher order proximity graphs.

Definition 4 *The* k-relative neighborhood graph*, denoted by* k-$\mathsf{RNG}(S)$*, is the graph in which p_i, p_j are adjacent if* $|\mathsf{O\text{-}LENS}(p_i, p_j) \cap S| \leq k$.

Definition 5 *The* k-Gabriel graph*, denoted by* k-$\mathsf{GG}(S)$*, is the graph in which p_i, p_j are adjacent if* $|\mathsf{C\text{-}DISC}(p_i, p_j) \cap S| \leq k + 2$.

Definition 6 *The* k-Delaunay graph*, denoted by* k-$\mathsf{DG}(S)$*, is the graph in which p_i, p_j are adjacent if there exists a circle through p_i and p_j that contains at most k points from S in its interior.*

It is well known that, for every point set S, k-$\mathsf{RNG}(S) \subseteq k$-$\mathsf{GG}(S) \subseteq k$-$\mathsf{DG}(S)$.

The results that were known before our work and the ones we prove are summarized in Tables 1 and 2. Subscripts containing a reference mean that the corresponding bound is proved in that reference. The asterisks indicate results that are well-known or trivial. Due to lack of space, in this extended abstract we only prove the bounds for the modified Gabriel graph and the k-Delaunay graph. The remaining proofs are given in [3].

Table 1: Bounds on properties of proximity graphs. No non-degeneracy assumptions are made.

	RCG	U-MST / RNG	MGG
$\min \lvert E\rvert$	$0_{[5]}$	$n-1$	$n-1$
$\max \lvert E\rvert \geq$	$2n-6$	$3n-8$	$4n-6\sqrt{n}+2$
$\max \lvert E\rvert \leq$	$2n-5_{[5]}$	$3n-8$	$4n-\frac{2\sqrt{n}}{3}$
$\max \delta$	$3_{[5]}$	5	$\in \{6,7\}$
$\max \Delta$	5	$n-1$	$n-1$
$\max \chi$	$3_{[5]}$	4	$\in \{4,\ldots,8\}$
$\max \omega$	2	$3_{[11]}$	4

Table 2: Bounds on properties of proximity graphs. Several non-degeneracy assumptions are made.

	k-RNG	k-GG	k-DG
$\min \lvert E\rvert \geq$	$\frac{(k+1)n}{2}$	$\frac{(k+1)n}{2}$	$(k+1)n_{[1]}$
$\min \lvert E\rvert \leq$	$kn + o(kn)$	$kn + o(kn)$	$\frac{3kn}{2} + o(kn)$
$\max \lvert E\rvert \geq$	$\frac{3\pi}{4\pi-3\sqrt{3}}nk$	$2kn + o(nk)$	$3kn + o(nk)$
$\max \lvert E\rvert \leq$	$\frac{5(k+1)n}{2}$	$3kn + o(nk)_{[1]}$	$3kn + o(nk)_{[1]}$
$\max \delta \geq$	$2k+2$	$3k+2$	$4k+3$
$\max \delta \leq$	$3k+3$	$6k+5$	$6k+5$
$\max \Delta$	$5k+5$	$n-1_*$	$n-1_*$
$\max \chi \geq$	$k+2$	$3k+3_{[2]}$	$4k+4_{[2]}$
$\max \chi \leq$	$3k+4$	$6k+6_{[2]}$	$6k+6_{[2]}$
$\max \omega \geq$	$k+2$	$3k+3$	$4k+4$
$\max \omega \leq$	$k+2$	$3k+3$	$4.74k+14$

2 The modified Gabriel graph

Since $\mathsf{U\text{-}MST}(S) \subseteq \mathsf{MGG}(S)$, we have that the number of edges of $\mathsf{MGG}(S)$ is at least $n-1$. This is attained when S is a set of points lying on a line l, since in this case $\mathsf{MGG}(S)$ only contains the edges connecting consecutive points in l.

Our lower bound on the maximum edges of $\mathsf{MGG}(S)$ is $4n-6\sqrt{n}+2$, and corresponds to the case where S is a square grid of size $\sqrt{n} \times \sqrt{n}$. Our upper bound uses a technical lemma, whose proof is omitted.

Lemma 1 *Let $H = (S, E)$ be a plane geometric graph formed by 4-cycles $p_i p_j p_l p_m p_i \in F_4$ such that $p_i, p_j, p_l, p_m \in S$ are the vertices of a closed rectangle which is empty of points from S, except for p_i, p_j, p_l, p_m. Let $|F_4|$ be the number of such 4-cycles. The number of edges of H that belong to exactly one of these 4-cycles is at least $4\sqrt{|F_4|}$.*

Theorem 2 *Every modified Gabriel graph on n vertices has at most $4n - \frac{2\sqrt{n}}{3}$ edges.*

Proof. Let $G = \mathsf{MGG}(S)$. Suppose that $p_i p_j$ and $p_l p_m$ are two crossing edges of G. Since $\mathsf{O\text{-}DISC}(p_i, p_j) \cap S = \emptyset$, we have that $\widehat{p_i p_l p_j} \leq \pi/2$ and $\widehat{p_i p_m p_j} \leq \pi/2$. Analogously, $\widehat{p_l p_i p_m} \leq \pi/2$ and

$\widehat{p_l p_j p_m} \leq \pi/2$. Thus all these inequalities are actually equalities, and $\mathsf{O\text{-}DISC}(p_i, p_j) = \mathsf{O\text{-}DISC}(p_l, p_m)$. In particular, if $e_1, e_2, e_3 \in E(\mathsf{MGG}(S))$ and e_1 crosses e_2 and e_3, then e_2 and e_3 also cross. If $e_1, e_2, \ldots, e_k$ are pairwise crossing edges then there exists a circle c such that the endpoints of each e_i are at antipodal points on c. Furthermore, there are no points of S in the interior of c. Add edges between points from the set of endpoints of $e_1, e_2, \ldots, e_k$ that are consecutive in c. Notice that these edges do not create crossings with the edges in G. We refer to the interior of the resulting $2k$-gon as a *crossing region*. If a crossing region is bounded by a cycle of 6 or more vertices, remove all the crossing edges and add chords of the cycle so that the region inside the cycle is triangulated (after this modification, the region is no longer called a *crossing region*). Finally, add edges to the region inside $CH(S)$ disjoint from the crossing regions so that each interior face is a triangle. Let G' be the resulting graph. By construction, $|E(G')| \geq |E(G)|$.

The graph G' subdivides the interior of $CH(S)$ into crossing regions and regions bounded by triangles, and the boundaries of the crossing regions are given by rectangles. We denote by $|F_4|$ the number of such rectangles, and by $|F_3|$ the number of triangles. Let G'_{pl} be a graph obtained from G' by deleting one edge in each rectangle. Clearly, G'_{pl} is a triangulation. Therefore, $|E(G'_{pl})| = 3n - h - 3$ and $|E(G')| = 3n - h - 3 + |F_4|$, where h is the size of the convex hull of S. By Euler's formula, $|F_3| + 2|F_4| + n = |E(G'_{pl})| + 1$. Combining these equations, we obtain that $|F_3| = 2n - 2|F_4| - h - 2$.

Suppose, for sake of contradiction, that $|E(G)| \geq 4n - \frac{2\sqrt{n}}{3}$. Then $|F_4| = |E(G')| - 3n + h + 3 \geq |E(G)| - 3n + h + 3 > n - \frac{2\sqrt{n}}{3}$. Consider the subgraph G'' of G' containing the edges of the rectangles bounding the crossing regions. By Lemma 1, the number of edges of G'' that are edges of exactly one rectangle of this graph is at least $4\sqrt{|F_4|}$. We call them *red* edges. Every red edge belongs either to the boundary of a triangle of G' or to the boundary of the convex hull of S. Thus the number of red edges is at most $3|F_3| + h$. Taking $4\sqrt{|F_4|} \leq 3|F_3| + h$, and substituting $|F_3| = 2n - 2|F_4| - h - 2$, we obtain $4\sqrt{|F_4|} \leq 6n - 6|F_4| - 2h - 6$. Since $|F_4| > n - \frac{2\sqrt{n}}{3}$, we get

$$0 < 4\sqrt{n} - 4\sqrt{n - \frac{2\sqrt{n}}{3}} - 2h - 6.$$

But for all $n \geq 1$, we have that $4\sqrt{n} - 4\sqrt{n - \frac{2\sqrt{n}}{3}} \leq 2$, which yields the contradiction. □

Since $|E(\mathsf{MGG}(S))| < 4n$, the handshaking lemma says that $\delta(\mathsf{MGG}(S)) \leq 7$. An example where all vertices have degree at least 6 is drawn in Fig. 1.

As for the maximum degree, MGG might contain a vertex of degree $n-1$. For example, if we place $n-1$ points of S on a circle centered at $p \in S$, in $\mathsf{MGG}(S)$ p is adjacent to all the other vertices.

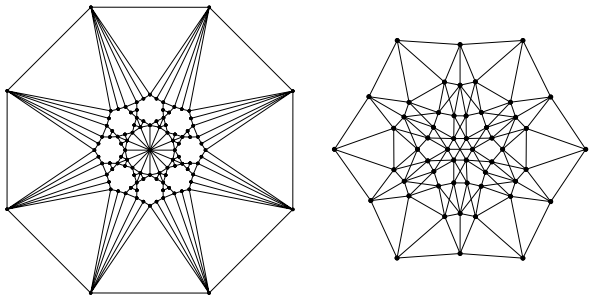


Figure 1: MGG where all vertices have degree at least 6. The interior of each empty dodecagon in the left figure is filled with extra points as in the right figure.

Let us next consider the chromatic number. We have seen that the modified Gabriel graph always contains a vertex of degree at most 7. Observe that, if $p_i p_j$ is an edge of $\mathsf{MGG}(S)$, this edge is also present in $\mathsf{MGG}(S \setminus \{p_l\})$ for any $p_l \in S$ ($p_l \neq p_i, p_j$). Thus, if $\mathsf{MGG}(S) \setminus S'$ is an induced subgraph of $\mathsf{MGG}(S)$ on n' vertices, then it is a subgraph of $\mathsf{MGG}(S \setminus S')$ and it contains a vertex of degree 7 or less. Therefore we can color $\mathsf{MGG}(S)$ with 8 colors using the minimum degree greedy algorithm. However, we have not been able to find such a graph having chromatic number larger than 4, attained by the square grid.

To end this section, we study the clique number.

Proposition 3 *For every point set S, $\omega(\mathsf{MGG}(S)) \leq 4$. This bound is tight.*

Proof. Suppose that a modified Gabriel graph contains a 4-clique whose four vertices form a triangle $\triangle p_i p_j p_k$ with an interior point p_l. Then p_l sees one of the edges of the triangle, for example $p_i p_j$, with an angle greater than $\pi/2$. This contradicts the fact that $p_i p_j$ is an edge of the graph. Consequently, in a modified Gabriel graph the four vertices of any 4-clique are in convex position and form a crossing. As seen in the proof of Theorem 2, in this situation the four vertices are concyclic and form an empty rectangle. This implies that there are not cliques of size greater than 4. To see that this is best possible, notice that MGG of the square grid has many 4-cliques. □

3 The k-Delaunay graph

The number of edges of k-$\mathsf{DG}(S)$ has been studied in [1]. In this paper they show that $|E(k\text{-}\mathsf{DG}(S))| \leq 3(k+1)n - 3(k+1)(k+2)$ and that, if $k < \frac{n}{2} - 1$, then $|E(k\text{-}\mathsf{DG}(S))| \geq (k+1)n$. In the next proposition we describe a k-DG with small number of edges, and one with large number of edges.

Proposition 4 *For any even values of k and $n \geq 4k+4$, there exists a point set S such that $|E(k\text{-DG}(S))| \leq \frac{3kn}{2} + 2k^2 + \frac{5n}{2} - 4k - 6$. For any $k \geq 0$ and $n \geq 2k+3$, there exists a point set S such that $|E(k\text{-DG}(S))| \geq 3nk - 9k^2 + 2n - 11k - 4$.*

Proof. For the first construction, we place points $p_1, \ldots, p_{n/2}$ such that p_i has coordinates $(i, 5)$, and points $q_1, \ldots, q_{n/2}$ such that q_i has coordinates $(i, 0)$ (see Fig. 2, left). Let i be such that $k+2 \leq i \leq n-k-1$ and let $G = k\text{-DG}(S)$. It is not difficult to see that $d_G(q_i) \leq 3k+5$ and $d_G(p_i) \leq 3k+5$. Similarly, if $i < k+2$ or $i > n-k-1$, we have that $d_G(q_i) \leq 4k+2$ and $d_G(p_i) \leq 4k+2$. This yields that $|E(k\text{-DG}(S))| \leq \frac{3kn}{2} + 2k^2 + \frac{5n}{2} - 4k - 6$. The set S can be perturbed so that it becomes non-degenerate.

For the second construction, refer to Fig. 2 (center). The upper and lower groups contain $k+1$ points each, and the remaining points are placed in the middle group. In k-DG(S), the points in the upper group form a clique, and the same holds for the points in the lower group. Each point in the middle group is connected to all upper and lower points, and also to its k predecessors and k successors in the middle group if it has enough points to its left and right. Consequently, if $G = k\text{-DG}(S)$, then $\sum_{q \in S} d_G(q) \geq 2(k+1)(n-k-2) + (n-4k-2)(4k+2)$. This construction can be perturbed so that the non-degeneracy assumptions are satisfied. □

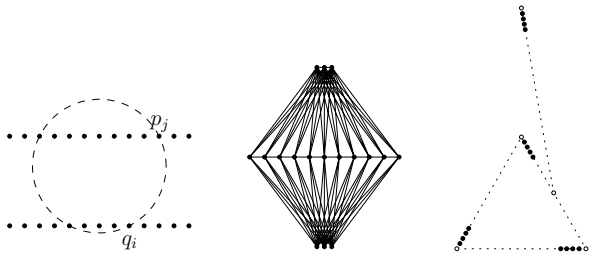


Figure 2: Left and center: sets of points whose k-DG have $\frac{3kn}{2} + o(kn)$ and $3kn + o(kn)$ edges, respectively. Right: set of points whose k-DG is the complete graph.

For every point set S, $\chi(k\text{-DG}(S)) \leq 6k+6$, and there exists a point set S such that $\chi(k\text{-DG}(S)) \geq 4k+4$ [2]. This point set is made of copies of a group of $4k+4$ points illustrated in Fig. 2 (right), whose k-DG is the complete graph. Thus, for this particular S, $\delta(k\text{-DG}(S)) \geq 4k+3$ and $\omega(k\text{-DG}(S)) \geq 4k+4$. In general, the minimum degree of k-DG(S) is at most $6k+5$, by the handshaking lemma. As for the clique number, it is known that, for every set S of points in the plane, there exist two points $p_i, p_j \in S$ such that every circle (the interior and the boundary) containing p_i and p_j contains at least $\frac{n}{4.74}$ points of S [6]. This implies that, for every S, $\omega(k\text{-DG}(S)) \leq 4.74k + 14$.

Finally, we can show that k-DG might contain a vertex of degree $n-1$ using a slightly perturbed version of the example where we place $n-1$ points of S on a circle centered at $p \in S$.

Acknowledgments

This research was initiated at the Bellairs Research Institute of McGill University, Holetown, Barbados. Thanks to the other workshop participants for providing a stimulating working environment. F.H. and V.S. were partially supported by projects MTM2009-07242, Gen. Cat. DGR 2009SGR1040, and ESF EUROCORES programme EuroGIGA, CRP ComPoSe: MICINN Project EUI-EURC-2011-4306, for Spain. M.S. was supported by GraDR EUROGIGA project No. GIG/11/E023, and in part by projects MTM2009-07242 and Gen. Cat. DGR 2009SGR1040.

References

[1] M. Abellanas, P. Bose, J. García, F. Hurtado, C.M. Nicolás, P. Ramos. *On structural and graph theoretic properties of higher order Delaunay graphs.* Internat. J. Comput. Geom. Appl. 19(6):595-615, 2009.

[2] P. Bose, S. Collette, F. Hurtado, M. Korman, S. Langerman, V. Sacristán, M. Saumell. *Some properties of k-Delaunay and k-Gabriel graphs.* Comput. Geom., to appear.

[3] P. Bose, V. Dujmović, F. Hurtado, J. Iacono, S. Langerman, H. Meijer, V. Sacristán, M. Saumell, D.R. Wood. *Proximity graphs: E, δ, Δ, χ and ω.* Submitted.

[4] M.S. Chang, C.Y. Tang, R.C.T. Lee. *Solving the Euclidean bottleneck matching problem by k-relative neighborhood graphs.* Algorithmica 8(3):177-194, 1992.

[5] R.J. Cimikowski. *Properties of some Euclidian proximity graphs.* Pattern Recogn. Lett. 13(6):417-423, 1992.

[6] H. Edelsbrunner, N. Hasan, R. Seidel, X. Shen. *Circles through two points that always enclose many points.* Geom. Dedicata 32:1-12, 1989.

[7] J.W. Jaromczyk, G.T. Toussaint. *Relative neighborhood graphs and their relatives.* Proc. IEEE 80(9):1502-1517, 1992.

[8] D.W. Matula, R.R. Sokal. *Properties of Gabriel graphs relevant to geographic variation research and clustering of points in the plane.* Geogr. Anal. 12(3):205-222, 1980.

[9] T.-H. Su, R.-Ch. Chang. *The k-Gabriel graphs and their applications.* Proc. SIGAL'90, pp. 66-75.

[10] G.T. Toussaint. *The relative neighbourhood graph of a finite planar set.* Pattern Recogn. 12(4):261-268, 1980.

[11] R.B. Urquhart. *Some properties of the planar Euclidean relative neighbourhood graph.* Pattern Recogn. Lett. 1(5):317-322, 1983.

The Erdős-Sós Conjecture for Geometric Graphs

Luis F. Barba * Ruy Fabila-Monroy † Dolores Lara ‡ Jesús Leaños§ Cynthia Rodríguez ‡
Gelasio Salazar ¶ Francisco Zaragoza ‡

Abstract

Let $f(n,k)$ be the minimum number of edges that must be removed from some complete geometric graph G on n points, so that there exists a tree on k vertices that is no longer a planar subgraph of G. In this paper we show that $\left(\frac{1}{2}\right)\frac{n^2}{k-1} - \frac{n}{2} \leq f(n,k) \leq 2\frac{n(n-2)}{k-2}$. For the case when $k = n$, we show that $2 \leq f(n,n) \leq 3$. For the case when $k = n$ and G is a geometric graph on a set of points in convex position, we show that at least three edges must be removed.

1 Introduction

One of the most notorious problems in extremal graph theory is the Erdős-Sós Conjecture, which states that every simple graph with average degree greater than $k-2$ contains every tree on k vertices as a subgraph. This conjecture was recently proved true for all sufficiently large k (unpublished work of Ajtai, Komlós, Simonovits, and Szemerédi).

In this paper we investigate a variation of this conjecture in the setting of geometric graphs. Recall that a *geometric graph* G consists of a set S of points in the plane (these are the vertices of G), plus a set of straight line segments, each of which joins two points in S (these are the edges of G). In particular, any set S of points in the plane in general position naturally induces a complete geometric graph. For brevity, we often refer to the edges of this graph simply as edges *of* S. If S is in convex position then G is a *convex geometric graph*. A geometric graph is *planar* if no two of its edges cross each other. An *embedding* of an abstract graph H into a geometric graph G is an isomorphism from H to a planar geometric subgraph of G. For $r \geq 0$, an *r-edge* is an edge of G such that in one of the two open semi-planes defined by the line containing it, there are exactly r points of G.

In this paper all point sets are in general position and G is a complete geometric graph on n points. It is well known that G contains every tree on k vertices as a planar subgraph [2], for every integer $1 \leq k \leq n$. Moreover, it is possible to embed any such tree into G, when the image of a given vertex is predefined [4].

Let F be a subset of edges of G, which we call *forbidden edges*. If T is a tree for which every embedding into G uses an edge of F, then we say that F *forbids* T. In this paper we study the question of what is the minimum size of F so that there is a tree on k vertices that is forbidden by F. Let $f(n,k)$ be the minimum of this number taken over all complete geometric graphs on n points. As $f(2,2) = 1$, $f(3,3) = 2$, $f(4,4) = 2$ and $f(n,2) = \binom{n}{2}$, we assume through out the paper that $n \geq 5$ and $k \geq 3$.

We show the following bounds on $f(n,k)$.

Theorem 1

$$\left(\frac{1}{2}\right)\frac{n^2}{k-1} - \frac{n}{2} \leq f(n,k) \leq 2\frac{n(n-2)}{k-2}$$

Theorem 2

$$2 \leq f(n,n) \leq 3$$

In the case when G is a convex complete geometric graph, we show that the minimum number of edges needed to forbid a tree on n vertices is three. Some results shown in [3] are closely related to this problem.

An equivalent formulation of the problem studied in this paper is to ask how many edges must be removed from G so that it no longer contains *some* planar subtree on k vertices. A different but related problem is to ask how many edges must be removed from G, so that it no longer contains *any* planar subtree on k vertices. For the case of $k = n$, in [5], it is proved that if any $n-2$ edges are removed from G, it still contains a planar spanning subtree. Note that if the $n-1$ edges incident to any vertex of G are removed, then G no longer contains a spanning subtree. In general, for $2 \leq k \leq n-1$, in [1], it is proved that if any set of $\left\lceil \frac{n(n-k+1)}{2} \right\rceil - 1$ edges are removed from G, it still contains a planar subtree on k vertices. In the same paper it is also shown that this bound is tight

2 Spanning Trees

In this section we consider the case when $k = n$. Let T be a tree on n vertices. Consider the following algorithm to embed T into G. Choose a vertex v of T; root T at v. For every vertex of T choose an arbitrary

*Instituto de Matemáticas, UNAM
†Departamento de Matemáticas, CINVESTAV. Partially supported by Conacyt of Mexico, Grant 153984.
‡Departamento de Sistemas, UAM Azcapotzalco
§Escuela de Matemáticas, UAZ
¶Instituto de Física, UASLP

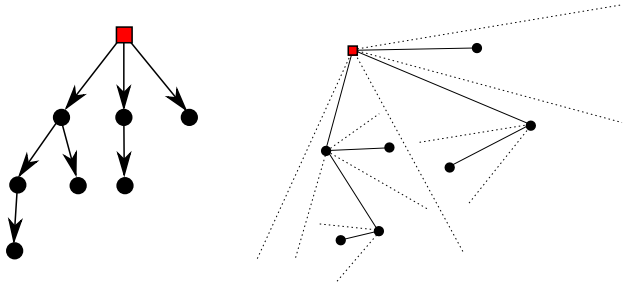


Figure 1: An embedding of a tree using the algorithm.

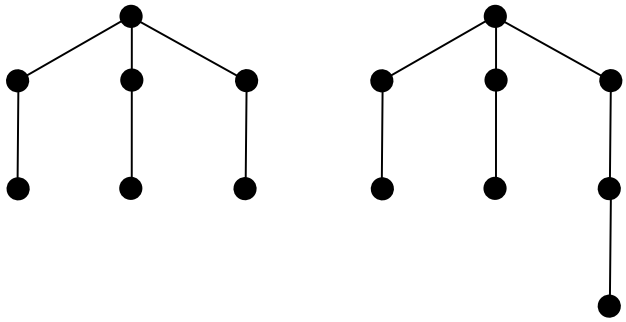


Figure 2: T_7 and T_8.

order of its children. Suppose that the neighbors of v are $u_1, \ldots, u_m$, and let $n_1, \ldots, n_m$ be the number of nodes in their corresponding subtrees. Choose a convex hull point p of G and embed v into p. Sort the remaining points of G counter-clockwise by angle around p. Choose $m+1$ rays centered at p so that the wedge between two consecutive rays is convex and between the i-th ray and the $(i+1)$-th ray there are exactly n_i points of G. Let S_i be this set of points. For each u_i choose a convex hull vertex of S_i visible from p and embed u_i into this point. Recursively embed the subtrees rooted at each u_i into S_i. Note that this algorithm provides an embedding of T into G. We will use this embedding frequently throughout the paper. See Figure 1.

For every integer $n \geq 2$ we define a tree T_n as follows: If $n = 2$, then T_n consists of only one edge; if n is odd, then T_n is constructed by subdividing once every edge of a star on $\frac{n-1}{2}$ vertices; if n is even and greater than 2, then T_n is constructed by subdividing an edge of T_{n-1}. These trees are particular cases of *spider trees*. See Figure 2.

We prove the lower bound of $f(n,n) \geq 2$ of Theorem 2.

Theorem 3 *If G has only one forbidden edge, then any tree on n vertices can be embedded into G, without using the forbidden edge.*

Proof. Let e be the forbidden edge of G. Let T be a tree on n vertices. Choose a root for T. Sort the children of each node of T, by increasing size of their corresponding subtree. Embed T into G with the embedding algorithm, choosing at all times the rightmost point as the root of the next subtree. Suppose that e is used in this embedding. Let $e := (p, q)$ so that u is embedded into p and v is embedded into q (note that u and v are vertices of T).

Suppose that the subtree rooted at v has at least two nodes. In the algorithm, we embedded this subtree rooted at v into a set of at least two points. We chose a convex hull point (q), of this set visible from p to embed v. In this case we may choose another convex hull point visible from p to embed v and continue with the algorithm. Note that (p, q) is no longer used in the final embedding.

Suppose that v is a leaf, and that v has a sibling v' whose subtree has at least two nodes. Then we may change the order of the children of u so that e is no longer used in the embedding, or if it is, then v' is embedded into q, but then we proceed as above.

Suppose that v is a leaf, and that all its siblings are leaves. The subtree rooted at u is a star. We choose a point distinct from p and q in the point set where this subtree is embedded, and embed u into this point. Afterwards we join it to the remaining points. This produces an embedding that avoids e.

Assume then, that v is a leaf and that it has no siblings. We distinguish the following cases:

1. u **has no siblings.** In this case, the subtree rooted at the parent of u is a path of length two. It is always possible to embed this subtree without using e. See Figure 3.

2. u **has a sibling u' whose subtree is not an edge.** We may change the order of the siblings of u, with respect to their parent, so that the subtree rooted at u' will be embedded into the point set containing p and q. In the initial order—increasing by size of their corresponding subtrees—u' is after u. We may assume that in the new ordering, the order of the siblings of u before it, stays the same. Therefore p is the rightmost point of the set into which the subtree rooted at u' will be embedded. Embed u' into p. Either we find an embedding not using e, or this embedding falls into one of the cases considered before.

3. u **has at least one sibling, all whose corresponding subtrees are edges**

   Suppose that u has no grandparent; then T is equal to T_n and n is odd. Let w be the parent of u. Embed w into p. Let $p_1, \ldots, p_{n-1}$ be the points of G different from p sorted counterclockwise by angle around p; choose p_1 so that the angle between two consecutive points is less than π. Let $u_1, \ldots, u_{(n-1)/2}$ be the neighbors of

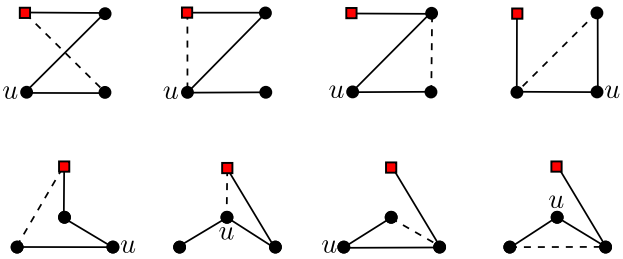


Figure 3: The embedding of a path of length three. The grandparent of u is highlighted and the forbidden edge is dashed.

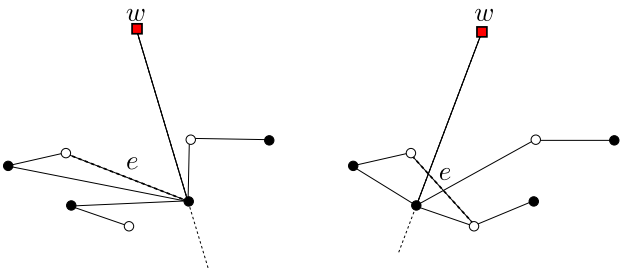


Figure 4: The two sub-cases, when u has a grandparent w, and all the subtrees of its children are edges. Odd points are painted in black and even points in white. The forbidden edges are dashed.

w. Embed each u_i into p_{2i-1} and its child into p_{2i}. If q equals p_{2j-1} for some j then embed u_j into p_{2j} and its child into p_{2j-1}. This embedding avoids e.

Suppose that w is the grandparent of u and let p' be the point into which w is embedded. Let S be the point set into which the subtree rooted at the parent of u is embedded. Note that S has an odd number of points. We replace the embedding as follows. Sort S counter-clockwise by angle around p'. Call a point *even* if it has an even number of points before it in this ordering. Call a point *odd* if it has an odd number of points before it in this ordering. If e is incident to an odd point, then we embed the parent of u into this point. The remaining subtree rooted at u can be embedded without using e. If the endpoints of e are both even, between them there is an odd point. We embed the parent of u into this point. The remaining vertices can be embedded without using e (see Figure 4).

□

The upper bound of $f(n,n) \leq 3$ of Theorem 2 follows directly from Lemma 6. Now we prove in Lemma 4 and Theorem 5, that if G is a convex geometric graph, at least three edges are needed to forbid some tree on n vertices. Lemma 4 can be proved easily using a previous result (Theorem 2.1 of [3]). We provide a self-contained proof for completeness.

Lemma 4 *Let T be a tree on n vertices. If G is a convex geometric graph, then T can be embedded into G using less than $\frac{n}{2}$ convex hull edges of G.*

Proof. If T is a star, then any embedding of T into G uses only two convex hull edges. If T is a path then it can be embedded into G using at most two convex hull edges. Therefore, we may assume that T is neither a star nor a path.

Since T is not a path, it has a vertex of degree at least three. Choose this vertex as the root. Since T is not a star, the root has a child whose subtree has at least two nodes. Sort the children of T so that this node is first. Embed T into G with the embedding algorithm.

Let u and v be vertices of T, so that u is the parent of v. Suppose that the subtree rooted at v has at least two nodes. Then in the embedding algorithm we have at least two choices to embed v once the ordering of the children of u has been chosen. At least one of which is such that (u,v) is not embedded into a convex hull edge. Therefore, we may assume that the embedding is such that all the convex hull edges used are incident to a leaf.

Since the first child of the root is not a leaf, there is at most one convex hull edge incident to the root in the embedding. Note that any vertex of T, other than the root, is incident to at most one convex hull edge in the embedding. If $n/2$ or more convex hull edges are used, then there are at least $n/2$ non-leaf vertices, each adjacent to a leaf. These vertices must be all the vertices in T and there are only $n/2$ such pairs (n must also be even). Therefore every non-leaf vertex has at most one child which is a leaf. In particular the root has at most one child which is a leaf. Since the root was chosen of degree at least three it has a child which is not a leaf nor the first child; we place this vertex last in the ordering of the children of the root. The leaf adjacent to the root can no longer be a convex hull edge and the embedding uses less than $n/2$ convex hull edges. □

Theorem 5 *If G is a convex geometric graph and has at most two forbidden edges, then any tree on n vertices can be embedded into G, without using a forbidden edge.*

Proof. Let f_0 be an embedding given by Lemma 4, of T into G. For $0 \leq i \leq n$, let f_i be the embedding produced by rotating f_0, i places to the right. Assume that in each of these rotations at least one forbidden edge is used, as otherwise we are done. Let $e_1, \ldots, e_m$ be the edges of T that are mapped to a forbidden

edge in some rotation. Assume that the two forbidden edges are an l-edge and an r-edge respectively.

Suppose that $l \neq r$. Then, each edge of T can be embedded into a forbidden edge at most once in all of the n rotations. Thus $m \geq n$. This is a contradiction, since T has $n-1$ edges.

Suppose that $l = r$. Then, each of the e_i is mapped twice to a forbidden edge. Thus $m \geq n/2$. By Lemma 4, f_0 uses less than $n/2$ convex hull edges. Therefore, l and r must be greater than 0. But a set of $n/2$ or more r-edges, with $r > 0$, must contain a pair of edges that cross. And we are done, since f_0 is an embedding. □

3 Bounds on $f(n,k)$

In this section we prove Theorem 1. First we show the upper bound which can also be seen as a consequence of Theorem 2.2 of [3]. However, we provide a self-contained proof for completeness.

Lemma 6 *If G is a convex geometric graph, then forbidding three consecutive convex hull edges of G forbids the embedding of T_n.*

Proof. Recall that T_n comes from subdividing a star, let v be the non leaf vertex of this star. Let $(p_1,p_2),(p_2,p_3),(p_3,p_4)$ be the forbidden edges, in clockwise order around the convex hull of G. Note that in any embedding of T_n into G, an edge incident to a leaf of T_n, must be embedded into a convex hull edge. Thus, the leaves of T_n nor its neighbors can be embedded into p_2 or p_3, without using a forbidden edge. Thus, v must be embedded into p_2 or p_3. Without loss of generality assume that v is embedded into p_2. But then, the embedding must use (p_2,p_3) or (p_3,p_4) □

Lemma 7 *If G is a convex geometric graph, then forbidding any three pairs of consecutive convex hull edges of G forbids the embedding of T_n.*

Proof. Let p_1, p_2 and p_3 be the vertices in the middle of the three pairs of consecutive forbidden edges of G. Note that a leaf of T_n, nor its neighbor can be embedded into p_1, p_2 or p_3, without using a forbidden edge. But at most two points do not fall into this category. □

Lemma 8 $f(n,k) \leq 2\frac{n(n-2)}{k-2}$

Proof. Let G be a complete convex geometric graph. We forbid every r-edge of G for $r = 0,\ldots,\left\lceil 2\frac{n-2}{k-2}-2\right\rceil$. Note that, in total we are forbidding at most $n\left(\left\lceil 2\frac{n-2}{k-2}-2\right\rceil+1\right) \leq 2\frac{n(n-2)}{k-2}$ edges. As every subset of points of G is in convex position, it suffices to show that every induced subgraph H of G on k vertices is in one of the two configurations of Lemma 6 and 7.

Assume then, that H does not contain three consecutive forbidden edges in its convex hull nor three pairs of consecutive forbidden edges in its convex hull. H has at most two (non-adjacent) pairs of consecutive forbidden edges in its convex hull. Therefore every forbidden edge of H in its convex hull—with the exception of at most two—must be preceded by an ℓ-edge (of G), with $\ell > \left\lceil 2\frac{n-2}{k-2}-2\right\rceil$. H contains at least $\frac{k-2}{2}$ of these edges. The points separated by these edges amount to more than $\frac{k-2}{2}\left\lceil 2\frac{n-2}{k-2}-2\right\rceil \geq n-k$ points of G. Together with the k points of H this is strictly more than n—a contradiction.

□

Now, we show the lower bound of Theorem 1.

Lemma 9 $f(n,k) \geq \left(\frac{1}{2}\right)\frac{n^2}{k-1} - \frac{n}{2}$

Proof. Let F be a set of edges whose removal from G forbids some k-tree. Let $H := G \setminus F$. Note that H contains no complete K_k as a subgraph, otherwise any k-tree can be embedded in this subgraph [2]. By Turán's Theorem [6], H cannot contain more than $\left(\frac{k-2}{k-1}\right)\frac{n^2}{2}$ edges. Thus F must have size at least $\left(\frac{1}{2}\right)\frac{n^2}{k-1} - \frac{n}{2}$. □

Acknowledgments

Part of this work was done at the "First Workshop in Combinatorial Optimization at Cinvestav". It was continued during a visit of L.F. Barba, R. Fabila-Monroy, J. Leaños and G. Salazar to Abacus research center.

References

[1] O. Aichholzer, S. Cabello, R. Fabila-Monroy, D. Flores-Peñaloza, T. Hackl, C. Huemer, F. Hurtado, and D. R. Wood. Edge-removal and non-crossing configurations in geometric graphs. *Discrete Math. Theor. Comput. Sci.*, 12(1):75–86, 2010.

[2] P. Bose, M. McAllister, and J. Snoeyink. Optimal algorithms to embed trees in a point set. *J. Graph Algorithms Appl.*, 1:no. 2, 15 pp. (electronic), 1997.

[3] A. García, C. Hernando, F. Hurtado, M. Noy, and J. Tejel. Packing trees into planar graphs. *J. Graph Theory*, 40(3):172–181, 2002.

[4] Y. Ikebe, M. A. Perles, A. Tamura, and S. Tokunaga. The rooted tree embedding problem into points in the plane. *Discrete Comput. Geom.*, 11(1):51–63, 1994.

[5] G. Károlyi, J. Pach, G. Tóth, and P. Valtr. Ramsey-type results for geometric graphs. II. *Discrete Comput. Geom.*, 20(3):375–388, 1998. ACM Symposium on Computational Geometry (Nice, 1997).

[6] P. Turán. On an extremal problem in graph theory. *Mat. Fiz. Lapok*, 48:436–452, 1941.

Planar β-skeletons via point location in monotone subdivisions of subset of lunes

Miroslaw Kowaluk*

Abstract

We present a new algorithm for lune-based β-skeletons for sets of n points in the plane, for $\beta \in (2, \infty]$, the only case when optimal algorithms are not known. The running time of the algorithm is $O(n^{3/2} \log^{1/2} n)$, which is the best known and is an improvement of Rao and Mukhopadhyay [10] result. The method is based on point location in monotonic subdivisions of arrangements of curve segments.

1 Introduction

β-skeletons [7] belong to the family of proximity graphs, geometric graphs in which two vertices (points) produce an edge if and only if they satisfy particular geometric requirements. The proximity graphs are both important and popular because of many practical applications. They span a broad spectrum of areas, from earlier work on mathematical morphology on lattices and graphs, pattern recognition, geographic information systems, data mining to more recent applications in ad hoc wireless networks and systems biology. Requirements defining proximity graphs can be formulated in many metrics, with the Euclidean metric being one of the most commonly used.

Definition 1 *For a given set P containing n points in R^2, distance function d and the parameter β we define β-skeleton as a graph (P, E), in which $xy \in E$ iff no point in $P \setminus \{x, y\}$ belongs to $R(x, y, \beta)$, where*

- *for $\beta = 0$, $R(x, y, \beta)$ is the segment xy;*
- *for $0 < \beta < 1$, $R(x, y, \beta)$ is the intersection of two discs with the radius $d(x, y)/2\beta$, which boundaries contain the both points x and y.*

For $1 \leq \beta \leq \infty$ there are two ways to define the region $R(x, y, \beta)$ leading to two different families of graphs. The lune-based definition is as follows:

- *For $1 \leq \beta \leq \infty$, $R(x, y, \beta)$ is a intersection of two discs with radius $\beta d(x, y)/2$ and centered in points $(1-\beta/2)x+(\beta/2)y$ and $(\beta/2)x+(1-\beta/2)y$, respectively.*
- *For $\beta = \infty$, $R(x, y, \beta)$ is an unbounded strip between two lines containing x and y, respectively, and perpendicular to the segment xy.*

The second definition defining a different family of graphs is circle-based.

- *For $1 \leq \beta < \infty$, $R(x, y, \beta)$ is the union of two discs, each having diameter $\beta d(x, y)$ and having the segment xy as its chord.*
- *For $\beta = \infty$, $R(x, y, \beta)$ is the union of hyperplanes having the segment xy on their border.*

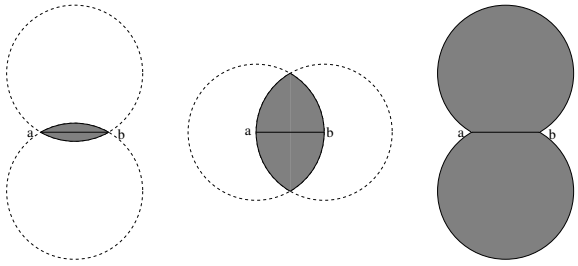


Figure 1: The region $R(a, b, \beta)$ for $0 < \beta < 1$ (left), the lune-based region $R(a, b, \beta)$ for $1 \leq \beta < \infty$ (middle) and the circle-based region $R(a, b, \beta)$ for $1 \leq \beta < \infty$ (right).

The open (closed, respectively) area $R(x, y, \beta)$ defines an open (closed, respectively) β-skeleton. A Gabriel Graph (GG) [4] is a closed 1-skeleton and a Relative Neighborhood Graph (RNG) [12] is an open 2-skeleton.

For $0 \leq \beta < 1$, there are in the worst case optimal $O(n^2)$ time algorithms [5] for both lune-based and circle-based β-skeletons. Also, any β-skeleton for $1 \leq \beta \leq 2$ can be computed in the optimal $O(n \log n)$ time ([6, 8, 9, 11]). For circle-based β-skeletons, where $1 \leq \beta \leq \infty$, there also is an optimal $O(n \log n)$ time algorithm [7].

For $\beta > 2$, the lune-based and circle-based regions $R(x, y, \beta)$ are very different and it has been reflected by higher running times of the best known algorithms for the lune-based β-skeletons. In particular, for $2 < \beta \leq \infty$, the fastest algorithm that has been reported for computing lune-based β-skeletons

*Institute of Informatics, University of Warsaw, kowaluk@mimuw.edu.pl

requires $O(n^{3/2} \log n)$ time [10]. The algorithm uses the sweeping-line technique.
In this paper, we will show a $O(n^{3/2} \log^{1/2} n)$ time algorithm computing a lune-based β-skeleton for $2 < \beta \leq \infty$. To this end, we will use arrangements of curve segments [1] and data structure for point location in monotone subdivisions [3].

2 Algorithm

It is well known, see e.g. [7], that β-skeletons for $2 < \beta$ are subgraphs of the Delaunay triangulation of the input point sets. For a set P of n points, they have size $O(n)$. Thus, to construct a β-skeleton, we can use their definition to eliminate edges that do not belong to β-skeleton by comparing edges of Delaunay triangulation against all of the points in the input set. To this end, we analyze lunes (for $2 < \beta < \infty$) or strips (for $\beta = \infty$) defined by the edges of the DT triangulation.
Specifically, we have to identify these lunes or strips that contain points in P.
Let us consider m edges of the Delaunay triangulation of P. m will be determined later to minimize the running time.

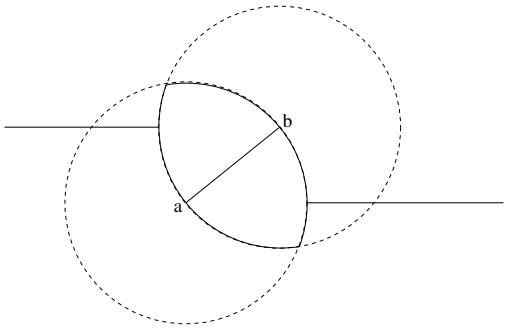


Figure 2: A subdivision of the plane into three monotone regions for one lune.

We construct arrangement of lunes defined by these edges. Specifically, for each of the lunes, we draw two horizontal half-lines that start at the points with the smallest and, respectively, largest x-coordinates. They yield three monotone regions per lune (see Fig. 2). The size of the arrangement is $O(m^2)$ and the time complexity of its construction is the same [1].
The intersection of monotone regions is monotone. The above construction results in a monotone subdivision of size $O(m^2)$ for the arrangement of size $O(m^2)$.
Then, we locate points of P in regions of the subdivision. The construction time of an auxiliary data structure is linear with respect to a size of the subdivision [3]. Hence it can be done in $O(m^2)$ time and $O(m^2)$ space. A query time is $O(\log m)$. The results of querying are stored in respective cells of the subdivision data structure.
We traverse the dual graph of the subdivision in a Depth-First-Search fashion in order to identify (and store) edges of the Delaunay triangulations that do not belong the β-skeleton and lunes (strips, respectively) containing currently visited region of the subdivision.

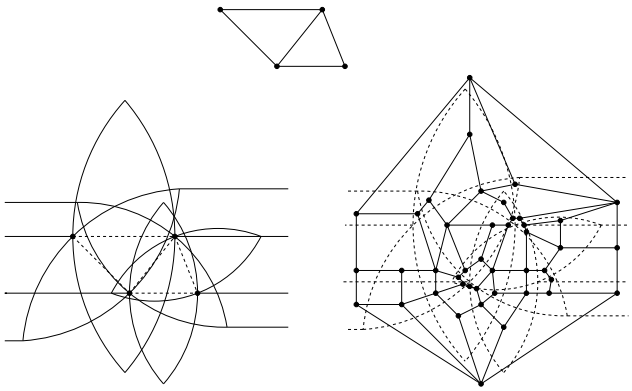


Figure 3: The Delaunay triangulation for a set of four points, the corresponding subdivision and its dual graph.

The structure is a table of lunes (strips, respectively) forming the subdivision with

- records storing - value *true* when the respective lune contains some point of P and *false* otherwise and
- pointers to a double linked list of lunes (strips, respectively) containing currently visited region which respective records are *false*.

Moving between two neighboring regions we visit or leave the lune or our status stays unchanged.
In the first case :

- if record of the auxiliary structure of this lune has value *false* we add the lune to the list in the structure; otherwise the list stays unchanged,
- if visited region contains a point of P we change values of records of all lunes belonging to the list on *true* and erase the list.

In the second case we remove the lune form the list if its record has value *false*.
In the third case all data structures stay unchanged.
Visiting all regions of the subdivision take $O(m^2)$ time.
We repeat the above process for all the groups in the partition of the edges into groups of size m.

Theorem 1 *For $2 < \beta \leq \infty$, β-skeleton of a set P of n points in the plane can be constructed in $O(n^{3/2} \log^{1/2} n)$ time and $O(n \log n)$ space.*

Proof. The time complexity of every steps of the algorithm is $O(n \log n) + n/m(O(m^2) + O(m^2) + O(n \log m) + O(m^2)) = O(n \log n) + n/m(O(m^2) + O(n \log m))$. The value of this function is minimal for $m = (n \log n)^{1/2}$. Hence the time complexity of the algorithm is $O(n^{3/2} \log^{1/2} n)$. Space complexity of the algorithm is $O(n) + O(m^2) = O(n \log n)$. □

3 Conclusion

We have presented a new algorithm for the lune based β-skeletons, where $\beta > 2$. For circle-based skeletons the approach would work as well, however better algorithms are known in that case. Although the algorithm provides only an incremental improvement compared to the previous one, the improvement is interesting due to a tantalizing jump in the algorithmic complexity of the β-skeleton problem that occurs for $\beta = 2$. While for $\beta \leq 2$ optimal algorithms are known, the optimality is open for $\beta > 2$. The presented algorithm narrows the gap, however more research is needed to better understand the reasons for this apparent difficulty for $\beta > 2$.

Finally, we believe that the point location approach can be also used for metrics other than Eucliedean.

References

[1] N. M. Amato, M. T. Goodrich, E. A. Ramos. *Computing the arrangement of curve segments: Divide-and-conquer algorithms via sampling.* Proc. 11th ACM-SIAM Sympos. Discrete Algorithms, 2000, 705-706.

[2] J. Cardinal, S. Collette, S. Langerman. *Empty region graphs.* Comput. Geom. 42(3), 2009, pp. 183-195.

[3] H. Edelsbrunner, L. J. Guibas, J. Stolfi. *Optimal point location in a monotone subdivision.* SIAM Journal on Computing, 15(2):317-340, 1986.

[4] K.R. Gabriel, R.R. Sokal. *A new statistical approach to geographic variation analysis.* Systematic Zoology 18 (1969) 259-278.

[5] F. Hurtado, G. Liotta, H. Meijer. *Optimal and suboptimal robust algorithms for proximity graphs.* Computational Geometry 25 (2003), pp. 35-49.

[6] J.W. Jaromczyk, M. Kowaluk, F. Yao. *An optimal algorithm for constructing β-skeletons in l_p metric.* Manuscript, 1989.

[7] D.G. Kirkpatrick, J.D. Radke. *A framework for computational morphology.* in: G.T. Toussaint (Ed.), Computational Geometry, North-Holland, Amsterdam, 1985, pp. 217-248.

[8] A. Lingas. *A linear-time construction of the relative neighborhood graph from the Delaunay triangulation.* Computational Geometry 4 (1994) 199-208.

[9] D.W. Matula, R.R. Sokal. *Properties of Gabriel graphs relevant to geographic variation research and clustering of points in the plane.* Geogr. Anal. 12 (3) (1980) 205-222.

[10] S.V. Rao, A. Mukhopadhyay. *Fast algorithm for computing β-skeletons and their relatives.* Proc. 8th Annu. Int. Symp. on Algorithms and Computation, in: Lecture Notes Comput. Sci., Vol. 1350, 1997, pp. 374-383.

[11] K.J. Supowit. *The relative neighborhood graph with an application to minimum spanning trees.* J. ACM 30 (3) (1983) 428-448.

[12] G. Toussaint. *The relative neighborhood graph of a finite planar set.* Pattern Recognition 12 (1980), 261-268.

Some Sparse Yao Graphs are Spanners

Matthew Bauer * Mirela Damian †

Abstract

We show that, for any integer $k \geq 3$, the Sparse-Yao graph (also known as a Yao-Yao graph) YY_{4k} constructed under the L_∞ norm is a spanner.

1 Introduction

Let $\mathcal{P}$ be a set of points in the plane of cardinality n. We will refer to the points in $\mathcal{P}$ as *nodes*, to distinguish them from other points in the plane. The Yao graph Y_k, with vertex set $\mathcal{P}$ and integer parameter $k > 0$, is defined as follows. Fix a coordinate system and shoot k, equally separating rays from the origin, with the first ray in the direction of the x-axis. These rays divide the plane into k cones of angle $2\pi/k$ each. Translate the cones to each node $a \in \mathcal{P}$. In each cone with apex a, pick a closest node b, if there is one, and add to Y_k the directed edge $\overrightarrow{ab}$. Ties are broken arbitrarily. The Yao construction guarantees that the out-degree of Y_k is bounded by k. However, a node can be nearest to many other nodes. For example, if $\mathcal{P}$ consists of $n-1$ nodes placed on the circumference of a circle with center node a, then each of the $n-1$ nodes will have an outgoing edge to a, because a is a nearest node in one of their cones. In this case, node a has out-degree k, but in-degree $n-1$.

To overcome the problem of potential high in-degree at a node, two straightforward Yao-based graphs have been introduced. The Symmetric-Yao graph SY_k keeps only the symmetric (bidirectional) Yao edges, and discards the rest: an edge $ab \in SY_k$ if and only if both $\overrightarrow{ab}$ and $\overrightarrow{ba}$ are in Y_k. Thus, the degree of SY_k is bounded above by k. One issue with the Symmetric-Yao graph is that it introduces very long detours, so it is possible that no "short" paths can be found between two nodes in SY_k. In an effort to overcome this new issue, the Sparse-Yao graph YY_k, also known as Yao-Yao graph, has been defined. Both names are suggestive, in the sense that YY_k is a sparse graph obtained by applying a second Yao step to the set of incoming Yao edges: for each node a and each cone rooted at a containing two or more incoming edges, retain a shortest incoming edge and discard the rest. (Ties are broken arbitrarily.) Then the degree of YY_k is bounded above by $2k$ (at most k outgoing edges and at most k incoming edges at each node). It is yet unknown whether the Sparse-Yao graph contains "short" paths between all pairs of nodes or not. Here, "short" paths are quantified by the notion of *t-spanner*. The *length* of a path is the sum of the lengths of its constituent edges. For a fixed constant $t \geq 1$, a graph G is a *t-spanner* if a shortest path between any pair of nodes $a, b \in G$ is at most t-times longer than the Euclidean distance between a and b.

The class of Yao-based graphs has been much studied. Table 1 summarizes the spanning properties of these graphs, to the best of our knowledge. Note that it is yet unknown whether all Sparse-Yao graphs, for large enough values of k, are spanners.

Our focus in this paper is on the open problem listed in the shaded cell of Table 1.

Graph	Parameter k	t-Spanner
Yao Y_k	$k \in \{2,3\}$	NO [4]
	$k = 4$	$t = O(1)$ [1]
	$k = 5$	OPEN
	$k = 6$	$t = O(1)$ [3]
	$k > 6$	$t = O(1)$ [1, 7]
Sparse Yao YY_k	$k \leq 4$	NO [5]
	$k = 5$	OPEN
	$k = 6$	NO [4]
	$k > 6$	OPEN
Symmetric Yao SY_k	$k \geq 1$	NO [6]

Table 1: Spanning properties of Yao graphs.

2 Preliminaries

For a fixed k, let $r_1, r_2, \ldots, r_k$ be the rays separating the cones at each point. We assume, without loss of generality, that r_1 is horizontal and points to the right. For each node $a \in \mathcal{P}$, we use $C_{k,i}(a)$ to denote the half-open cone delimited by r_i and r_{i+1}, including r_i but excluding r_{i+1}. Here, r_{i+1} refers to the first ray counter-clockwise from r_i and r_j refers to $r_{j \bmod (k+1)+1}$, for any j. See Fig. 1(a). For a point a, let x_a denote the x-coordinate of a, and y_a the y-coordinate of a. We will distinguish between Euclidean L_2 distances,

*Department of Computing Sciences, Villanova University, Villanova, PA, USA. mbauer03@villanova.edu.

†Department of Computing Sciences, Villanova University, Villanova, PA, USA. mirela.damian@villanova.edu.

$|ab| = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}$), and supremum L_∞ distances $|ab|_\infty = \max\{|x_a - x_b|, |y_a - y_b|\}$.

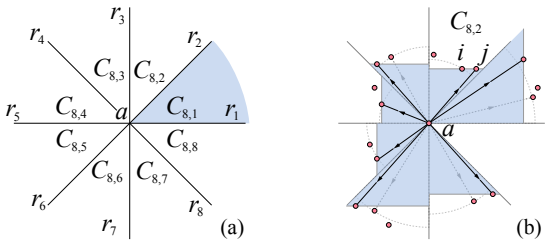


Figure 1: Definitions. (a) Cones $C_{k,i}$, for $k = 8$. (b) Edges in Y_8^∞ outgoing from a.

A *path* p is defined as a sequence of edges. The length of p is denoted $|p|$ under the L_2 norm and $|p|_\infty$ under the L_∞ norm. Let Y_k^∞ denote the Yao graph constructed using L_∞ distances between nodes. Unlike Y_k, in constructing Y_k^∞, ties are broken by always selecting the *most clockwise* edge. Fig. 1b shows the Y_8^∞ edges outgoing from node a. (Dashed light-shaded edges are the original Y_8 edges, and the darker solid edges are the Y_8^∞ edges.) Note that the two nodes labeled i and j in Fig. 1b are at the same L_∞ distance from a, and Y_8^∞ selects the most clockwise edge $(\overrightarrow{aj})$. Similar definitions hold for YY_k^∞, the Sparse-Yao graph in the L_∞ metric. We now state a result established in [1], which we will use in our proofs in Sec. 3.

Theorem 1 [1] *Y_4^∞ is a plane graph. In addition, Y_4^∞ is an 8-spanner in the L_∞ metric. In other words, for any two nodes a and b, there is a path between a and b whose length (in the L_∞-metric) is at most $8|ab|_\infty$.*

For any two points a and b, define $R(a, b)$ to be the *closed* axis-aligned rectangle with diagonal ab and define ∂R to be the boundary of R. For any two nodes $a, b \in \mathcal{P}$ and fixed plane rectangle R, let $\Pi_{[Y_4]}(a, b, R)$ denote the Greedy path that starts at a and follows the directed edges in Y_4^∞ pointing towards b, until it exits $R \setminus \partial R$. This idea is depicted in Fig. 2a, for $k = 1$ and rectangle $R = R(a, b)$. Although possible, a formal definition of $\Pi_{[Y_4]}$ is hardly necessary. Additionally, from here on, we will ignore the parameter R, with the implicit assumption that $\Pi_{[Y_4]}(a, b)$ is defined with respect to $R(a, b)$. We also remind the reader that, although we do not explicitly use the symbol ∞ in the definition of $\Pi_{[Y_4]}$, it should be understood that all path edges are from Y_4^∞.

We say that two segments ab and cd *properly cross* (or *cross*, for short) if they share a point other than an endpoint (a, b, c or d); we say that

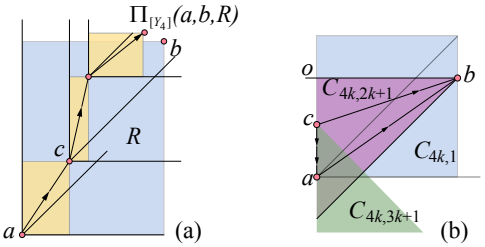


Figure 2: (a) Rectangle $R = R(a, b)$ and path $\Pi_{[Y_4]}(a, b, R)$. (b) Lemma 3, case 1: ac vertical.

ab and cd *intersect* if they share either an interior point or an endpoint (so two segments that cross also intersect, but the reverse is not necessarily true). Most often we ignore the direction of an edge ab; we refer to the directed version $\overrightarrow{ab}$ of ab only when its origin (a) is important and unclear from the context.

3 Spanning ratio under the L_∞ norm

Throughout this section, lengths and distances are measured under the L_∞ norm. Our goal here is to prove that YY_{4k}^∞ is a spanner in the L_∞ metric. Our approach takes advantage of the plane property of Y_4^∞ stated in Thm. 1, and seeks short paths in YY_{4k}^∞ between the endnodes of each edge in Y_4^∞. Combined with the result of Thm. 1, this yields short paths between any pair of nodes in $\mathcal{P}$. Fundamental to our approach is also the immediate property stated in Lem. 2 below. (We skip the proof of this lemma, due to space constraints.)

Lemma 2 *Y_4^∞ is a subgraph of Y_{4k}^∞, for $k \geq 1$.*

Lemma 3 *Fix $k \geq 3$ and let $\theta = \frac{\pi}{2k}$ be the angle between consecutive rays in YY_{4k}^∞. For each edge $ab \in Y_4^\infty$, the graph YY_{4k}^∞ contains a path $pp(a, b)$ between a and b of length $|pp(a, b)|_\infty \leq t|ab|_\infty$, for any $t \geq \frac{1}{1-\tan\theta}$.*

Proof. The proof is by induction on the length of the Y_4^∞ edges. For the base case, consider an edge $\overrightarrow{ab} \in Y_4^\infty$ of minimum length $|ab|_\infty$. For simplicity, rotate the point set $\mathcal{P}$ so that ab lies in the cone $C_{4k,1}(a)$ and $|ab|_\infty = |x_b - x_a|$. By Lem. 2, $\overrightarrow{ab} \in Y_{4k}^\infty$. We now show that $\overrightarrow{ab} \in YY_{4k}^\infty$. Assume to the contrary that $\overrightarrow{ab} \notin YY_{4k}^\infty$. Let $C = C_{4k,2k+1}(b)$ be the cone with apex b containing a. (Refer to Fig. 2b.) Because C contains at least one incoming edge from Y_{4k}^∞ (in particular it contains $\overrightarrow{ab}$), C must also contain one edge $\overrightarrow{cb} \in YY_{4k}^\infty$, selected in the second Yao step. If $|cb|_\infty < |ab|_\infty$, we have an immediate contradiction to $|ab|_\infty$ being minimum. If $|cb|_\infty = |ab|_\infty$,

This is an extended abstract of a presentation given at EuroCG 2012. 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 in a conference with formal proceedings and/or in a journal.

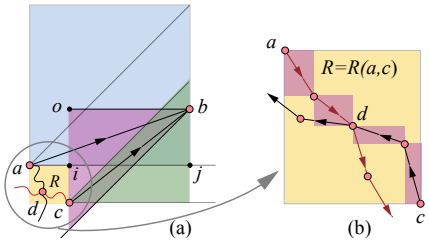


Figure 3: Lemma 3, case 2 (a) c right of a. (b) $\Pi_{[Y_4]}(a,c)$ and $\Pi_{[Y_4]}(c,a)$ meeting in node d.

then cb must lie clockwise from ab with respect to b (by the ways ties are broken). This situation is depicted in Fig. 2b. Note that the cone C does not include the lower boundary line (recall that cones are half-open and half-closed), and therefore a cannot lie on the lower boundary line of C. It follows that $|ac|_\infty = |y_c - y_a| < |x_b - x_a| = |ab|_\infty$ (because $\angle abc < \frac{\pi}{2k} \leq \pi/6$). In this case, $\overrightarrow{ac}$ would be a potential candidate for the edge outgoing from a in Y_4^∞, thus contradicting our assumption that $|ab|_\infty$ is a shortest edge in Y_4^∞.

For the inductive step, pick an arbitrary edge $\overrightarrow{ab} \in Y_4^\infty$, and assume the claim of the lemma holds for each edge in Y_4^∞ strictly shorter than $|ab|_\infty$. (Recall that all measurements use the L_∞ norm.) As in the base case, rotate the point set $\mathcal{P}$ so that ab lies in the cone $C_{4k,1}(a)$ and $|ab|_\infty = |x_b - x_a|$. By Lem. 2, $\overrightarrow{ab} \in Y_{4k}^\infty$. We inductively construct a short path $pp(a,b)$ in YY_{4k}^∞ between a and b. If $\overrightarrow{ab} \in YY_{4k}^\infty$, then $pp(a,b) = \{ab\}$, and the lemma holds. So assume that $\overrightarrow{ab} \notin YY_{4k}^\infty$. Then there is $\overrightarrow{cb} \in YY_{4k}^\infty$, with $|cb|_\infty \leq |ab|_\infty$, selected in the second Yao step for b. In this case we seek a short path $p(a,c)$ between a and c in Y_4^∞, that will enable us to use the inductive hypothesis.

Case 1. First consider the simplest scenario in which $|ab|_\infty = |cb|_\infty$, meaning that ac is a vertical line segment. If c lies below a, then $\overrightarrow{ab} \in YY_{4k}^\infty$ by the way ties are broken. This is a contradiction to our assumption that $\overrightarrow{ab} \notin YY_{4k}^\infty$, so c must lie above a. This situation is depicted in Fig. 2b. Note that a lies in the cone $C_{4k,3k+1}(c)$ apexed at c, which by definition contains the vertical boundary line. The plane property of Y_4^∞ (Thm. 1) implies that $\Pi_{[Y_4]}(c,a)$ does not cross $ab \in Y_4^\infty$. (Observe that $R(c,a)$ is a degenerate rectangle reduced to the line segment ac.) Furthermore, $\Pi_{[Y_4]}(c,a)$ contains no nodes strictly interior to $\triangle abc \subset R(a,b)$. This is because $ab \in Y_4^\infty$, and consequently no points from $\mathcal{P}$ may lie interior to $R(a,b)$. It follows that $\Pi_{[Y_4]}(c,a)$ is a path with endnodes c and a consisting of vertical edges only. Let $p(a,c) = \Pi_{[Y_4]}(c,a)$, and let o be the top left corner of $R(a,b)$. Then

$$|p(a,c)|_\infty = |ac| \leq |ao| < |ab|_\infty \tan\theta.$$

This latter inequality follows from the fact that $\tan\theta > \tan(\angle abo) = |ao|/|ab|_\infty$. Because $\theta \leq \pi/6$, $\tan\theta < 1$ and therefore $|p(a,c)|_\infty < |ab|_\infty$. This further implies that each edge on $p(a,c)$ is shorter than $|ab|_\infty$, so we can apply the inductive hypothesis to prove the existence of a path $pp(a,c)$ in YY_{4k}^∞ of length $|pp(a,c)|_\infty < t|ab|_\infty \tan\theta$. This path, concatenated with cb, yields a path $pp(a,b)$ of length

$$\begin{aligned} |pp(a,b)|_\infty &< t|ab|_\infty \tan\theta + |cb|_\infty \\ &= t|ab|_\infty + (t\tan\theta - t + 1)|ab|_\infty. \end{aligned}$$

Here we used the fact that $|cb|_\infty = |ab|_\infty$. Note that the last term in the inequality above is non-positive for any $t \geq 1/(1-\tan\theta)$, so the lemma holds for this case.

Case 2. Consider now the case in which ac is not vertical. Then c must lie strictly to the right of a, because $|cb|_\infty < |ab|_\infty = |x_b - x_a|$. As before, let $R = R(a,c)$. In this case, R is a non-degenerate rectangle. We establish two properties:

(P1) No edge on $\Pi_{[Y_4]}(a,c)$ intersects the right side of R in a point other than c.

(P2) No edge on $\Pi_{[Y_4]}(c,a)$ intersects the upper side of R in a point other than a.

Due to space constraints, we skip the proof of these two properties, and discuss only their implications in our main proof. In particular, we show that these properties lead to the existence of a path $p(a,c)$ in Y_4^∞, which is no longer than $|ab|_\infty$. This will enable us to use the inductive hypothesis on each edge of $p(a,c)$. If $\Pi_{[Y_4]}(a,c)$ ends at node c, set $p(a,c) = \Pi_{[Y_4]}(a,c)$. Otherwise, if $\Pi_{[Y_4]}(c,a)$ ends at node a, then set $p(a,c) = \Pi_{[Y_4]}(c,a)$. Otherwise, properties (P1) and (P2), along with the plane property of Y_4^∞ (Thm. 1), imply that $\Pi_{[Y_4]}(a,c)$ and $\Pi_{[Y_4]}(c,a)$ must meet in a common node $d \in R(a,c)$. In this case, we set $p(a,c)$ to be the subpath of $\Pi_{[Y_4]}(a,c)$ that extends from a to d, concatenated with the subpath of $\Pi_{[Y_4]}(c,a)$ that extends from c to d (see Fig. 3b.)

In either of these cases, the path $p(a,c)$ is monotone and lies entirely inside $R(a,c)$. By definition, each edge $xy \in p(a,c)$ is no longer than the longer side of $R(x,y)$ (under the L_∞ norm). Summing up these inequalities for all edges on $p(a,c)$,

we obtain that $|p(a,c)|_\infty$ is smaller than half the perimeter of $R(a,c)$.

Let ai be the upper side of $R(a,c)$, aj the lower side of $R(a,b)$, and co the left side $R(c,b)$. Refer to Fig. 3a. Then

$$|p(a,c)|_\infty \leq |ai| + |ci| < |ai| + |ij| = |ab|_\infty$$

In the inequality above we use the fact that $|ci| \leq |co| < ij$, which follows immediately from the fact that $1 > \tan\theta > \tan(\angle cbo) = |co|/|ob| = |co|/|ij|$. In addition, the length of each edge on $p(a,c)$ is strictly smaller than $|ab|_\infty$, so we can use the inductive hypothesis to show that YY_{4k}^∞ contains a path $pp(a,c)$ between a and c of length $|pp(a,c)|_\infty < t(|ai| + |ic|)$. (Notice the use of the stronger bound here, necessary in further calculations.) This path, concatenated with cb (with $|cb|_\infty = |ij|$), gives us a path $pp(a,b)$ between a and b of length

$$\begin{aligned} |pp(a,b)|_\infty &\leq |ij| + t(|ai| + |ic|) \\ &= t(|ai| + |ij|) + |ij| - t|ij| + t|ic| \\ &< t|ab|_\infty + |ij|(1 - t + t\cdot\tan\theta) \end{aligned}$$

In deriving the latter inequality, we used the fact that $|ic| < |ij|\tan\theta$, which follows from $\tan\theta > \tan(\angle obc) = |oc|/|ob| = |oc|/|ij| \geq |ic|/|ij|$. Now note that, for the values of t allowed by the lemma, the term $1 - t + t\tan\theta \leq 0$ and therefore $|pp(a,b)|_\infty < t|ab|_\infty$. So the lemma holds for this case as well. □

Our main result is stated below, and follows immediately from the result of Lem. 3, combined with the result of Thm. 1.

Theorem 4 *The Yao-Yao graph YY_{4k}^∞ with parameter $k \geq 3$ is a t-spanner under the L_∞ norm, for $t = \frac{8}{1-\tan(\pi/2k)}$.*

4 Spanning ratio under the L_2 norm

It can be easily verified that in the L_∞ norm, all Euclidean distances are scaled up by a factor of at most $\sqrt{2}$. In other words, for any two nodes a and b, the following relationship holds:

$$\frac{|ab|}{\sqrt{2}} \leq |ab|_\infty \leq |ab|.$$

Recall that $|ab|$ is the length of the segment ab under the L_2 norm, and $|ab|_\infty$ is the length of the segment ab under the L_∞ norm. This relationship, combined with the result of Thm. 4, leads to the following result:

Theorem 5 *The Yao-Yao graph YY_{4k}^∞ with parameter $k \geq 3$ is a t-spanner under the L_2 norm, for $t = \frac{8\sqrt{2}}{1-\tan(\pi/2k)}$.*

5 Conclusions

Our results show that the sparse Yao graph YY_s^∞ constructed in the L_∞ metric is a spanner, for the special case when $s \geq 12$ is a multiple of 4. We believe that our method can be extended to prove that YY_s^∞ is a spanner, for any $s \geq 9$. Establishing whether the sparse Yao graph YY_s constructed in the L_2 metric is a spanner or not remains open.

References

[1] Prosenjit Bose, Mirela Damian, Karim Douïeb, Joseph O'Rourke, Ben Seamone, Michiel H. M. Smid, and Stefanie Wuhrer. Pi/2-angle Yao graphs are spanners. In *Proc. of the 21st International Symposium on Algorithms and Computation* LNCS vol. 6507(II) (2010) pp. 446–457.

[2] Prosenjit Bose, Mirela Damian, Karim Douïeb, Joseph O'Rourke, Ben Seamone, Michiel H. M. Smid, and Stefanie Wuhrer. Pi/2-angle Yao graphs are spanners. Online, `arXiv:1001.2913`, Jan. 2010.

[3] Mirela Damian and Kristin Raudonis. Yao graphs span Theta graphs. In *COCOA'10: Proc. of the 4th International Conference on Combinatorial Optimization and Applications*, December 2010. To appear.

[4] Nawar Molla. Yao spanners for wireless ad hoc networks. Technical report, M.S. Thesis, Department of Computer Science, Villanova University (December 2009).

[5] Mirela Damian, Nawar Molla, and Val Pinciu. Spanner properties of $\pi/2$-angle yao graphs. In *EuroCG'09: Proc. of the 25th European Workshop on Computational Geometry*, pages 21–24, March 2009.

[6] Matthias Grünewald, Tamás Lukovszki, Christian Schindelhauer, and Klaus Volbert. Distributed maintenance of resource efficient wireless network topologies (distinguished paper). In *Euro-Par '02: Proceedings of the 8th International Euro-Par Conference on Parallel Processing*, pages 935–946, London, UK, 2002. Springer-Verlag.

[7] Prosenjit Bose, Anil Maheshwari, Giri Narasimhan, Michiel H. M. Smid, and Norbert Zeh. Approximating geometric bottleneck shortest paths. *Computational Geometry: Theory and Applications*, 29(3):233–249, 2004.

This is an extended abstract of a presentation given at EuroCG 2012. 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 in a conference with formal proceedings and/or in a journal.

On the order-k Voronoi diagram of line segments*

Evanthia Papadopoulou† Maksym Zavershynskyi†

Abstract

We analyze the structural properties of the order-k Voronoi diagram of n disjoint line-segments. We show that order-k Voronoi regions of line-segments may be disconnected, where a single Voronoi region may disconnect to $\Omega(n)$ disjoint faces. However, the overall structural complexity of the order-k line-segment Voronoi diagram remains $O(k(n-k))$ as in the case of points.

1 Introduction

Given a set S of n geometric objects in the plane, called sites, the order-k Voronoi diagram of S is a partitioning of the plane into maximal regions, such that all points within an order-k region have the same k nearest sites. The order-1 Voronoi diagram is the *nearest-neighbor Voronoi diagram*, and the order-$(n-1)$ Voronoi diagram corresponds to the *farthest Voronoi diagram*.

For point sites, the order-k Voronoi diagram has been studied in [11, 5, 7, 1, 3]. Its structural complexity has been shown to be $O(k(n-k))$ [11]. The order-k Voronoi diagram of line segments has been largely ignored in the literature. Only the nearest-neighbor line-segment Voronoi diagram has been intensively studied, see e.g. [9, 12, 15, 10], and the farthest line-segment Voronoi diagram in [4].

In this paper, we analyze the structural properties of the order-k Voronoi diagram of segments. We show that a single order-k Voronoi region can disconnect to $\Omega(n)$ faces, in the worst case. However, the overall structural complexity remains $O(k(n-k))$ in the case of disjoint line segments, despite disconnected regions.

The main difficulty of proving the structural complexity lies in the analysis of unbounded faces. In the case of points, the analysis relies on results from k-set theory [2, 5] as well as an important symmetry property, which states that the number of unbounded faces in an order-k diagram equals the number of unbounded faces in the order-$(n-k)$ diagram. However, results from k-set theory are not directly applicable to line-segments and the symmetry property does not hold in the case of line segments. Our proof requires different kind of analysis, which is explained in this paper.

2 Preliminaries

Given is a set $S = \{s_1, s_2, \ldots, s_n\}$ of n line segments in $\mathbb{R}^2$. The distance from a point p to a line segment s is the ordinary minimum distance $d(p,s) = \min_{q \in s} d(p,q)$, where $d(p,q)$ is the Euclidean distance between points p and q. The bisector between two segments s_i, s_j is the locus of points equidistant to both, $b(s_i, s_j) = \{x \in \mathbb{R}^2 \mid d(x,s_i) = d(x,s_j)\}$.

Definition 1 *Given a set S of n segments and an integer k, $1 \leq k \leq n-1$, the order-k Voronoi diagram of S, denoted as $V_k(S)$, subdivides the plane into maximal regions such that every point within an order-k region has the same k nearest sites. Formally, the Voronoi region is $\mathcal{V}_k(H,S) = \{x \mid \forall s \in H\ \forall t \in S \setminus H\ d(x,s) \leq d(x,t)\}$, where H is a set of k segments of S.*

The distance from point x to a set of k segments H is measured as the maximum distance $d_f(x,H) = \max_{s \in H} d(x,s)$. An order-$k$ Voronoi region $\mathcal{V}_k(H,S)$ can be defined equivalently as the locus of points closer to H, according to $d_f(x,H)$, than to any other subset of S of size k. The connected components of $\mathcal{V}_k(H,S)$ are called faces. If a region consists of more than one face, the region is called disconnected.

For simplicity, in this paper we assume that segments are in general position. That is, there are no four segments touching the same circle, and there are no three collinear endpoints. We consider disjoint line segments, i.e., segments are not allowed to touch or intersect.

The following lemma is a simple generalization of [4] for $1 \leq k \leq n-1$.

Lemma 1 *Consider a face F of region $\mathcal{V}_k(H,S)$. F is unbounded (in the direction r) iff there exists an open halfplane (normal to r) which intersects all segments in H but no segment in $S \setminus H$.*

Corollary 2 *There is an unbounded Voronoi edge separating regions $\mathcal{V}_k(H \cup \{s_1\}, S)$ and $\mathcal{V}_k(H \cup \{s_2\}, S)$ iff a line through the endpoints of s_1 and s_2 induces an*

*Supported in part by the Swiss National Science Foundation (SNF) grant 200021-127137. Also by SNF grant 20GG21-134355 within the collaborative research project EuroGIGA/VORONOI of the European Science Foundation.

†Faculty of Informatics, Università della Svizzera italiana, Lugano, Switzerland,
{evanthia.papadopoulou, maksym.zavershynskyi}@usi.ch

open halfplane $r(s_1, s_2)$ such that $r(s_1, s_2)$ intersects all segments in H but no segment in $S \setminus H$.

3 Disconnected regions

The order-k line segment Voronoi diagram may have disconnected regions, unlike its counterpart of points, see e.g., Fig. 1. This phenomenon was first studied in [4] for the farthest Voronoi diagram of line segments. For the farthest Voronoi diagram the number of disconnected faces of a single region was shown to be $\Theta(n)$, in the worst case.

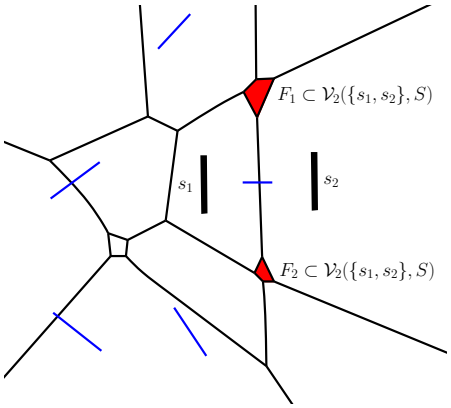


Figure 1: An order-2 Voronoi diagram, with two disconnected faces, induced by the same pair of sites.

Lemma 3 *An order-k region of $V_k(S)$ can have $\Omega(n)$ disconnected faces, in the worst case.*

Proof.
Claim 1: An order-k Voronoi region of $V_k(S)$ can have $\Omega(n-k)$ disconnected bounded faces, in the worst case.
We describe an example where an order-k Voronoi region is disconnected in $\Omega(n-k)$ faces. Consider k almost parallel long segments H. These segments induce a region $\mathcal{V}_k(H, S)$. Consider a minimum disk, that intersects all segments of H, and moves along their length. We place the remaining $n-k$ segments of $S \setminus H$ in such way, that they create obstacles for the disk. While the disk moves along the tree of $V_{k-1}(H)$ (the farthest Voronoi diagram of H) it intersects the segments of $S \setminus H$, and one by one creates $\Omega(n-k)$ disconnectivities (see Fig. 2).

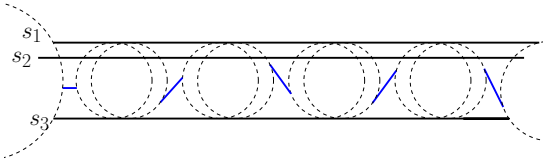


Figure 2: While the circle moves it encounters 5 obstacles, which induce $\Omega(n-k)$ disconnectivities of the region $\mathcal{V}_3(H, S)$.

Claim 2: An order-k Voronoi region of $V_k(S)$ can have $\Omega(k)$ disconnected unbounded faces, in the worst case.
We follow the example in [4] for $k = n-1$. Consider $n-k$ segments $S \setminus H$ degenerated into points placed close to each other. The remaining k non-degenerate segments H are organized in a cyclic fashion around them (see Fig. 3). Consider a directed line g through one of the degenerate segments s'. Rotate g around s' and consider the open halfplane to the left of it. During the rotation, the positions in which the halfplane intersects all k segments, alternate with the positions in which it does not. The positions in which the halfplane touches an endpoint of non-degenerate segment, correspond to unbounded Voronoi edges. Each pair of consecutive unbounded edges bounds a disjoint unbounded face, i.e. the bisectors $b(s', s_i)$ and $b(s', s_{i+1})$ bound a disjoint unbounded face of $\mathcal{V}_k(H, S)$, where s_i and s_{i+1} - are consecutive segments of H.

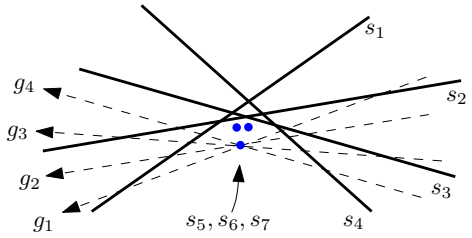


Figure 3: $\mathcal{V}_4(\{s_1, s_2, s_3, s_4\}, S)$ has $k = 4$ disconnected unbounded faces.

Note that for small k, $0 < k \leq n/2$, $\Omega(n-k) = \Omega(n)$, while for large k, $n/2 \leq k \leq n-1$, to $\Omega(k) = \Omega(n)$. □

Lemma 4 *An order-k region $\mathcal{V}_k(H, S)$ has $O(k)$ unbounded disconnected faces.*

Proof. We prove the following claim: The endpoint p of a segment $s \in H$ may induce at most two unbounded Voronoi edges bordering $\mathcal{V}_k(H, S)$.

Consider two such unbounded Voronoi edges. By Corollary 2 there are open halfplanes $r(s, t_1)$, $r(s, t_2)$, for $s \in H$ and $t_1, t_2 \in S \setminus H$, that intersect all segments in H but no segments in $S \setminus H$ (see Fig. 4). Thus, any halfplane $r(s, t_3)$, $t_3 \in S \setminus H$ must intersect either t_1 or t_2 (see Fig. 4).

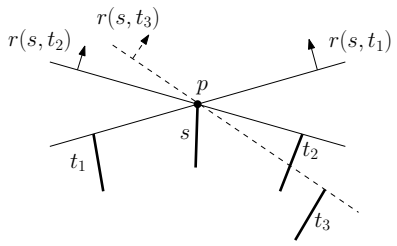


Figure 4: Every endpoint of a segment $s \in H$ can induce at most 2 halfplanes. □

4 Structural complexity

Euler's formula and standard properties of Voronoi diagrams imply that the structural complexity of the order-k line-segment Voronoi diagram is proportional to the number of its faces.

4.1 Number of faces

The following lemma generalizes the result of Lee [11] for the case of disjoint line segments.

Lemma 5 *The number of Voronoi faces in order-k Voronoi diagram of segments is:*

$$F_k = 2kn - k^2 - n + 1 - \sum_{i=1}^{k-1} S_i \qquad (1)$$

where F_k is the number of faces of the order-k Voronoi diagram of segments, and S_i is the number of unbounded faces of the order-i Voronoi diagram of segments.

Proof. The proof can be derived following [11] and using the properties below. There are three crucial properties of order-k Voronoi diagrams that remain valid in the case of segments. We give them without a proof.

1. Consider a face F of the region $\mathcal{V}_k(H, S)$. The edges of $V_{k-1}(S)$ enclosed in F form a single connected component of $V_{k-1}(H)$, where $V_{k-1}(H)$ is the farthest Voronoi diagram of H.
2. The farthest Voronoi diagram of line segments is a tree [4].
3. Every Voronoi vertex of $V_k(S)$ is either *new* or *old*. A Voronoi vertex is called *new* if it is incident to the regions induced by the sets of segments $H \cup \{a\}$, $H \cup \{b\}$ and $H \cup \{c\}$, of size k each. A Voronoi vertex called *old* if it is incident to the regions induced by the sets of segments $H' \cup \{a, b\}$, $H' \cup \{b, c\}$ and $H' \cup \{c, a\}$, of size k each. A *new* Voronoi vertex in $V_k(S)$ is an *old* Voronoi vertex in $V_{k+1}(S)$. In addition, a *new* Voronoi vertex in $V_k(S)$ does not exist in $V_{k-1}(S)$ and an *old* Voronoi vertex of $V_k(S)$ does not exist in $V_{k+1}(S)$ (assuming general position).

Further details are omitted in this abstract. □

Lemma 5 directly implies that $F_k = O(k(n-k))$, for $1 \le k \le n/2$. But for $n/2 \le k \le n-1$, $2kn - k^2 - n + 1 = \Omega(nk)$. To derive a good upper bound on $n/2 \le k \le n-1$, we need to analyze $\sum_{i=1}^{k-1} S_i$.

For points, using [2] and a symmetry property stating that $S_i = S_{n-i}$, it can be shown that $\sum_{i=1}^{k-1} S_i \ge nk$, $n/2 \le k \le n-1$ [5]. Unfortunately, this approach does not generalize to line segments, because $S_k \ne S_{n-k}$, and no k-set theory results are available for line-segments, to the best of our knowledge.

4.2 Number of unbounded faces

It is not hard to see that $\sum_{i=1}^{n-1} S_i = n(n-1)$. This is because every pair of segments defines exactly two open halfplanes, where each halfplane induces an unbounded Voronoi edge for some order k. Combining it with (1) we obtain:

$$F_k = 1 - (n-k)^2 + \sum_{i=k}^{n-1} S_i \qquad (2)$$

We can restate our goal as finding an upper bound for $\sum_{i=k}^{n-1} S_i$.

Lemma 6 *For a given set of n disjoint segments, $\sum_{i=k}^{n-1} S_i$ is $O(k(n-k))$, where $n/2 \le k \le n-1$.*

Proof. Following [4], we use the well-known point-line duality transformation T, which maps a point $p = (a, b)$ in the primal plane to a line $T(p) : y = ax - b$ in the dual plane, and vice versa. We call the set of points below both lines $T(p)$ and $T(q)$ the *wedge* of s. Consider a line r and a segment $s = (p, q)$. The segment s is below line r iff point $T(r)$ is strictly below lines $T(p)$ and $T(q)$ [4].

Consider the arrangement W of the wedges w_i, $i = 1, \ldots, n$, defined by the segments of $S = \{s_1, \ldots, s_n\}$. For our analysis we need the notions of k-level and ($\le k$)-level (see e.g. [14]). The k-level of W is a set of edges such that every point on it is above k wedges. Note, that the k-level shares its vertices with the $(k-1)$-level and the $(k+1)$-level. The ($\le k$)-level of W is a set of edges such that every point on it is above at most k wedges. The complexity of the k-level and the ($\le k$)-level is the number of their edges. We denote the maximum complexity of the k-level and the ($\le k$)-level of n wedges as $g_k(n)$ and $g_{\le k}(n)$, respectively.

Claim: The number of unbounded Voronoi edges of $V_{k+1}(S)$, unbounded in direction $\phi \in [0, \pi]$, is exactly the number of vertices of the k-level of W that are not in the $(k-1)$-level of W.

Proof of claim: Consider a vertex p (see Fig. 5) of the k-level. Let w_i and w_j be the wedges, that intersect at p, and let $W_p = \{w_1, w_2, \ldots, w_k\}$ be the set of wedges strictly below p. Point p corresponds to a line $T(p)$ in the primal plane, such that the halfplane above it, is the halfplane of the unbounded edge, that separates regions $\mathcal{V}_k(S_p \cup \{s_i\}, S)$ and $\mathcal{V}_k(S_p \cup \{s_j\}, S)$, where S_p is the set of edges that define the edges W_p. Therefore $S_{k+1} = O(g_k(n))$.

Consider the arrangement W^* obtained by flipping W upside-down. The maximum complexities of the k-level and the ($\le k$)-level of W^* are $g^*_k(n)$ and $g^*_{\le k}(n)$, respectively. Clearly $g_k(n) = g^*_{n-k-1}(n)$. Thus, $S_{k+1} = O(g^*_{n-k-1}(n))$, and

$$\sum_{i=k}^{n-1} S_i = O(g^*_{\le n-k}(n)) \qquad (3)$$

Since the arrangement of wedges is a special case of the arrangement of Jordan curves, we can use the formula[1] from [14] to bound the complexity of the ($\leq k$)-level:

$$g^*_{\leq k}(n) = O\left((k+1)^2 g^*_0\left(\left\lfloor \frac{n}{k+1} \right\rfloor\right)\right) \qquad (4)$$

The complexity of the lower envelope of such wedges is $g^*_0(x) = O(x)$ [6, 4], therefore $g^*_{\leq k}(n) = O(n(k+1))$. Substituting it into formula (3) we obtain $\sum_{i=k}^{n-1} S_i = O(n(n-k))$. Since $n/2 \leq k \leq n-1$ this also means $\sum_{i=k}^{n-1} S_i = O(k(n-k))$.

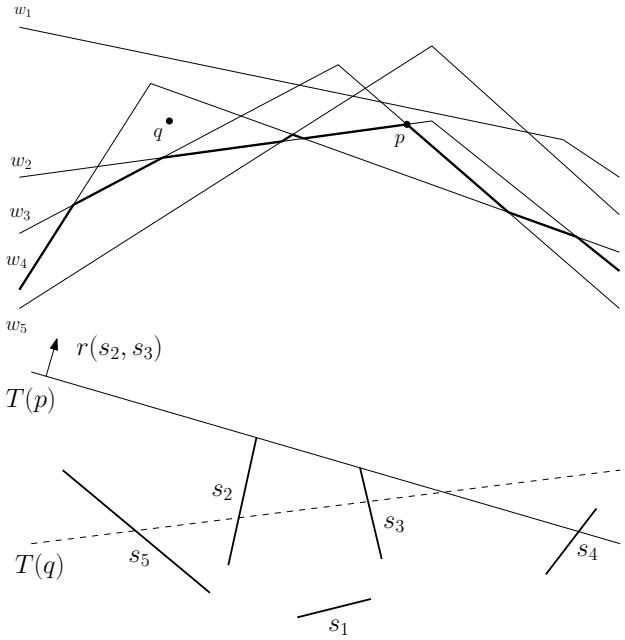


Figure 5: (a) In the dual plane point p belongs to the 1-level of the arrangement W. (b) In the primal plane the halfplane $r(s_2, s_3)$ above $T(p)$ defines the unbounded Voronoi edge that separates $\mathcal{V}_2(\{s_2, s_4\}, S)$ and $\mathcal{V}_2(\{s_3, s_4\}, S)$. □

Combining the above lemma with (2) we obtain the following theorem.

Theorem 7 *The number of Voronoi faces in order-k Voronoi diagram of n segments is* $F_k = O(k(n-k))$.

5 Conclusion

We presented an analysis of the structure of the order-k Voronoi diagram of disjoint line segments. We have proved that the structural complexity of order-k Voronoi diagram of n disjoint line segments is $O(k(n-k))$ as in the case of points. The structural complexity remains the same despite the fact that a single region can disconnect to $\Omega(n)$ faces. The order-k Voronoi diagram of line segments can be constructed in $O(k^2 n \log n)$ time by adapting any iterative approach to compute higher order Voronoi diagrams. We are currently considering more efficient algorithmic techniques.

[1] The exact formula is $g^*_{\leq k}(n) = O\left((k+1)^2 \lambda_s\left(\left\lfloor \frac{n}{k+1} \right\rfloor\right)\right)$ [14]. Its proof first shows $g^*_{\leq k}(n) = O\left((k+1)^2 g^*_0\left(\left\lfloor \frac{n}{k+1} \right\rfloor\right)\right)$ and then substitutes $g^*_0(x) = O(\lambda_s(x))$, where $\lambda_s(x)$ - is the maximum length of (x, s) Davenport-Schinzel sequence [14].

References

[1] P. Agarwal, M. de Berg, J. Matousek, and O. Schwarzkopf. *Constructing levels in arrangements and higher order Voronoi diagrams.* SIAM J. Comput. 27(3): 654-667 (1998)

[2] N. Alon and E. Györi. *The number of small semispaces of a finite set of points in the plane.* J. Comb. Theory, Ser. A 41(1): 154-157 (1986)

[3] F. Aurenhammer *Voronoi Diagrams - A Survey of a Fundamental Geometric Data Structure* ACM Comput. Surv. 23(3): 345-405 (1991)

[4] F. Aurenhammer, R. Drysdale, and H. Krasser. *Farthest line segment Voronoi diagrams.* Inf. Process. Lett. 100(6): 220-225 (2006)

[5] H. Edelsbrunner. *Algorithms in combinatorial geometry.* EATCS monographs on theoretical computer science. Springer, 1987.

[6] H. Edelsbrunner, H. A. Maurer, F. P. Preparata, A. L. Rosenberg, E. Welzl and D. Wood. *Stabbing Line Segments.* BIT 22(3): 274-281 (1982)

[7] H. Edelsbrunner and R. Seidel. *Voronoi diagrams and arrangements.* Discrete & Computational Geometry 1: 25-44 (1986)

[8] S. Fortune. *A Sweepline Algorithm for Voronoi Diagrams.* Algorithmica 2: 153-174 (1987)

[9] M. I. Karavelas. *A robust and efficient implementation for the segment Voronoi diagram.* In Proc. 1st Int. Symp. on Voronoi Diagrams in Science and Engineering, Tokyo: 51-62 (2004)

[10] D. G. Kirkpatrick. *Efficient Computation of Continuous Skeletons* FOCS 1979: 18-27

[11] D. T. Lee. *On k-Nearest Neighbor Voronoi Diagrams in the Plane.* IEEE Trans. Computers 31(6): 478-487 (1982)

[12] D. T. Lee, R. L. S. Drysdale. *Generalization of Voronoi Diagrams in the Plane.* SIAM J. Comput. 10(1): 73-87 (1981)

[13] M. I. Shamos and D. Hoey. *Closest-point problems.* In Proc. 16th IEEE Symp. on Foundations of Comput. Sci.: 151-162 (1975)

[14] M. Sharir and P. Agarwal. *Davenport-Schinzel Sequences and their Geometric Applications.* Cambridge University Press, 1995.

[15] C.-K. Yap. *An $O(n \log n)$ Algorithm for the Voronoi Diagram of a Set of Simple Curve Segments.* Discrete & Computational Geometry 2: 365-393 (1987)

On the Farthest Line-Segment Voronoi Diagram*

Evanthia Papadopoulou† Sandeep Kumar Dey†

Abstract

The farthest line-segment Voronoi diagram shows properties surprisingly different than the farthest point Voronoi diagram: Voronoi regions may be disconnected and they are not characterized by convex-hull properties. In this paper we introduce the *farthest line-segment hull*, a cyclic structure that relates to the farthest line-segment Voronoi diagram similarly to the way an ordinary convex hull relates to the farthest-point Voronoi diagram and provide $O(n \log n)$-time algorithms for its construction. Using the farthest line-segment hull, we derive a more tight bound on the (linear) size of the farthest line-segment Voronoi diagram. We also illustrate properties of the L_∞ farthest line-segment Voronoi diagram, which finds applications in VLSI Design Automation.

1 Introduction

Let S be a set of n simple geometric objects in the plane, such as points or line segments, called sites. The *farthest-site Voronoi diagram* of S is a subdivision of the plane into regions such that the region of a site s is the locus of points farther away from s than from any other site. Surprisingly, the farthest-line-segment Voronoi diagram illustrates different properties than its counterpart for points [1]. For example, Voronoi regions are not characterized by convex-hull properties and they may be disconnected; a Voronoi region may consist of $\Theta(n)$ disconnected faces. Nevertheless, the graph structure of the diagram remains a tree and its structural complexity is $O(n)$. An abstract framework on the farthest-site Voronoi diagram (which does not include the case of intersecting line-segments) was given in [5]. Related is the farthest-polygon Voronoi diagram, recently addressed in [3].

In this paper we further study the structural properties of the farthest line-segment Voronoi diagram. We introduce the *farthest line-segment hull*, a cyclic structure that is related to the farthest line-segment Voronoi diagram similarly to the way an ordinary convex hull is related to the farthest-point Voronoi diagram. Using the farthest line-segment hull, we improve the (linear) upper bound on the structural complexity of the diagram in [1] by a constant factor. We provide $O(n \log n)$-time algorithms for the construction of the farthest line-segment hull by adapting standard approaches for the construction of an ordinary convex hull such as divide and conquer, Graham's scan, etc. We also characterize the farthest line-segment hull and the corresponding Voronoi diagram in the simpler L_∞, L_1 metrics. Once the farthest line-segment hull is available, the farthest line-segment Voronoi diagram can be constructed in additional $O(h \log h)$ time, by the simple algorithm given in [1], where h is the size of the farthest line-segment hull ($h \in O(n)$). In the L_∞ or L_1 metrics the construction simplifies to additional $O(h)$-time.

The farthest line-segment Voronoi diagram finds applications in computing the smallest disk that overlaps all given line-segments. It is necessary in defining and computing the *Hausdorff Voronoi diagram* of clusters of line segments, which finds several applications in VLSI design automation, see e.g., [6].

2 Preliminaries and Definitions

Let $S = \{s_1, \ldots, s_n\}$ be a set of n arbitrary line-segments in the plane. Line-segments may intersect or touch at single points. The distance between a point q and a line-segment s_i is $d(q, s_i) = \min\{d(q, y), \forall y \in s_i\}$, where $d(q, y)$ denotes the ordinary distance between two points, q, y, in the L_p metric, $p, 1 \leq p \leq \infty$. The farthest Voronoi region of a line-segment s_i is

$$freg(s_i) = \{x \in \mathbb{R}^2 \mid d(x, s_i) \geq d(x, s_j), 1 \leq j \leq n\}$$

The collection of all farthest Voronoi regions, together with their bounding edges and vertices, constitute the *farthest line-segment Voronoi diagram* of S, denoted as *FVD(S)* (see Fig. 1). Any maximally connected subset of a region in *FVD(S)* is called a *face*. Any Voronoi edge bounding two neighboring regions, $freg(s_i)$ and $freg(s_j)$, is portion of bisector $b(s_i, s_j)$, i.e., the locus of points equidistant from s_i and s_j. For line-segments in general position that are non-intersecting, $b(s_i, s_j)$ is an unbounded curve that consists of a constant number of simple pieces as induced by elementary bisectors between the endpoints and open portions of s_i, s_j. For more information on line-segment bisectors, see e.g., [4, 8] and references therein.

*Supported in part by the Swiss National Science Foundation (SNF), grant 200021-127137. Also by SNF grant 20GG21-134355 within the collaborative research project EuroGIGA/VORONOI of the European Science Foundation.

†Faculty of Informatics, University of Lugano (Università della Svizzera italiana), Lugano, Switzerland,
{evanthia.papadopoulou, deys}@usi.ch

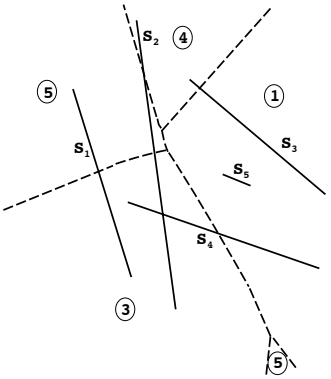


Figure 1: A farthest line-segment Voronoi diagram.

3 The farthest line-segment hull

Definition 1 *A line l through the endpoint p of a line-segment s, $s \in S$, is called a* supporting line *of S if and only if an open halfplane induced by l, denoted $H(l)$, intersects all segments in S, except s (and possibly except additional segments incident to p). Point p is said to* admit a supporting line. *The unit normal of l, pointing away from $H(l)$, is called the* unit vector *of l, denoted $\nu(l)$. A line-segment s, $s \in S$, such that the line l through s is a supporting line of S and $H(l)$ intersects all segments in $S \setminus \{s\}$, is called a* hull segment*; s is said to* admit a supporting line *and the unit vector of l is denoted as $\nu(s)$.*

For a supporting line l, through line-segment s, that intersects all segments in S, both open halfplanes of l, denoted $H(s)$ and $H'(s)$, intersect all segments in $S \setminus \{s\}$. In this case s is said to admit two supporting lines and it results in two different hull segments.

Definition 2 *The line segment $\overline{pq}$ joining the endpoints p, q, of two line segments $s_i, s_j \in S$ is called a* supporting segment *of S if and only if an open halfplane induced by the line l through $\overline{pq}$, denoted $H(\overline{pq})$, intersects all segments in S, except s_i, s_j (and possibly except additional segments incident to p, q). The unit normal of $\overline{pq}$ pointing away from $H(\overline{pq})$ is called the* unit vector, *$\nu(\overline{pq})$.*

Definition 3 *The closed polygonal curve obtained by the sequence of hull segments in S (line-segments in S that admit a supporting line), ordered according to the angular order of their unit vectors, interleaved by maximal chains of supporting segments, is called the farthest line-segment hull of S (for brevity, the farthest hull), denoted as f-hull(S).*

The vertices of the farthest hull are exactly the endpoints of S that admit a supporting line. The edges are of two types: supporting segments and hull segments. If line-segments degenerate to points, the farthest hull corresponds exactly to the convex hull of S.

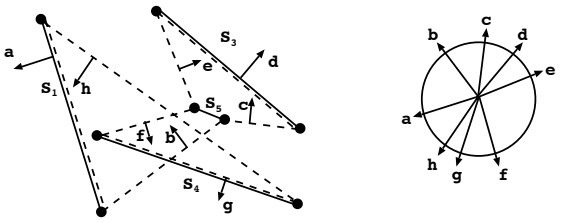


Figure 2: The farthest line-segment hull of Fig. 1

By definition of a supporting segment, any maximal chain of supporting segments between two consecutive hull segments must be convex. Fig. 2 illustrates the farthest hull of Fig. 1. Clearly, the unit vector of any supporting line of the farthest hull is unique.

Consider the circular list of the unit vectors of all edges of the farthest hull ordered angularly around the origin of the unit circle K_o. They induce a mapping of the farthest hull onto the circumference of K_o, such that every edge e is mapped to a point on the circumference of K_o as obtained by the unit vector $\nu(e)$, and every vertex is mapped to arcs as delimited by the unit vectors of the incident edges. This mapping is called the *Gaussian map* of S, for short *Gmap(S)*. (For more information on the Gmap see e.g., [2, 7]). The same notation, $\nu(e)$, is used to denote both the unit vector and the corresponding point on the circumference of K_o. *Gmap(S)* can be viewed as a cyclic sequence of arcs on the circumference of K_o, where each arc corresponds to a vertex of the farthest hull.

Lemma 1 *Consider any two consecutive unit vectors ν_i, ν_{i+1}, along Gmap(S) corresponding to edges e, e' of the farthest hull such that $\nu_i = \nu(e)$ and $\nu_{i+1} = \nu(e')$. Edges e, e' must be incident upon a common vertex v of f-hull(S), and any point along the arc on K_o from ν_i to ν_{i+1}, corresponds to the unit vector of a supporting line through v.*

Corollary 2 *FVD(S) has exactly one unbounded bisector for every supporting segment s of f-hull(S), which is unbounded in the direction opposite to $\nu(s)$. Unbounded bisectors in FVD(S) are cyclically ordered following exactly the cyclic ordering of the Gmap.*

By Corollary 2, *FVD(S)* has exactly one face for every pair of consecutive supporting segments on *f-hull(S)*. By Lemma 1, any two consecutive supporting segments are incident to the same line-segment s (either to the same endpoint or to one endpoint each), which must be the owner of the face delimited by the corresponding pair of consecutive unbounded bisectors. This face is unbounded in any direction induced by the unit vectors of points along the corresponding arc of the Gmap.

3.1 Properties of the farthest line-segment hull

Considering the Gmap as a cyclic sequence of arcs on the circumference of K_o, it forms a Davenport-Schinzel sequence of length 3 (see also [1]). Let the *upper* (resp. *lower) Gmap* denote the portion of the *Gmap* above (resp. below) the horizontal diameter of K_o. Equivalently for the *upper* and *lower f-hull*. In the following we always assume a traversal of the upper Gmap from left to right and we characterize the right and left endpoints of a segment s as a *start-vertex* and an *end-vertex* respectively.

Let an *interval* $[a_i, a_{i+1}]$ denote the portion of the upper Gmap between two consecutive (but not adjacent) occurrences of arcs for the same segment $s_a = (a', a)$, where a, a' denote the start-vertex and end-vertex of s_a respectively. Interval $[a_i, a_{i+1}]$ is assumed to be *non-trivial* i.e., it contains in its interior at least one arc other than a, a'.

Lemma 3 *Let $[a_i, a_{i+1}]$ be a (non-trivial)* interval *of segment $s_a = (a', a)$ on upper Gmap(S). We have the following properties: 1. On the upper Gmap, all occurrences of start-vertex a must precede all occurrences of end-vertex a'. 2. The vertex following a_i (resp. preceding a_{i+1}) in $[a_i, a_{i+1}]$ must be a start-vertex (resp. end-vertex). 3. If a_i is a start-vertex, i.e., $a_i = a$, (resp. a_{i+1} is end-vertex a'), no other start-vertex (resp. end-vertex) b in the interval $[a_i, a_{i+1}]$ can appear before a_i or past a_{i+1} on the upper Gmap, and no end-vertex (resp. start-vertex) e in $[a_i, a_{i+1}]$ can appear before a_i (resp. past a_{i+1}) on the upper Gmap.*

Proof. (Sketch). The proof is easier to derive using the duality transformation given in [1] that transforms a line segment to a wedge below two intersecting lines and reduces the problem of identifying the upper Gmap into the problem of computing the union of those wedges (see Fig. 3 of [1]). Here, we only point out the equivalence between the arcs of the upper Gmap and edges along the union of wedges in [1]. The proof is not hard to see. □

Lemma 4 *The re-appearance along the upper Gmap of any endpoint of a segment s_a can be charged to a unique vertex u of the upper f-hull such that no other re-appearance of a segment endpoint on the upper Gmap can be charged to u.*

Proof. (Sketch). For an interval $[a_i, a_{i+1}]$ such that $a_i = a$ is a start-vertex, let u be the vertex following a_i in $[a_i, a_{i+1}]$, which must be a start-vertex by Lemma 3; the appearance of a_{i+1} is charged to u. Similarly, for an interval $[a_i, a_{i+1}]$ such that $a_{i+1} = a'$ is an end-vertex, let u be the vertex preceding a_{i+1}, which must be an end-vertex; the appearance of a_i is charged to u. Using the properties of Lemma 3 it is not hard to see that no other interval $[c_j, c_{j+1}]$ of any segment s_c can be charged to u. □

Theorem 5 *The total number of faces of the farthest line-segment Voronoi diagram of a set S of n arbitrary line segments is at most $6n - 2$.*

Proof. (Sketch). We count the number of *maximal arcs* along the upper Gmap There are two types of maximal arcs: *composite arcs*, corresponding to hull segments, which consist of the segment unit vector and the two incident arcs of the segment endpoints, and *single-vertex* arcs, corresponding to single vertices along convex chains of the hull, which are single arcs bounded by the unit vectors of the two incident supporting segments. Consider the sequence of all occurrences of a single segment $s_a = (a', a)$ on the upper Gmap. It is a sequence of the form $\ldots a \ldots a a' \ldots a' \ldots$ or $\ldots a \ldots a \ldots a' \ldots a' \ldots$. The number of its maximal arcs is exactly one plus the number of (non-trivial) intervals. Thus, summing over all segments, the total number of maximal arcs on the upper Gmap is at most n plus the total number of vertices that may get charged for the reappearance of a single vertex arc, which is $2n$ by Lemma 4. Thus, in total $3n$. Similarly for the lower Gmap. □

The above theorem improves the $8n + 4$ upper bound on the number of faces of the farthest line-segment Voronoi diagram given in [1]. A known lower bound is $4n - 4$ [1]. For disjoint segments the corresponding upper bound is $2n - 2$ (see [3]) as the Gmap corresponds to a Davenport-Schinzel sequence of length 2.

4 Algorithms for the farthest line-segment hull

Most approaches to compute an ordinary convex hull can be adapted to compute the farthest line-segment hull within the same time complexity. Lemma 3 gives insight for a Graham's scan type of approach (to be given in a more complete version of this paper). A standard divide and conquer approach can be designed using the properties listed below. A two-pass divide and conquer construction in the dual space (under the standard point-line duality) is given in [1].

Let L, R be two subsets partitioning S. Given a unit vector $\nu(e)$ in *Gmap(L)*, let its *neighboring supporting vertex* in R be the vertex m in *Gmap(R)* such that $\nu(e)$ lies along an arc of m. A unit vector in *Gmap(L)* is called *valid* iff its corresponding supporting line remains a supporting line in *Gmap($L \cup R$)*. (Respectively for *Gmap(R)*).

Lemma 6 *Edge e of f-hull(L) remains valid iff the neighboring supporting vertex q of $\nu(e)$ in R lies in $H(e)$.*

Lemma 7 *Vertex $m \in$ f-hull(L) remains valid iff either an incident farthest hull edge remains valid, or m*

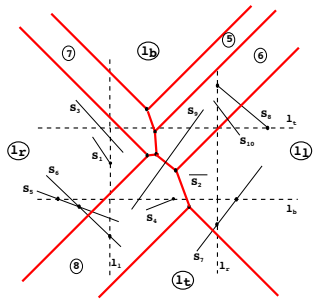


Figure 3: L_∞ *FVD(S).*

is the neighboring supporting vertex in L of an invalid unit vector in f-hull(R).

Gmap(S) can be obtained by merging *Gmap(L)* and *Gmap(R)* as follows: 1. Mark all unit vectors and vertex arcs along *Gmap(L)* and *Gmap(R)* as valid or invalid. 2. Merge *Gmap(L)* and *Gmap(R)* into *Gmap(L)*∪*Gmap(R)* following the angular order of unit vectors. 3. For any two consecutive valid vectors, one in $Gmap(L)$ and one in $Gmap(R)$, insert a unit vector as defined by the supporting segment joining the vertices of the two incident arcs. 4. For any valid vertex m of $Gmap(L)$ between two consecutive valid vectors of $Gmap(R)$, insert new unit vectors for the corresponding supporting segments incident to m. 5. Delete all invalid vectors.

5 The L_∞ farthest line segment Voronoi diagram

In L_∞, the farthest line-segment Voronoi diagram does not maintain the $O(1)$ structural complexity as its counterpart for points (see Fig. 3). All its faces are unbounded in one of the eight possible directions as implied by rays of slope $\pm 1, 0, \infty$. The L_∞ farthest line-segment hull simplifies as shown in Fig. 4. Let four bounding lines be defined as follows: let l_t (resp. l_b) be the horizontal line passing through the topmost lower-endpoint (resp. the bottommost upper-endpoint) of all segments in S and let l_l (resp. l_r) be the vertical line passing through the leftmost right-endpoint (resp. the rightmost left-endpoint) of all segments in S. They partition the plane into four quadrants, labeled $1-4$ in counterclockwise order as shown in Fig. 4. Let $E_i, i = 1, 2$, (resp. $E_j, j = 3, 4$) denote the upper (resp. lower) envelope of the line segments straddling quadrant i (resp. j). (If no segments straddle quadrant i, let E_i be the corner point O_i of quadrant i).

Definition 4 *The L_∞ farthest line-segment hull of S is a closed polygonal curve as derived by a counterclockwise traversal of $E_1, l_{t'}, E_2, l_{l'}, E_3, l_{b'}, E_4, l_{r'}$, where $l_{i'}$ denotes the portion of bounding line l_i between its two incident envelopes.*

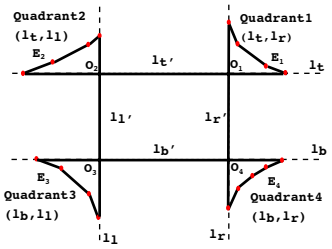


Figure 4: The L_∞ *farthest line-segment hull.*

The L_∞ farthest line-segment Voronoi diagram consists of exactly one unbounded face for each edge of the L_∞ farthest hull. A Voronoi region may consist of only a constant number of disjoint faces. Once the L_∞ farthest hull is available, the L_∞ farthest Voronoi diagram can be constructed in additional $O(h)$ time by adapting the simple algorithm in [1], where h is the size of the farthest hull which may vary from $O(1)$ to $O(n)$. The above results can be easily adapted for L_1, which is equivalent to L_∞ under rotation.

Acknowledgment. We thank the anonymous reviewers for useful comments.

References

[1] F. Aurenhammer, R. L. S. Drysdale and H. Krasser, *Farthest line segment Voronoi diagrams*, Information Processing Letters, vol. 100 no. 6, pp. 220-225, 2006.

[2] L. L. Chen, S. Y. Chou, and T. C. Woo, *Parting directions for mould and die design*, Computer-Aided Design, Vol. 25, no.12, pp. 762-768, 1993.

[3] O. Cheong, H. Everett, M. Glisse, J. Gudmundsson, S. Hornus, S. Lazard, M. Lee, and H. S. Na, *Farthest-Polygon Voronoi Diagrams*, arXiv:1001.3593v1 [cs.CG], 2010.

[4] M.I. Karavelas, *A robust and efficient implementation for the segment Voronoi diagram*, Proc. 1st Int. Symp. on Voronoi Diagrams in Science and Engineering, pp. 51-62, Tokyo, 2004.

[5] K. Mehlhorn, S. Meiser, and R. Rasch, *Furthest site abstract Voronoi diagrams*, Int. J. of Comput. Geometry and Applications, vol. 11, no. 6, 583-616, 2001.

[6] E. Papadopoulou, *Net-aware critical area extraction for opens in VLSI circuits via higher-order Voronoi diagrams*, IEEE Trans. on Comp.-Aided Design, vol. 30, no 5, 704-716, 2011.

[7] E. Papadopoulou and D. T. Lee, *The Hausdorff Voronoi diagram of polygonal objects: A divide and conquer approach*, Int. J. of Computat. Geometry and Applications, vol. 14, no. 6, 421-452, 2004.

[8] E. Papadopoulou and D. T. Lee, *The L_∞ Voronoi Diagram of Segments and VLSI Applications*, Int. J. Comp. Geom. and Applic., vol 11 no 5, 503-528, 2001.

Optimally solving a general transportation problem using Voronoi diagrams

Darius Geiß* Rolf Klein* Rainer Penninger*

Abstract

Let S be a set of n point sites in $\mathbb{R}^d$. A bounded set $C \subset \mathbb{R}^d$ is to be distributed among the sites $p \in S$ such that (i), each p receives a subset C_p of prescribed volume and (ii), the average distance of all points z of C from their respective sites p is minimized. In our model, volume is quantified by a measure μ, and the distance between a site p and a point z is given by a function $d_p(z)$. Under quite liberal technical assumptions on C and on the functions $d_p(\cdot)$ we show that a solution of minimum total cost can be obtained by intersecting with C the Voronoi diagram of the sites in S, based on the functions $d_p(\cdot)$ equipped with suitable additive weights. Moreover, this optimum partition is unique, up to subsets of C of measure zero. Unlike the the deep analytic methods of classical transportation theory, our proof is based on direct geometric arguments.

Keywords: Monge-Kantorovich transportation problem, earth mover's distance, Voronoi diagram with additive weights, Wasserstein metric.

1 Introduction

In 1781, Gaspard Monge [7] raised the following problem. Given two sets C and S of equal mass in $\mathbb{R}^d$, transport each mass unit of C to a mass unit of S at minimal cost. More precisely, given two measures μ and ν, find a map f satisfying $\mu(f^{-1}(\cdot)) = \nu(\cdot)$ that minimizes

$$\int d(z, f(z))\, \mathrm{d}\mu(z),$$

where $d(z, z')$ describes the cost of moving z to z'.

Because of its obvious relevance to economics, and perhaps due to its mathematical challenge, this problem has received a lot of attention. Even with the Euclidean distance as cost function d it is not at all clear in which cases an optimal map f exists. Progress by Appell [1] was honored with a price by the Academy of Paris in 1887. Kantorovich [6] achieved a breakthrough by solving a relaxed version of Monge's original problem. In 1975, he received a Nobel price in Economics; see Gangbo and McCann [4] for mathematical and historical details.

While usually known as the *Monge-Kantorovich transportation problem*, the minimum cost of a transportation is sometimes called *Wasserstein metric* or, in computer science, *earth mover's distance* between the two measures μ and ν. It can be used to measure similarity in image retrieval; see Rubner et al. [9]. If both measures μ and ν have finite support, Monge's problem becomes the minimum weight matching problem for complete bipartite graphs, where edge weights represent transportation cost; see Rote [8] and Vaidya [10].

We are interested in the case where only measure ν has finite support. More precisely, we assume that a set S of n point sites p_i is given, and numbers $\lambda_i > 0$ whose sum equals 1. A body C of volume 1 must be split into subsets C_i of volume λ_i in such a way that the total cost of transporting, for each i, all points of C_i to their site p_i becomes a minimum. In this setting, volume is measured by some (continuous) measure μ, and transport cost $d(z, z')$ by some measure of distance.

Gangbo and McCann [4] report on the cases where either $d(z, z') = h(z - z')$ with a strictly convex function h, or $d(z, z') = l(|z - z'|)$ with a non-negative, strictly concave function l of the Euclidean norm. (Strictness seems necessary to ensure that the cost function's gradient has an inverse.) As a consequence of deep results on the general Monge-Kantorovich problem, they prove a surprising fact. The minimum cost partition of C is given by the additively weighted Voronoi diagram of the sites p_i, based on cost function d and additive weights w_i. In this structure, the Voronoi region of p_i contains all points z satisfying

$$d(p_i, z) - w_i \;<\; d(p_j, z) - w_j \text{ for all } j \neq i.$$

Figure 1 depicts how to obtain this structure in dimension $d = 2$. For each point site $p \in S$ we con-

*University of Bonn, Institute of Computer Science I, Friedrich-Ebert-Allee 144, D-53113 Bonn, Germany. This work was supported by the European Science Foundation (ESF) in the EUROCORES collaborative research project EuroGIGA/VORONOI

struct in $\mathbb{R}^3$ the cone $\{(z, d(p,z) - w_p) \mid z \in \mathbb{R}^2\}$. The lower envelope of the cones, projected onto the XY–plane, results in the Voronoi diagram. Independently, Aurenhammer et al. [2] have studied the case where $d(z,z') = |z - z'|^2$. Here, a *power diagram* (compare [3]) gives an optimum splitting of C. In addition to structural results, they provide algorithms for computing the weights w_i.

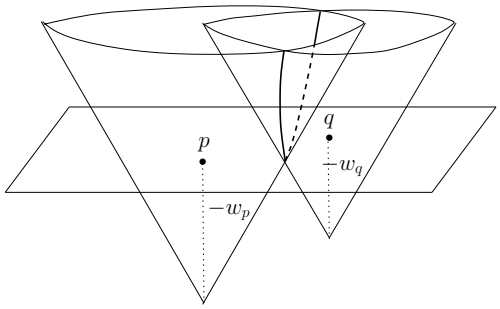


Figure 1: An additively weighted Voronoi diagram as the lower envelope of cones.

In this paper we consider a rather general situation where the cost $d(p_i, z)$ of transporting point z to site p_i is given by individual distance functions $d_{p_i}(z)$. Here, cost need no longer be invariant under translation, as in [4, 2]. All we require is that the weighted Voronoi diagram based on the functions $d_{p_i}(\cdot)$ is well-behaved, in the following sense. Voronoi regions may be disconnected, but together with the Voronoi diagram they form a finite cell decomposition of $\mathbb{R}^d$. Bisectors are $d-1$–dimensional, and increasing weight w_i causes the bisectors of p_i to sweep d–space in a continuous way. These requirements are fulfilled for the Euclidean metric, which was not covered by [4, 2] and, at least in dimension 2, if each site is assigned a strictly convex distance function.

We show that, under these assumptions, the weighted Voronoi diagram based on the functions $d_{p_i}(\cdot)$ optimally solves the transportation problem. After stating some definitions in Section 2 we generalize arguments from [2] to prove, in Section 3, that C can be split into parts of arbitrary given volumes by a weighted Voronoi diagram, for a suitable choice of weights. If C is connected, these weights are uniquely determined, up to addition of a constant. Then, in Section 4, a flow augmentation argument shows that such a partition is optimal, and unique.

2 Definitions

Let μ be a measure defined for all Lebesgue-measurable subsets of $\mathbb{R}^d$. We assume that μ vanishes exactly on the sets of Lebesgue measure zero.

Let S denote a set of n point sites in $\mathbb{R}^d$. For each $p \in S$ we are given a continuous function

$$d_p : \mathbb{R}^d \longrightarrow \mathbb{R}_{\geq 0}$$

that assigns, to each point z of $\mathbb{R}^d$, a nonnegative value $d_p(z)$ as the "distance" from site p to z.

For $p \neq q \in S$ and $\gamma \in \mathbb{R}$ we define

$$B_\gamma(p,q) := \{z \in \mathbb{R}^d \mid d_p(z) - d_q(z) = \gamma\}$$

and

$$R_\gamma(p,q) := \{z \in \mathbb{R}^d \mid d_p(z) - d_q(z) < \gamma\}.$$

The sets $R_\gamma(p,q)$ are open and increase with γ. Now let us assume that for each site $p \in S$ an additive weight w_p is given, and let

$$w = (w_1, w_2, \ldots, w_n)$$

denote the vector of all weights, according to some ordering $p_1, p_2, \ldots$ of S. Then, $B_{w_i - w_j}(p_i, p_j)$ is called the *additively weighted bisector* of p_i and p_j, and

$$\mathrm{VR}_w(p_i, S) := \bigcap_{j \neq i} R_{w_i - w_j}(p_i, p_j)$$

is the *additively weighted Voronoi region* of p_i with respect to S. It consists of all points z for which $d_{p_i}(z) - w_i$ is smaller than all values $d_{p_j}(z) - w_j$, by definition. As usual,

$$V_w(S) := \mathbb{R}^d \setminus \bigcup_i \mathrm{VR}_w(p_i, S)$$

is called the *additively weighted Voronoi diagram* of S; compare [3]. Clearly, $\mathrm{VR}_w(p_i, S)$ and $V_w(S)$ do not change if the same constant is added to all weights in w. Therefore, we may assume that $\min\{w_i | 1 \leq i \leq n\} = 0$ holds whenever this is convenient. Increasing a single value w_i will increase the size of p_i's Voronoi region.

Example. Let $d = 2$ and $d_p(z) = |p - z|$ the Euclidean distance. Given weights w_p, w_q, the bisector $B_{w_p - w_q}(p,q)$ is a line if $w_p = w_q$, and a hyperbola otherwise. Figure 2 (i) shows how $B_\gamma(p,q)$ sweeps across the plane as γ grows from $-\infty$ to ∞. It forms the boundary of the set $R_\gamma(p,q)$ which increases with γ. Each bounded set C in the plane will be reached at some point. From then on the volume of $C \cap R_\gamma(p,q)$ is continuously growing until all of C is contained in $R_\gamma(p,q)$. The increase in

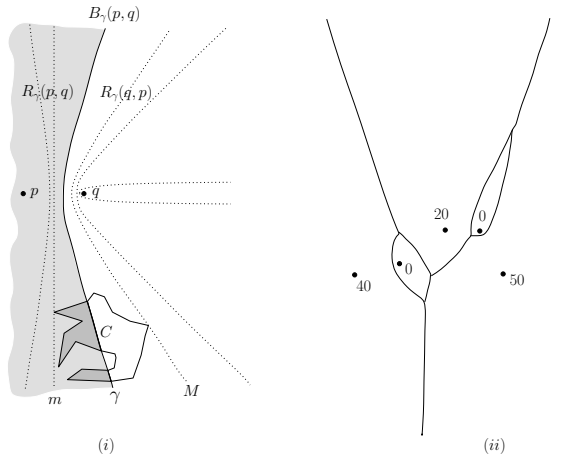


Figure 2: (i) $R_\gamma(p,q)$ for increasing values of γ. (ii) An additively weighted Voronoi diagram.

volume will be strict if each point z of C is an interior point, i. e., if C is open, and C is connected. Given n points p_j with additive weights w_j, raising the value of a single weight w_i will cause all sets $R_{w_i-w_j}(p_i,p_j)$ to grow until C is fully contained in the Voronoi region $V_w(p_i,S)$ of p_i.

In (ii) an additively weighted Voronoi diagram $V_w(S)$ based on the Euclidean distance is shown. It partitions the plane into 5 two-dimensional cells (Voronoi regions), and consists of 9 cells of dimension 1 (Voronoi edges without endpoints) and of 5 cells of dimension 0 (Voronoi vertices). Each cell is homeomorphic to an open sphere of appropriate dimension.

The next definition generalizes the above properties to the setting used in this paper.

Definition 1 *A system of continuous distance functions $d_p(\cdot)$, where $p \in S$, is called* admissible *if the following properties are fulfilled.*

(A) For all $p \neq q \in S$, and for each bounded open set $C \subset \mathbb{R}^d$, $\gamma \longmapsto \mu\left(C \cap R_\gamma(p,q)\right)$ is continuously increasing from 0 to $\mu(C)$ as γ grows from $m_{p,q}$ to $M_{p,q}$, where $C \cap R_\gamma(p,q) = \emptyset$ if $\gamma \leq m_{p,q}$ and $C \subset R_\gamma(p,q)$ if $M_{p,q} \leq \gamma$.

(B) For every nonempty subset $T \subseteq S$ of S the Voronoi regions $VR_w(p_i,T), p_i \in T$, and the Voronoi diagram $V_w(T)$ form a finite *cell decomposition of $\mathbb{R}^d$.*

In (A) we have $m_{q,p} = -M_{p,q}$ and $M_{q,p} = -m_{p,q}$; compare Figure 2, (i). Property (B) ensures that Voronoi diagrams have reasonable structural properties. While Voronoi regions are not required to be connected, they may be split into only finitely many cells of dimension d each. Their boundaries consist of finitely many bisector pieces of dimension $d-1$, that are again bounded by lower dimensional cells consisting of points equidistant to three or more sites. The closures of all Voronoi regions together cover the whole space $\mathbb{R}^d$.

This definition rules out phenomena known for degenerate placement of sites in, e. g., the L_1 norm, where bisectors of points in d-space may be of dimension d. More information about cell decompositions can be found in Hatcher [5].

3 Partitions of prescribed size

Let $d_{p_i}(\cdot)$, $1 \leq i \leq n$, where $n \geq 2$, be an admissible system as in Definition 1, and let C denote a bounded and open subset of $\mathbb{R}^d$. Suppose we are given n real numbers $\lambda_i > 0$ such that $\lambda_1+\lambda_2+\ldots+\lambda_n = 1$ holds. The following theorem shows that we can use an addititively weighted Voronoi diagram based on the functions d_p to partition C into subsets of size $\lambda_i \cdot \mu(C)$.

Theorem 1 *There exists a weight vector $w = (w_1, w_2, \ldots, w_n)$ such that*

$$\mu(C \cap VR_w(p_i,S)) = \lambda_i \cdot \mu(C)$$

holds for $1 \leq i \leq n$. Moreover, if C is pathwise connected then w is unique up to addition of a constant to all w_i.

Proof. W. l. o. g. let $\mu(C) = 1$. The function

$$\Phi(w) := \sum_{i=1}^{n} |\mu(C \cap \mathrm{VR}_w(p_i,S)) - \lambda_i|$$

measures how close w comes to fulfilling the theorem. A simple proof shows that function $\Phi(\cdot)$ is continuous.

Existence. Let $D := \max\{M_{p,q} \mid p \neq q\}$ with $M_{p,q}$ as in Definition 1, (A). On the compact set $[0,D]^n$ function Φ attains its minimum value at some argument w. If $\Phi(w) = 0$ we are done. Suppose that $0 < \Phi(w)$. Since the volumes of the Voronoi regions inside C add up to 1, there must be some sites p_j whose Voronoi regions have an intersection with C of volume $> \lambda_j$, while other region's intersections with C are too small.

In the full version we show how to construct from w a weight vector w' where $\mu(C \cap \mathrm{VR}_{w'}(s,S)) > 0$ holds for all $s \in S$, and also satisfying $\Phi(w') \leq \Phi(w)$.

If $0 < \Phi(w')$ we do the following: We chose some value $\delta > 0$ and raise the weights of all sites whose Voronoi regions currently have too small

an intersection with C by δ, increasing the size of at least one of those regions. Choosing δ small enough we ensure that the size of any region increased this way will stay too small, and the size of any region previously too large will also remain too large. Moreover, the size of the intersection of any region with C will remain positive.

We repeat this process, each time choosing a new value δ, until $\Phi(w')$ decreases. If the increase of weights does not result in a decrease of $\Phi(w')$ the size of at least one region which has the right size will become too small. Thus, if $\Phi(w')$ has not decreased after $n-2$ iterations, all regions have non-empty intersections with C that are either too small or too large. As we increase once more simultaneously the weights of all regions that are too small, their gain in size is caused only by losses of regions too large. For the resulting weight vector w'' this implies $\Phi(w'') < \Phi(w') \leq \Phi(w)$. (If $0 = \Phi(w')$ we have $\Phi(w') < \Phi(w)$ and continue with w' instead of w''.)

Since all Voronoi regions based on w'' have intersections of positive size with C, property (A) of Definition 1 implies that

$$w''_i - w''_j < M_{p_i,p_j} \leq D$$

holds for all $i \neq j$. Let w''' result from w'' by subtracting the minimum weight w''_j from all w''_i. Then, $w''' \in [0, D]^n$ and $\Phi(w''') = \Phi(w'') < \Phi(w)$—a contradiction!

Uniqueness. The proof of uniqueness can be found in the full version. □

It is easy to see why connectedness of C is a condition necessary for the uniqueness of w. Namely, assume that each connected component of C is completely contained in a separate Voronoi region. Now if the weights of the points in S are modified by some small amount then the resulting Voronoi diagram could still be the same inside C.

4 Optimality

Again, let $d_p(\cdot)$, $p \in S = \{p_1, p_2, \ldots, p_n\}$, be an admissible system as in Definition 1, and let C denote a bounded and open subset of $\mathbb{R}^d$. Moreover, let real numbers $\lambda_i > 0$ be given such that $\lambda_1 + \lambda_2 + \ldots + \lambda_n = 1$ holds.

By the existence part of Theorem 1, there exists a weight vector $w = (w_1, w_2, \ldots, w_n)$ satisfying

$$\mu(C \cap \mathrm{VR}_w(p_i, S)) = \lambda_i \cdot \mu(C) \text{ for } i = 1, \ldots, n.$$

Now we prove that this subdivision of C minimizes the transportation cost, i. e., the average distance from each point to its site. It is convenient to describe partitions of C by maps $f : C \longrightarrow S$ where, for each $p \in S$, $f^{-1}(p)$ denotes the region assigned to p. Let F_Λ denote the set of those maps f satisfying $\mu(f^{-1}(p_i)) = \lambda_i \cdot \mu(C)$ for $i = 1, \ldots, n$.

Theorem 2 *The partition of C into regions $C_i := C \cap VR_w(p_i, S)$ minimizes*

$$cost(f) := \sum_{p \in S} \int_{f^{-1}(p)} d_p(z)\, \mathrm{d}\mu(z)$$

over all maps $f \in F_\Lambda$. Any other partition of minimal cost differs at most by sets of measure zero from $(C_i)_i$.

The proof of Theorem 2 combines a flow argument used for improving non-optimal transport plans (based on cyclical monotonicity, see e.g. [11]) with observations based on the structural properties of Voronoi diagrams.

Note that Theorem 2 and Theorem 1 together imply in particular that even if C is not connected, then two partitions of C into subsets of size $\lambda_i \cdot \mu(C)$, obtained using additively weighed Voronoi diagrams, differ at most by sets of measure zero.

References

[1] P. Appell. Mémoire sur les déblais et les remblais des systèmes continues ou discontinus. Mémoires présentes par divers Savants à l'Academie des Sciences de l'Institut de France 29 (1887): 1–208.

[2] F. Aurenhammer, F. Hoffmann, and B. Aronov. Minkowski-type theorems and least-squares clustering. Algorithmica 20 (1998): 61–76.

[3] F. Aurenhammer and R. Klein. Voronoi diagrams. In: J.R. Sack and G. Urrutia (Eds.), Handbook on Computational Geometry, Elsevier (1999): 201–290.

[4] W. Gangbo and R. J. McCann. The geometry of optimal transportation. Acta Math. 177 (1996): 113–161.

[5] A. Hatcher. Algebraic Topology. Cambridge University Press, 2001.

[6] L. Kantorovich. On a problem of Monge (In Russian). Uspekhi Math. Nauk. 3 (1948): 225–226.

[7] G. Monge. Mémoire sur la théorie des déblais et de remblais. Histoire de l'Académie Royale des Sciences de Paris, avec les Mémoires de Mathématique et de Physique pour la même année 29 (1781): 666–704.

[8] G. Rote. Two applications of point matching. Abstracts of the 25th European Workshop on Computational Geometry (EuroCG'09): 187–189

[9] Y. Rubner, C. Tomasi, and L. J. Guibas. A metric for distributions with applications to image databases. Proceedings International Conference on Computer Vision (ICCV '98): 59–66.

[10] P.M. Vaidya. Geometry helps in matching. SIAM J. Comput. 18 (1989): 1201–1225.

[11] C. Villani. *Optimal Transport, Old and New*, volume 338 of *Grundlehren der mathematischen Wissenschaften*. Springer, 2009.

Higher Order City Voronoi Diagrams

Andreas Gemsa* D. T. Lee† Chih-Hung Liu† Dorothea Wagner*

*Karlsruhe Institute of Technology, Germany, †Academia Sinica, Taiwan

Abstract

We investigate the higher-order Voronoi diagrams in the L_1 metric in the presence of a transportation network, which is commonly referred to as city metric. For the structural complexity of k^{th}-order city Voronoi diagrams we show a lower bound of $\Omega(n+kc)$ and an upper bound of $O(k(n-k)+kc)$, where c is the complexity of the transportation network. This is quite different from the bound $O(k(n-k))$ in the Euclidean metric [8], especially for the case when $k = n-1$. Furthermore, we develop an $O(k^2(n+c)\log n)$-time algorithm for the k^{th}-order city Voronoi diagram and an $O(nc\log^2(n+c)\log n)$-time algorithm for the farthest-site city Voronoi diagram.

1 Introduction

The road network of modern, well-designed cities, like Manhattan, resembles a grid. Pedestrians can only move either horizontally or vertically. In larger cities we usually require a high-speed public transportation network (e.g. bus and rail systems) to ensure easy and fast travel between two places.

We assume that the traveling speed on the transportation network is a given parameter $\nu > 1$; otherwise the traveling speed is 1. Further, we assume that the transportation network can be accessed at any point. Shortest paths in the L_1 metric accelerated by a high-speed transportation network induce a new metric, the so-called *city metric*. In this metric the distance between two points is the minimum time required to travel between them.

Given a set of sites, the k^{th}-order city Voronoi diagram partitions the plane into regions such that all points in a region share the same k nearest sites with respect to the city metric. Higher-order city Voronoi diagrams can be used to resolve the following situation: a pedestrian wants to know the k nearest stores, facilities, or hospitals such that he can make a well-informed decision as to which facility to go.

First-order city Voronoi diagrams have already been well-studied [1, 3, 4, 5, 7]. Their space complexity has been shown to be in $O(n+c)$ [1], and they can be constructed in $O((n+c)\log(n+c))$ time [5], where c is the complexity of the transportation network.

Although higher-order Euclidean Voronoi diagrams have been well-studied, higher-order city Voronoi diagrams have not yet been investigated. One of the most significant differences between the Euclidean metric and the city metric that influences the computation and complexity of Voronoi diagrams is the complexity of a bisector of two points. In the Euclidean metric or the L_1 metric such a bisector has constant complexity, while in the city metric the complexity may be $\Omega(c)$ [1]. This makes it non-trivial to apply existing approaches for constructing ordinary Voronoi diagrams to the city Voronoi diagrams.

Our Contribution. In this paper, we make use of ideas presented in the iterative construction [8] and the wavefront model [1, 5] to derive the structural complexity of the k^{th}-order city Voronoi diagram. We show that it has an upper bound $O(k(n-k)+kc)$ and a lower bound $\Omega(n+kc)$, where c is the complexity of the transportation network, which is quite different from the $O(k(n-k))$ bound in the Euclidean metric [8]. When $k = n-1$, the structural complexity in the Euclidean metric is $O(n)$, while it is $\Theta(nc)$ in the city metric. We also develop an iterative algorithm to compute the k^{th}-order city Voronoi diagram in $O(k^2(n+c)\log(n+c))$ time and a divide-and-conquer approach to compute the farthest-site city Voronoi diagram in $O(nc\log n\log^2(n+c))$ time.

2 Preliminaries

Here, we introduce definitions and explain two commonly used concepts, the wavefront model and shortest path map, for the subsequent sections.

A *transportation network* is a plane graph $C = (V_C, E_C)$ with a fixed straight-line embedding and only *isothetic* edges, i.e., edges that are either horizontal or vertical. Each edge has the same traveling speed ν. Throughout this paper we refer to the number of vertices in the transportation network as c. Since the number of edges in a transportation network is in $O(c)$ its complexity is also in $O(c)$.

The distance d_C between two points in the city metric, which we denote with d_C, is the minimum time required to travel between those two points. Likewise, let d_1 be the distance between two points in the L_1 metric. Similarly, let B_C denote the bisector between two points w.r.t. the city metric and let B_1 denote the bisector between two points w.r.t. the L_1 metric.

This is an extended abstract of a presentation given at EuroCG 2012. 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 in a conference with formal proceedings and/or in a journal.

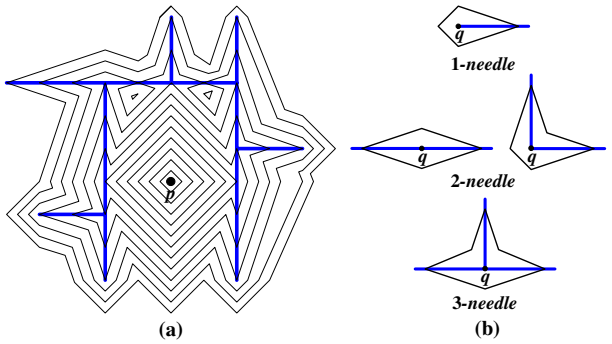


Figure 1: (a) wavefront propagation. (b) j-needles.

For a given set S of n point sites and a transportation network C, the k^{th}-order city Voronoi diagram $V_k(S)$ partitions the plane into a number of *Voronoi regions* $V_k(H, S)$ where $H \subset S$ and $|H| = k$. All points in $V_k(H, S)$ share the same k nearest sites H with respect to the city metric. The common boundary between two Voronoi regions $V_k(H_1, S)$ and $V_k(H_2, S)$ is a Voronoi edge, which is a part of $B_C(p, q)$ [8], where $|H_1 \cap H_2| = k - 1$, $H_1 \setminus H_2 = \{p\}$ and $H_2 \setminus H_1 = \{q\}$. The common intersection among more than two Voronoi regions is a *Voronoi vertex*.

For a point $v \in \mathbb{R}^2$, let $P(v)$ denote the *isothetic projection* of v onto the transportation network, i.e., we shoot an isothetic half–ray starting at v in each of the four directions and for each half–ray we add its first intersection with an edge of the transportation network to $P(v)$. For a set $X \subset \mathbb{R}^2$ we denote the *isothetic projection of the set* X as $\mathcal{P}(X) = \bigcup_{v \in X} P(v)$. Finally, for a site $p \in S$ we call the set $A(p) = P(p) \cup V_C \cup \mathcal{P}(V_C) \cup \{p\}$ *activation points*.

Following [2], we make the general position assumption: no point in the plane is equidistant from four sites in S in the city metric. Hence, the degree of a Voronoi vertex is exactly three.

Wavefront Model. For a fixed site $p \in S$, let $W_p(x) = \{q \mid q \in \mathbb{R}^2, d_C(p, q) = x\}$. This means that for a fixed $x \in \mathbb{R}_0^+$ the wavefront $W_p(x)$ is the circle centered at p with radius x. We call p the *source* of $W_p(x)$. Note that we can view $W_p(x)$ as the wavefront at time x of the wave that originated in p at time 0. We refer to the wavefront as W_p if the exact value of x is unimportant.

Initially, W_p is diamond shaped. Only if it hits a point in $A(p)$ it changes its shape [1]. Since the shape of W_p can become very complex after it hits multiple activation points, we make the following assumption: if a wavefront W_p touches a point $q \in A(p)$ we do not change the propagation speed of W_p. Instead we start a new wavefront at q, which, in turn, starts new wavefronts at points in $A(p)$ if it reaches them earlier than any other wavefront. Hereafter, the propagation of a new wavefront is called an *event*. The

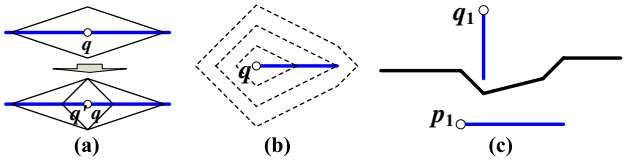


Figure 2: (a) A 2-needle is two 1-needles. (b) Propagation of 1-needle. (c) A bisector between $\eta_p(p_1)$ and $\eta_q(q_1)$.

shape of this new wavefront depends on the position of q on the transportation network. It can be categorized into three different shapes: 1-needle, 2-needle, and 3-needle [1] (see Fig. 1(b)). Henceforth we treat a 2(3)-needle as two (three) 1-needles (see Fig. 2(a)).

When a 1-needle reaches the end of a network segment, as shown in Fig. 2(b), its shape will change [1]. In order to interpret the propagation of 1-needle, Bae et al. [5] introduced the term *needle*. A needle $\eta_p(q, q')$ is the network segment $\overline{qq'}$ with weight $d_C(p, q)$, where $p \in S$ and $q, q' \in A(p)$. Propagating a wavefront from $\eta_p(q, q')$ is equivalent to propagating a 1-needle from q on a network segment $\overline{qq'}$ at time $d_C(p, q)$. If q' is obvious or unimportant we may refer to $\eta_p(q, q')$ as $\eta_p(q)$. Bae et al. [5] also defined the L_1 distance between a point and a needle and the L_1 bisector between two needles (see Fig. 2(c)).

Shortest Path Map For a site $p \in S$ its shortest path map $\mathcal{SPM}_p$ is a planar subdivision that can be obtained as follows: start by propagating a wavefront from the site p. When a point $q \in A(p)$ is touched for the first time by a wavefront, propagate an additional wavefront from q. Eventually, each point $r \in \mathbb{R}^2$ is touched for the first time by a wavefront propagated from a source $q \in A(p)$ and we associate r with q. This induces $\mathcal{SPM}_p$ which partitions the plane into at most $|A(p)|$ regions $\mathcal{SPM}_p(q)$. As proved in [5], an edge of $\mathcal{SPM}_p$ is part of a bisector between two needles $\eta_p(q)$ and $\eta_p(q')$, where $q, q' \in A(p)$. For an illustration of shortest path maps see Fig. 3. The black polyline is the Voronoi edge between $V_1(\{p\}, \{p, q\})$ and $V_1(\{q\}, \{p, q\})$. Both regions are further partitioned by $\mathcal{SPM}_p$ and $\mathcal{SPM}_q$, respectively.

3 Structural Complexity

In this section we show an upper and a lower bound for the complexity of k^{th}-order city Voronoi diagrams.

For two sites $p, q \in S$ and a Voronoi edge e which is part of $B(p, q)$, a point r on e is *a mixed vertex* if there are $p_1, p_2 \in A(p)$ and $q_1 \in A(q)$ such that $r \in \mathcal{SPM}_p(p_1) \cap \mathcal{SPM}_p(p_2) \cap \mathcal{SPM}_q(q_1)$. This implies: $d_1(r, \eta_p(p_1)) = d_1(r, \eta_p(p_2)) = d_1(r, \eta_q(q_1))$. A similar concept is known for farthest-polygon Voronoi diagrams and called mixed Voronoi vertices [6]. For example, as shown in Fig. 3, since $m_2 \in \mathcal{SPM}_p(p_1) \cap \mathcal{SPM}_p(p_2) \cap \mathcal{SPM}_q(q_1)$, m_2 is a mixed vertex.

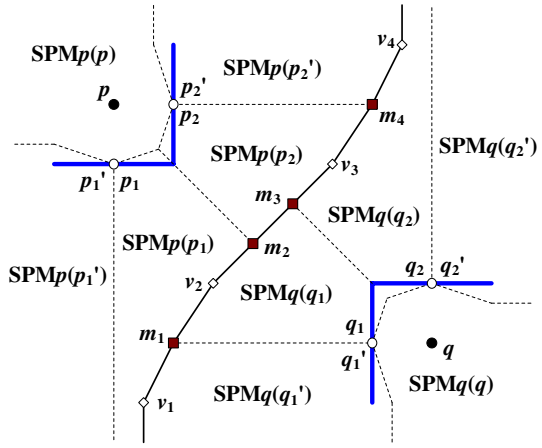


Figure 3: $B_C(p, q)$, where m_i (1≤i≤4) is a mixed vertex.

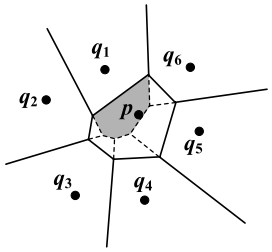


Figure 4: Solid segments form $V_1(S)$, dashed segments are part of $V_1(Q)$ and form $V_2(S) \cap V_1(\{p\}, S)$, and gray region is part of $V_2(\{p, q_1\}, S)$, where $Q = \bigcup_{1 \le i \le 6} q_i$ and $S = Q \cup \{p\}$.

Lemma 1 *The complexity of the k^{th}-order city Voronoi diagram is $O(M + k(n - k))$, where M is the total number of mixed vertices.*

Proof. According to the definition of mixed vertices, if a Voronoi edge e_i contains m_i mixed vertices, e_i consists of $m_i + 1$ parts. We can show that each of these parts belongs to a bisector between two needles. Since the size of a bisector between two needles is $O(1)$ [4], the complexity of e_i is $O(m_i + 1)$. Suppose $V_k(S)$ contains N_e Voronoi edges, $e_1, e_2, \ldots, e_{N_e}$, and each e_i contains m_i mixed vertices. Then, the complexity of all edges is $\sum_{i=1}^{N_e} O(m_i + 1) = O(M + N_e)$. By [8], we know that $N_e = O(k(n - k))$, and thus it follows: $O(M + N_e) = O(M + k(n - k))$. □

In order to derive an upper bound for the complexity of city Voronoi diagrams, we further categorize the mixed vertices. Let m be a mixed vertex on the Voronoi edge between $V_k(H_1, S)$ and $V_k(H_2, S)$, where $H_1 \setminus H_2 = \{p\}$ and $H_2 \setminus H_1 = \{q\}$. We call m an *interior mixed vertex* of $V_k(H_1, S)$ if $m \in \mathcal{SPM}_p(p_1) \cap \mathcal{SPM}_p(p_2) \cap \mathcal{SPM}_q(q_1)$, for some $p_1, p_2 \in A(p)$ and $q_1 \in A(q)$; otherwise, we call m an *exterior mixed vertex* of $V_k(H_1, S)$. In Fig.3 the vertices m_2 and m_4 both are interior mixed vertices of $V_1(\{p\}, \{p, q\})$ and exterior mixed vertices of $V_1(\{q\}, \{p, q\})$.

Upper Bound. The $(j+1)$-th order Voronoi diagram $V_{j+1}(S)$ of a set S of point sites can be constructed from $V_j(S)$ [8]. The idea is as follows: consider a Voronoi region $V_j(H, S)$. Suppose that $V_j(H, S)$ has ℓ adjacent Voronoi regions $V_j(H_i, S)$ for $1 \le i \le \ell$. Further, let $H_i \setminus H$ be $\{q_i\}$ and let Q be $\bigcup_{1 \le i \le \ell} q_i$. It can be shown that $V_j(H, S) \cap V_1(Q) = V_j(H, S) \cap V_{j+1}(S)$ [8] which lets us determine $V_{j+1}(S)$. Fig. 4 shows an example for the Euclidean metric. This technique is equivalent to propagating a wavefront from each site $q_i \in Q$ into the Voronoi region $V_j(H, S)$. A point $r \in \mathbb{R}^2$ is in $V_{j+1}(H \cup \{q_i\}, S)$ if it is first touched by the wavefront that propagated from q_i.

We extend this wavefront approach to the city metric. Suppose e_i is the Voronoi edge between $V_j(H_i, S)$ and $V_j(H, S)$ and e_i contains m exterior mixed vertices with respect to $V_j(H, S)$, i.e., e_i intersects $m+1$ regions in $\mathcal{SPM}_{q_i}$. Further, let the $m + 1$ regions be $\mathcal{SPM}_{q_i}(v_z)$ for $1 \le z \le m + 1$. Instead of propagating a wavefront from q_i we propagate $m + 1$ wavefronts from $\eta_{q_i}(v_1), \ldots, \eta_{q_i}(v_{m+1})$. When a point r is *first* touched by a wavefront from $\eta_{q_i}(v), v \in A(q_i)$, we propagate a new wavefront from r if (i) $r \in \mathcal{P}(V_C) \cup V_C$ or (ii) $q_i = v$ and $r \in P(q_i)$.

Lemma 2 *Let $V_j(S)$ be a j-th order city Voronoi diagram for a set S of point sites. If a Voronoi region $V_j(H, S)$ contains m exterior mixed vertices, then $V_j(H, S) \cap V_{j+1}(S)$ contains at most $m+2c'+2a'$ mixed vertices, where $c' = |(\mathcal{P}(V_C) \cup V_C) \cap V_j(H, S)|$ and a' is the number of events associated to points in $\mathcal{P}(S)$.*

Proof. Let $V_k(H, S)$ have ℓ adjacent regions and let Q be the set as described above. Further, let $\ell' = |Q|$. According to the above discussion, we propagate $m + \ell'$ wavefronts into $V_j(H, S)$. All those wavefronts combined generate c' new wavefronts from points in $(\mathcal{P}(V_C) \cup V_C) \cap V_j(H, S)$, and a' new wavefronts from points in $\mathcal{P}(S) \cap V_j(H, S)$. Let W be the set of those $m + \ell' + c' + a'$ wavefronts. For each point $r \in V_j(H, S)$, if r is first touched by a wavefront $w \in W$ it is associated with w. This will partition $V_j(H, S)$ into $m + \ell' + c' + a'$ regions. We view those regions as a special Voronoi diagram $V_1(W)$. Note that $m + \ell'$ of those regions are unbounded.

It is clear that $V_j(H, S) \cap V_{j+1}(S)$ is a subgraph of $V_1(W)$. Without loss of generality, we assume each vertex of $V_1(W)$ has degree 3. Since $V_1(W)$ contains $m+\ell'$ unbounded regions, by using Euler's formula we can show that $V_1(W)$ contains $m + \ell' + 2c' + 2a' - 2$ vertices. Since $V_{j+1}(H \cup \{q_i\}, S) \cap V_j(H, S)$ is unbounded and $|Q| = \ell'$, there are $\ell' - 2$ Voronoi vertices in $V_j(H, S) \cap V_{j+1}(S)$ and thus at most $(m + \ell' + 2c' + 2a' - 2) - (\ell' - 2) = m + 2c' + 2a'$ mixed vertices. □

By Lemma 2, computing $V_{j+1}(S)$ from $V_j(S)$ creates at most $m_j + O(c) + 2a_j$ new mixed vertices, where m_j is the number of mixed vertices in $V_j(S)$

and a_j is the number of events associated with points in $\mathcal{P}(S)$. We can show that the total number of events associated with points in $\mathcal{P}(S)$ to compute $V_{j+1}(S)$ for $1 \leq j \leq k-1$ by this approach is $O(n)$. Since $m_1 = O(n+c)$ and $\sum_{j=1}^{k-1} a_j = O(n)$ we can show that $\sum_{j=1}^{k}(m_j + O(c) + 2a_j) = O(n+kc)$ which proves, combined with Lemma 1, the following:

Theorem 1 *$V_k(S)$ contains $O(n+kc)$ mixed vertices, and its complexity is $O(k(n-k)+kc)$.*

Lower Bound. We give a worst-case example to derive a lower bound for the number of mixed-Voronoi vertices. The example consists of a left and a right part which are placed with a sufficiently large distance between them. As shown in Fig. 5, we place one network segment in the left part and build a stair-like transportation network in the right part. There are $k+1$ sites in the right part and the remaining $n-k-1$ sites are in the left part. Since the distance between the left and right parts is large, the $n-k-1$ sites in the left part hardly influence the formation of $V_k(S)$ in the right part. Therefore, $V_k(S)$ in the right part forms the farthest-site city Voronoi diagram of the $k+1$ sites, where all points in Region i share the same farthest site s_i. Note that sharing the same k nearest sites among $k+1$ sites is equivalent to sharing the same farthest site. The common Voronoi edge between Region i and Region $(i+1)$ contains at least $\frac{c-6}{2}$ mixed vertices, implying that the right part has $(k-1)\frac{c-6}{2} = \Omega(kc)$ mixed vertices. The example in Fig. 5 shows the case for $k=3$. This example can be extended to the case $k > 4$ by inserting $s_5, s_6, \ldots, s_{k+1}$ vertically between s_1 and s_4. Since $V_k(S)$ has $\Omega(n)$ regions, we conclude the following:

Theorem 2 *The complexity of $V_k(S)$ is $\Omega(n+kc)$.*

4 Algorithm

In this section we describe an algorithm that computes k^{th}-order city Voronoi diagrams based on the ideas described in Section 3 and Bae et al.'s [5] $O((n+c)\log(n+c))$-time algorithm for first-order city Voronoi diagrams.

We give the description of the algorithm for a single Voronoi region $V_j(H,S)$. All of its four steps have to be repeated for each Voronoi region of $V_j(S)$. Let $V_j(H,S)$ have ℓ adjacent regions $V_j(H_i,S)$ with $H_i \setminus H = \{q\}$ and let $Q = \bigcup q_i$: i) For each i, let the Voronoi edge between $V_j(H_i,S)$ and $V_j(H,S)$ intersect m'_i regions $\mathcal{SPM}_{q_i}(v_z)$, $1 \leq z \leq m'_i$. Insert $\eta_{q_i}(v_z)$ into a set S'. ii) Reduce the transportation network C to C': for every edge e in C insert it into C' if and only if there is a point v on e that is in $(\mathcal{P}(V_C) \cup \mathcal{P}(Q) \cup V_C) \cap V_j(H,S)$. iii) Perform Bae et al.'s wavefront-based approach to compute $V_1(S')$ under C'. iv) Determine $V_j(H,S) \cap V_{j+1}(S)$ from $V_1(S')$.

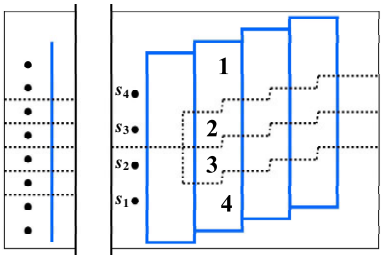


Figure 5: A worst-case example where bold solid segments composes C and dashed segments compose $V_k(S)$.

Theorem 3 *$V_{j+1}(S)$ can be computed from $V_j(S)$ in $O((j(n-j)+jc)\log(n+c))$ time, and $V_k(S)$ can be computed in $O(k^2(n+c)\log(n+c))$ time.*

By replacing the medial axes in [6] with the $\mathcal{SPM}$s, we can modify the divide-and-conquer algorithm in [6] for the farthest-polygon Voronoi diagram to compute the farthest-site city Voronoi diagram.

Theorem 4 *$V_{n-1}(S)$ can be computed in $O(nc\log n\log^2(n+c))$ time.*

Acknowledgment. A. Gemsa received financial support by the *Concept for the Future* of KIT within the framework of the German Excellence Initiative. D. T. Lee and C.-H. Liu are supported by the National Science Council, Taiwan under grants No. NSC-98-2221-E-001-007-MY3 and No. NSC-99-2911-I-001-506.

References

[1] O. Aichholzer, F. Aurenhammer, and B. Palop, "Quickest paths, straight skeletons, and the city Voronoi diagram," *Discrete & Comput. Geom.*, vol. 31, pp. 17–35, 2004.

[2] F. Aurenhammer and R. Klein, "Voronoi Diagrams," *Handbook of Computational Geometry*, Elseiver, 2000 .

[3] S. W. Bae and K.-Y. Chwa, "Voronoi Diagrams with a Transportation Network on the Euclidean Plane," *Proc. Internat. Symposium on Algorithm and Computation*, pp. 101–112, 2004.

[4] S. W. Bae and K.-Y. Chwa, "Shortest paths and Voronoi diagrams with transportation networks under general distances," *Proc. Internat. Symposium on Algorithm and Computation*, pp. 1007–1018, 2005.

[5] S. W. Bae, J.-H. Kim, K-Y. Chwa, "Optimal construction of the city Voronoi diagram," *Internat. J. Comput. Geom. Appl.*, vol. 19, no. 2, pp. 95–117, 2009.

[6] O. Cheong, H. Everett, M. Glisse, J. Gudmundsson, S. Hornus, S. Lazard, M. Lee, and H.-S. Na, "Farthest-polygon Voronoi diagrams," *Comput. Geom. Theory Appl.*, vol. 44, pp. 234–247, 2011.

[7] R. Görke, C.-S. Shin, and A. Wolff, "Constructing the city Voronoi diagram faster," *Internat. J. Comput. Geom. Appl.*, vol. 18, no. 4, pp. 275–294, 2008.

[8] D. T. Lee, "On k-nearest neighbor Voronoi Diagrams in the plane," *IEEE Trans. on Computers*, Vol. 31, No. 6, pp. 478–487, 1982.

Colored Quadrangulations with Steiner Points

Victor Alvarez* Atsuhiro Nakamoto†

Abstract

Let P be a k-colored set of n points in general position on the plane, where $k \geq 2$. A k-colored quadrangulation of P is a *maximal* straight-edge plane graph with vertex set P satisfying the property that every interior face is a properly colored quadrilateral, *i.e.*, no edge connects vertices of the same color. It is easy to check that in general not every set of points admits a k-colored quadrangulation, and hence the use of extra points, for which we can choose the color among the k available colors, is required in order to obtain one. The extra points are known in the literature as *Steiner points*. In this paper, we show that if P satisfies some condition for the colors of the points in $Conv(P)$, then a k-colored quadrangulation of P can always be constructed using less than $\frac{(16k-2)n+7k-2}{39k-6}$ Steiner points. Our upper bound improves the previously known upper bound for $k = 3$, and represents the first bounds for $k \geq 4$.

1 Introduction

Let P be an *n-point set*, that is, a set of n points in general position on the plane. We say that P is k-colored if every point of P is colored with *exactly* one of k available colors. A quadrangulation of P is a *maximal* straight-edge plane graph with vertex set P such that every interior face is a quadrilateral. For a k-colored point set P, a quadrangulation of P is said to be k-colored if no edge of the quadrangulation connects vertices of the same color. From now on, unless stated otherwise, we will always consider P as being k-colored. Also, in our setting we are interested in having $Conv(P)$ as the outer cycle of the k-colored quadrangulations of P, thus we will always assume that any two consecutive points on $Conv(P)$ have distinct colors.

The study on k-colored quadrangulations of point sets is rather new. It is easy to see that even when $Conv(P)$ is properly colored, k-colored quadrangulations do not always exist, see [1]. Thus, if a k-colored point set P is given, and one insists on constructing a k-colored quadrangulation of P, then the use of extra points is in general needed. These extra points are known in the literature as Steiner points, and in our setting, the color of each Steiner point can be chosen from the k available colors. In [2] it was shown that for any bichromatic n-point set P, one can construct a bichromatic quadrangulation of P with the use of roughly $\frac{5n}{12}$ interior Steiner points. Those are Steiner points introduced only inside $Conv(P)$. There, it was also shown that $\frac{n}{3}$ interior Steiner points are sometimes necessary. They also considered the case when $k = 3$, and showed a surprising fact, there are 3-colored point sets that *do not* admit 3-colored quadrangulations regardless of the number of interior Steiner points used, which is definitely an unexpected result.

The strange phenomenon of not admitting 3-colored quadrangulations, even with the use of Steiner points, was recently explained in [3], where the authors showed an elegant characterization of the 3-colored point sets that admit 3-colored quadrangulations using a *finite* number of interior Steiner points. In the same paper, the authors showed that if possible, a 3-colored quadrangulation can be constructed with the use of at most $\frac{7n+17m-48}{18}$ interior Steiner points, where $|P| = n$ and $|Conv(P)| = m$. Note however that this number depends on the size of $Conv(P)$, and can get larger than wished whenever m and n are comparable in size. For example, if $m = \frac{3n}{4}$, then the bound becomes $\frac{79n-192}{72}$ which is larger than n already when $n \geq 5$.

In this paper, we show how one can use the algorithm for the bichromatic case to obtain an algorithm for a k-colored point set, for general $k \geq 3$. Our algorithm uses less than $\frac{(16k-2)n+7k-2}{39k-6}$ interior Steiner points to construct a k-colored quadrangulation of P. Our bound has the following advantages: (1) Our algorithm fully replaces the algorithm shown in [3], since it performs equally good when $Conv(P)$ is small, but it improves the worst-case behavior when $Conv(P)$ is large. For comparison, our bound for $k = 3$, at worst, is essentially $\frac{46n}{111} < \frac{5n}{12}$, while the one presented in [3] can grow larger than n if the right conditions are met. (2) Our bound represents the first bounds for the cases when $k \geq 4$.

We will divide the paper as follows: in section 2 we give the necessary definitions and the precise statement of our result, and in section 3 we prove our main Theorem.

*Fachrichtung Informatik, Universität des Saarlandes, alvarez@cs.uni-saarland.de. Partially Supported by CONACYT-DAAD of México.

†Department of Mathematics, Yokohama National University, nakamoto@ynu.ac.jp.

2 Preliminaries

In order to make this paper more self-contained, we will state the results from other papers that will be used, and will be referred to. Let us first start with some terminology. Let Q be an m-sided convex polygon, with $m \geq 4$ even, and suppose that Q is properly k-colored, where $k \geq 2$. The following are the definitions taken from [3].

Let us assume that the k chromatic classes used to color Q are $1, 2, \ldots, k$, and allow us denote the color of a vertex v of Q by $c(v)$. Let us define an orientation $\mathcal{O}$ for the edges of Q as follows: if $e = uv$ is an edge of Q, then we orient e from u to v if $c(u) < c(v)$, and from v to u otherwise. Let $e^+_{\mathcal{O}}(Q)$ and $e^-_{\mathcal{O}}(Q)$ be the number of edges in clockwise and in counter-clockwise direction respectively.

Definition 1 (Winding number) *Let $\mathcal{O}$ be an orientation of Q as explained before. The winding number of Q, denoted by $\omega(Q)$, is defined as:*

$$\omega(Q) = |e^+_{\mathcal{O}}(Q) - e^-_{\mathcal{O}}(Q)|$$

for $k = 3$, and $\omega(Q) = 0$ for $k \neq 3$.

Observe that the winding number of a polygon Q is non-trivial only when Q is 3-colored.

For a point set P, we will use $\omega(P)$ as a shorthand for $\omega(Conv(P))$, extending the definition of winding number for polygons to sets of points. Finally, if P is k-colored, with $k \geq 2$, we will say that P can be *k-quadrangulated* if P admits a k-colored quadrangulation.

The following result is the one described in the introduction characterizing the 3-colored point sets which can be 3-quadrangulated with Steiner points added [3].

Theorem 1 (S.Kato, R. Mori, A. Nakamoto) *Let P be a 3-colored n-point set in general position on the plane such that $|Conv(P)| = m$. Then there exists a set $\mathbb{S}$ of Steiner points such that $P \cup \mathbb{S}$ can be 3-quadrangulated if and only if $\omega(P) = 0$. In such a case, $|\mathbb{S}| \leq \frac{7n+17m-48}{18}$.*

Now we can easily decide whether a 3-colored point set admits a 3-colored quadrangulation. Nevertheless, as we mentioned before, the number of Steiner points required by Theorem 1 can get larger than wished when m and n are comparable in size.

The main contribution of this paper is the following:

Theorem 2 *Let $k \geq 2$ be an integer, and let P be a k-colored n-point set in general position on the plane. If $\omega(P) = 0$ or $k \geq 4$, then there exists a set $\mathbb{S}$ of Steiner points such that $P \cup \mathbb{S}$ can be k-quadrangulated, and $|\mathbb{S}| < \frac{(16k-2)n+7k-2}{39k-6}$.*

Note that our Theorem, besides of being able to work with more than three chromatic classes, depends only on n and k, which is a great improvement over the previously known bound for $k = 3$.

3 Main Theorem

In order to prove our Theorem, we will need some intermediate results, the first one is easily proven using the well known Euler's formula:

Lemma 3 *Let P be an n-point set in general position on the plane where m of them lie on $Conv(P)$. Then any quadrangulation of P has $(n-1) - \frac{m}{2}$ quadrilaterals and $2(n-2) - \frac{m}{2}$ edges.*

In [3] the following Lemma was shown:

Lemma 4 (S.Kato, R. Mori, A. Nakamoto) *Let Q be a 3-colored convex polygon colored by three colors c_1, c_2, c_3. Then the winding number of Q is invariant for any bijection from $\{c_1, c_2, c_3\}$ to $\{1, 2, 3\}$.*

That is, the winding number is well-defined, and we may assume that if Q is a 3-colored convex polygon, then it is colored by $\{1, 2, 3\}$. We now have the following Lemma:

Lemma 5 *Let Q be a properly k-colored convex polygon of $m \geq 4$ sides such that $\omega(Q) = 0$. Then Q can be partitioned into $r = \frac{m-2}{2}$ properly colored quadrilaterals $Q_1, \ldots, Q_r$ such that $\omega(Q_i) = 0$ for every $1 \leq i \leq r$.*

Proof. The most interesting case is when $k = 3$, which is the one we will explain here:

By Lemma 4, we may assume that the chromatic classes are exactly $\{1, 2, 3\}$. Observe that there is a vertex $v \in Q$ such that its two neighbors are of the same color. For otherwise, i.e., if every vertex of Q has two neighbors with distinct colors, then we can easily check that Q has a periodic cyclic sequence of colors $1, 2, 3$, which is contrary to $\omega(Q) = 0$. See the left in Figure 1.

Now assume that all edges of Q are oriented as explained before. Let $v \in Q$ be a vertex with two neighbors $u, w \in Q$ of the same color, where u is the right neighbor of v, and w the left neighbor. Let $x \in Q$ be the right neighbor of u. Since Q is properly colored, x has a color distinct from those of u and w, and hence we can add an edge wx to create the properly colored quadrilateral $Q_1 = xuvw$. Now, let Q' be the cycle on the convex hull of $Q \setminus \{u, v\}$. We first observe that $\omega(Q_1) = 0$ since u and w have the same color. Secondly observe that $\omega(Q') = 0$, which can be explained as follows. Since u and w have the same color, the orientations of the two edges vw and uv

are canceled in the computation of $\omega(Q)$. Moreover, the edges ux and wx are both oriented away from x. Hence we get $\omega(Q) = \omega(Q') = 0$.

We can repeat these procedures inductively on Q', as shown to the right in Figure 1. That the total number of created quadrilaterals is $\frac{m-2}{2}$ follows from Lemma 3. □

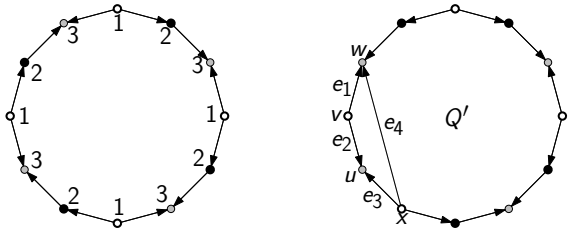


Figure 1:

The last result we need from [3] is the following:

Lemma 6 (S.Kato, R. Mori, A. Nakamoto) *Let $P = c_1 \cup c_2$ be a 2-colored n-point set in general position on the plane such that $|Conv(P)| = m$, where c_1 and c_2 are the color classes of P with $|c_1| \geq |c_2|$. Then there exists a set $\mathbb{S}$ of Steiner points such that $P \cup \mathbb{S}$ can be 2-quadrangulated, and:*

$$|\mathbb{S}| \leq \left\lfloor \frac{|c_1|}{3} \right\rfloor + \left\lfloor \frac{|c_2| - (m/2)}{2} \right\rfloor \leq \frac{5n}{12} - 1$$

The previous Lemma is essentially one of the main results of [2], and it is proven using exactly the same techniques as for Theorem 1 of [2], however, they are applied differently so the constant term on the bound of $|\mathbb{S}|$ is improved in the worst case from $(-1/3)$, in [2], to -1, in [3]. This negligible improvement of constants will play a useful role when proving Theorem 2.

The next Lemma is the last one before we proceed with the proof of Theorem 2.

Lemma 7 *Let $k \geq 2$ be an integer, and let P be a k-colored $(q+4)$-point set such that $|Conv(P)| = 4$. Then there exist two sets of Steiner points $\mathbb{S}_\Gamma$ and $\mathbb{S}_\Delta$ such that:*

- *$P \cup \mathbb{S}_\Gamma$ can be k-quadrangulated, and $|\mathbb{S}_\Gamma| \leq \frac{5q+8}{12}$.*
- *$P \cup \mathbb{S}_\Delta$ can be k-quadrangulated, and $|\mathbb{S}_\Delta| < \frac{(2k+1)q+16k}{6k}$.*

Proof. Let us divide the proof into two parts, one considering $\mathbb{S}_\Gamma$ and the other considering $\mathbb{S}_\Delta$. For simplicity, let us denote $Conv(P)$ by Q.

- Note that P can be regarded as a bichromatic point set as follows: if Q is bichromatic itself, say using colors c_1, c_2, then we can recolor every interior point of color different from c_2 with color c_1. We will rename the chromatic classes as $c_\alpha = c_1$ and $c_\beta = c_2$.

  If Q is 3-colored, say using colors c_1, c_2, c_3, then one color must appear twice on Q, say without loss of generality c_2. Proceed as before, recolor every point of color different than c_2 with a new color c_α. Rename the chromatic class c_2 as c_β.

  If Q is 4-colored, say using colors c_1, c_2, c_3, c_4, assume that c_1, c_3 and c_2, c_4 appear in diagonally opposite vertices of Q in clockwise order. Now recolor P with two new colors c_α and c_β as follows: every point of color c_2, c_4 receives color c_β. The rest of the points receive color c_α.

  As we end up having a bichromatic point set, using colors c_α, c_β, say without loss of generality that $|c_\beta| \leq |c_\alpha|$. Thus by Lemma 6 we obtain a quadrangulation of $P \cup \mathbb{S}_\Gamma$ such that:

$$|\mathbb{S}_\Gamma| \leq \left\lfloor \frac{|c_\alpha|}{3} \right\rfloor + \left\lfloor \frac{|c_\beta| - 2}{2} \right\rfloor \leq \frac{5|P|}{12} - 1 = \frac{5q+8}{12}$$

- Let us now do the following: say without loss of generality that c_1 is the smallest chromatic class among the k chromatic classes. Let us assume that Q is colored with colors other than c_1, we will see later on that this assumption only worsens the upper bound. Now let us introduce two Steiner points of color c_1 inside Q, very close to two opposite vertices of Q, and in such a way that we create a new quadrilateral Q' that is still properly colored and still contains the q interior points. Let P' be the point set formed by the vertices of Q' and the q points in its interior, see Figure 2.

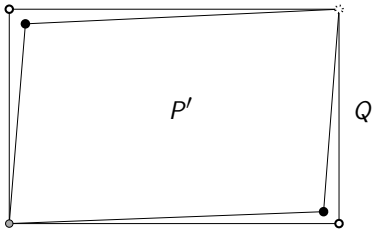


Figure 2: Points colored with color c_1 are represented in black. Quadrilateral Q' still contains the q interior points that quadrilateral Q originally contained.

  Now recolor every point of P' of color different than c_1 with a new color c. This leaves only two chromatic classes, c_1 and c, where c_1 is still the smallest one. We can now proceed with the quadrangulation of a bichromatic point set again,

this time obtaining:

$$\begin{aligned} |\mathbb{S}_\Delta| &\leq \left\lfloor \frac{|c|}{3} \right\rfloor + \left\lfloor \frac{(|c_1|+2)-2}{2} \right\rfloor + 2 \\ &\leq \frac{|c|}{3} + \frac{|c_1|}{2} + 2 = \frac{|c|+|c_1|}{3} + \frac{|c_1|}{6} + 2 \\ &< \frac{q+2}{3} + \frac{q}{6k} + 2 \\ &= \frac{q(2k+1)+16k}{6k} \end{aligned}$$

where the first inequality is obtained using Lemma 6 again. The last inequality is obtained by $|c| = q + 2 - |c_1|$ and the assumption that c_1 is the smallest chromatic class, so $|c_1| < \frac{q}{k}$. If $|c_1| = \frac{q}{k}$, then we have $|c_1| = \cdots = |c_k| = \frac{q}{k}$, and hence we can take c_1 so that c_1 does appear on Q. In this case, only at most one Steiner point is required in the beginning to obtain Q'. Hence we would obtain $|\mathbb{S}_\Delta| \leq \frac{q(2k+1)+7k}{6k}$ which is slightly smaller, but it would still play a role reducing the bound on Theorem 2.

□

We are finally ready to prove Theorem 2:

Proof. Let P be a k-colored n-point set on the plane, where $|Conv(P)| = m$ and $q = n - m$. Then P has q interior points. If $\omega(P) = 0$, by Lemma 5, we know that we can partition $Conv(P)$ into $r = \frac{m-2}{2}$ convex quadrilaterals Q_i, $1 \leq i \leq r$, each of which is properly colored and has $\omega(Q_i) = 0$. If $\omega(P) \neq 0$, then by Theorem 1 the only case that makes senses is $k \geq 4$. That is, P is colored with at least four colors but only three of them appear in $Conv(P)$, causing $\omega(P) \neq 0$. In this case we cannot apply Lemma 5 directly, so we will introduce one Steiner point s inside $Conv(P)$, and very close to one vertex v of $Conv(P)$ such that s replaces v in $Conv(P)$. If the color of s is chosen such that the new $Conv(P)$ is 4-colored, and observe that this is always the case, we can proceed with Lemma 5 as before.

Let q_i be the number of interior points in quadrilateral Q_i. Using the first case of Lemma 7 on each Q_i, we get overall a set $\mathbb{S}_\Gamma = \mathbb{S}_\Gamma^1 \cup \mathbb{S}_\Gamma^2 \cup \cdots \cup \mathbb{S}_\Gamma^r$ of Steiner points, where $\mathbb{S}_\Gamma^i$ denotes the set of Steiner points used to k-quadrangulate Q_i such that:

$$\begin{aligned} |\mathbb{S}_\Gamma| &= \sum_{i=1}^{r} |\mathbb{S}_\Gamma^i| \leq \sum_{i=1}^{r} \frac{5q_i+8}{12} = \frac{2r}{3} + \sum_{i=1}^{r} \frac{5q_i}{12} \\ &= \frac{m-2}{3} + \frac{5q}{12} \end{aligned}$$

Now, if we use the second case of Lemma 7 on each Q_i we get overall:

$$\begin{aligned} |\mathbb{S}_\Delta| &= \sum_{i=1}^{r} |\mathbb{S}_\Delta^i| < \sum_{i=1}^{r} \frac{(2k+1)q_i+16k}{6k} \\ &= \frac{8r}{3} + \sum_{i=1}^{r} \frac{(2k+1)q_i}{6k} \\ &= \frac{4(m-2)}{3} + \frac{(2k+1)q}{6k} \end{aligned}$$

We are assuming that in each Q_i all the chromatic classes appear. If that is not the case, say there is at least one chromatic class not appearing in some Q_j, $1 \leq j \leq r$, then the size of the smallest chromatic class in Q_j is 0. In such a case, as the reader can verify, we would obtain an improvement on $|\mathbb{S}_\Delta^j|$, which would clearly improve $|\mathbb{S}_\Delta|$.

Now we would like to see which one of $\mathbb{S}_\Gamma, \mathbb{S}_\Delta$ performs better, and under what circumstances. For the following, we note that $q = n - m$. If $|\mathbb{S}_\Gamma| \leq |\mathbb{S}_\Delta|$, then we have:

$$\frac{m-2}{3} + \frac{5(n-m)}{12} < \frac{4(m-2)}{3} + \frac{(2k+1)(n-m)}{6k}$$

and hence $m > \frac{k(n+24)-2n}{13k-2}$. The bound on m in turn implies $q < \frac{12k(n-2)}{13k-2}$.

Let $\mathbb{S}$ be a set of Steiner points added for k-quadrangulating P, and estimate $|\mathbb{S}|$ by $\min\{|\mathbb{S}_\Gamma|, |\mathbb{S}_\Delta|\}$. Then, if $m > \frac{k(n+24)-2n}{13k-2}$, we obtain:

$$\begin{aligned} |\mathbb{S}| &\leq |\mathbb{S}_\Gamma| + 1 = \frac{m-2}{3} + \frac{5q}{12} + 1 = \frac{4n+q+4}{12} \\ &< \frac{(16k-2)n+7k-2}{39k-6} \end{aligned}$$

where the second equality follows from $m = n - q$. On the other hand, if $m \leq \frac{k(n+24)-2n}{13k-2}$, then

$$|\mathbb{S}| \leq |\mathbb{S}_\Delta| + 1 < \frac{(16k-2)n+7k-2}{39k-6}$$

The Theorem now follows entirely. □

References

[1] C. Cortés, A. Márquez, A. Nakamoto, J. Valenzula. *Quadrangulations and 2-colorations.* 21st European Workshop on Computational Geometry, Eindhoven, March 911, 6568, 2005.

[2] V. Alvarez, T. Sakai, J. Urrutia, *Bichromatic Quadrangulations with Steiner Points*, Graph. Comb., **23**(1):85–98, 2007.

[3] S. Kato, R. Mori, A. Nakamoto, *Quadrangulations on 3-colored point sets with Steiner points and their winding numbers*, Submitted. Preliminary version in Proc. XIV Spanish Meeting on Computational Geometry, 133–136, 2011

Extremal problems in colored point sets in the plane

Viola Mészáros*

Abstract

The goal of this paper is to give an overview of the results and open questions in the area of long alternating paths and separated matchings. We include a new proof of the construction class of Hajnal and Mészáros showing there are at most $\frac{4}{3}n + O(\sqrt{n})$ points in any separated matching and on any noncrossing alternating path, respectively, in a specific convex $2n$-element equally two-colored planar point set. This is currently the best upper bound for alternating paths and it is conjectured to be asymptotically tight. The survey is written based on my Ph.D. thesis [13].

1 Introduction

One of the basic problems in geometric graph theory is to decide if a given graph can be drawn on a given planar point set by pairwise noncrossing straight line edges. It is an easy observation that any planar point set in general position admits a noncrossing Hamiltonian path. In a more demanding version of the problem, the points and the vertices of the graph are colored and each vertex has to be placed in a point of the same color (see [8] for references). Interesting and not so easy questions arise already if the points are colored by two colors and we want to embed a path where the colors are alternating. Several authors have focused on this question recently and obtained appealing results.

Consider an arbitrary $2n$-element point set in the plane colored by two colors in a balanced way. We require n points to be red and n points to be blue. A point set with this property is *equicolored*. We would like to determine or estimate the number of points on the longest noncrossing path such that edges join points of different color and are straight line segments. We are talking about combinatorial length here, that is, the *length* of a path is determined by the number of its points. We call a path with edges connecting points of different color *alternating*.

In case our point set is in general position the following results were gained. Abellanas et al. showed in 1999 that if the color classes are separated by a line, there is a noncrossing, alternating Hamiltonian path on the point set [1]. The same holds if the set of vertices of the convex hull coincides with one of the color classes [1]. If the color classes are not separated by a line, there are colored point sets with no noncrossing, alternating Hamiltonian path for $n \geq 8$, even if the points are in convex position.

The existence of halving lines and the result on the separated color classes in [1] together guarantee that for any equicolored point set of $2n$ points the longest noncrossing, alternating path contains at least n points.

Erdős [6] posed the convex version of the problem already in the eighties. It inspired numerous researchers. In the following we state their results.

$$\ell(\mathcal{P}) = \max_{U \text{ is a noncrossing alternating path}} \ell(U),$$

where $\ell(U)$ is the number of points on U.

$$\ell(n) = \min_{\mathcal{P} \text{ is equicolored}} \ell(\mathcal{P}),$$

where $\mathcal{P}$ is any colored planar $2n$-element convex point set.

When the points are in convex position without loss of generality we may assume that they are on a circle. Erdős conjectured that the following configuration was asymptotically extremal. Let n be divisible by four. Divide the circle into four intervals that consist of $\frac{n}{2}$ red, $\frac{n}{4}$ blue, $\frac{n}{2}$ red and $\frac{3n}{4}$ blue points, respectively. It is not hard to check that in this configuration there are $\frac{3n}{2} + 2$ points on the longest noncrossing, alternating path.

Kynčl, Pach and Tóth [10] disproved Erdős' conjecture by constructing a single configuration in 2008 and showed the $\frac{4}{3}n + O(\sqrt{n})$ upper bound. In the same paper they also proved a lower bound of $n + \Omega\left(\sqrt{n/\log n}\right)$ points. Abellanas et al. independently of the previously mentioned researchers constructed a very similar configuration for the same upper bound [2]. The upper bound is conjectured to be asymptotically tight.

We improved the lower bound of [10] with Hajnal. We showed that there are at least $n + \Omega(\sqrt{n})$ points on the longest noncrossing alternating path [7]. The proof of the lower bound in [10] is based on arcs which contain significantly more points from one color class than from the other. We do the same but using a completely different idea that gives us a better result. Regarding the upper bound we presented a class of

*Institute for Mathematics, Technical University of Berlin, meszaros@math.tu-berlin.de. Work on this paper was partially supported by ESF EuroGiga project ComPoSe (IP03), by OTKA Grant K76099 and by OTKA Grant 102029.

configurations that allows at most $\frac{4}{3}n+O(\sqrt{n})$ points on any noncrossing alternating path [7]. Jan Kynčl also found this class by computer search. Our class differs from the previous constructions for it allows two intervals with alternating short monochromatic arcs while the previous ones allowed only one.

The proof techniques in the research of alternating paths introduced the notion of separated matchings. *Separated matchings* are matchings that have the following properties: there is a line that crosses all edges of the matching and no two edges cross geometrically in the matching. We call the line crossing all edges of the matching the *axe*. Separated matchings gained an importance recently because they have advantages compared to alternating paths. It is a new rising topic with room for results. Separated matchings are simpler than alternating paths but give a building element of them. If we consider examples where the difference in cardinality of the color classes is bounded on any interval (that is, the *discrepancy* of the point set is small) that alone guarantees a long noncrossing, alternating path. This is a consequence of the frequent alternations of colors along the circle. Regarding separated matchings small discrepancy coloring is a case that we should consider. It could lead to a deeper understanding of the problem.

Recently several new constructions were gained allowing at most $\frac{4}{3}n + O(\sqrt{n})$ points in any separated matching. There is a class of configurations [14] that differs from all known previous constructions in a considerable feature. It contains arbitrary many intervals of alternating short monochromatic arcs while beforehand all constructions contained at most two of such arcs. A type of coloring was presented so that among these colorings in the optimal one any separated matching contains at most $\frac{4}{3}n + O(\sqrt{n})$ points [12, 15]. Regarding the small discrepancy colorings it was shown that for any coloring with discrepancy $d \leq 3$ there is a separated matching containing at least $\frac{4n}{3}$ points [14]. It would be feasible to extend and strengthen this results with new ideas.

Finally, we mention that in case we perform a change in the construction of [10] we can improve the upper bound for separated matchings to $\frac{4}{3}n + O(1)$. This contributes to the separated matching conjecture stated in [10]: If $2k$ denotes the number of alternations among the two colors in a $2n$-element point set on the circle, then for any fixed k and large n, any configuration admits a separated matching that contains at least $\frac{2k-1}{3k-2}2n + o(n)$ points. On the other hand, making this modification in the construction of [10] does not improve the result for alternating paths.

Returning to the general setting, Kaneko et al. [9] considered point sets with odd cardinality in 2004. *An equitable* coloring means that the sizes of the two color classes differ by at most one. They proved that any equitably colored set of at most 12 points or of 14 points admits a noncrossing, alternating Hamiltonian path. On the other hand, Kaneko et al. [9] gave examples of equitably colored sets of n points allowing no noncrossing, alternating Hamiltonian path for any $n > 12$, $n \neq 14$.

We also considered equitably colored point sets with Cibulka, Kynčl, Stolař and Valtr in 2009. Our points were on a double-chain defined in the following way. A *convex* or a *concave chain* is a finite set of points in the plane lying on the graph of a strictly convex or a strictly concave function, respectively. A *double-chain* consists of a convex chain and a concave chain such that any line determined by any of the chains does not intersect the other chain. We proved [4] if both chains of the double-chain contain at least one fifth of all the points, then there exists a noncrossing, alternating Hamiltonian path in the point set. But the above property does not hold if one of the chains contains at most $\approx 1/29$ of all the points [4]. This result is so far the strongest evidence by it that the convex setting of our point set might be an extremal case. Otherwise, it is not clear what role convexity plays in the problem.

2 The new proof

Now we give an alternative proof for the construction class described in [7]. It shows the upper bound of $\frac{4}{3}n+O(\sqrt{n})$ points for noncrossing alternating paths. This bound was known before (see in [10]) still the class in [7] is a contribution to the area of long noncrossing alternating paths for the reason that it is a class. Previously only single constructions were presented. It also differs from the earlier configurations significantly.

First we introduce a few definitions. Our $2n-$*element* equicolored point set is on the circle. Therefore, there is a natural ordering among the points. We call a maximal monochromatic arc a *run*. Let $M_{r\times\ell}$ denote r many consecutive runs alternating in color where each run contains ℓ points. A building block of this type $M_{r\times\ell}$ is a *mixed run*. Note, in the following when we use the notion of run we do not refer to the runs belonging to a mixed run (if not stated otherwise) but to the other runs. Thus, we distinguish these two types of building elements. Let B_L and R_L denote a blue run and a red run of length L, respectively. Then we set $\alpha \in [-1, 1]$, and finally we introduce $\mathcal{P}_{\alpha,\ell}$:

$$\begin{array}{ccccccc} B_{2L}, & R_{(1+\alpha)L}, & M_{r\times\ell}, & R_{(1+\alpha)L}, & B_{2L}, \\ & R_{(1-\alpha)L}, & M_{r'\times\ell}, & R_{(1-\alpha)L}, \end{array}$$

where ℓ is arbitrary, r, r' are divisible by two and the following equalities hold $r\ell = (2-2\alpha)L$ and $r'\ell = (2+2\alpha)L$. Therefore, $\mathcal{P}_{\alpha,\ell}$ is an equicolored point set of $2n = 12L$ points. We assume that αL is an

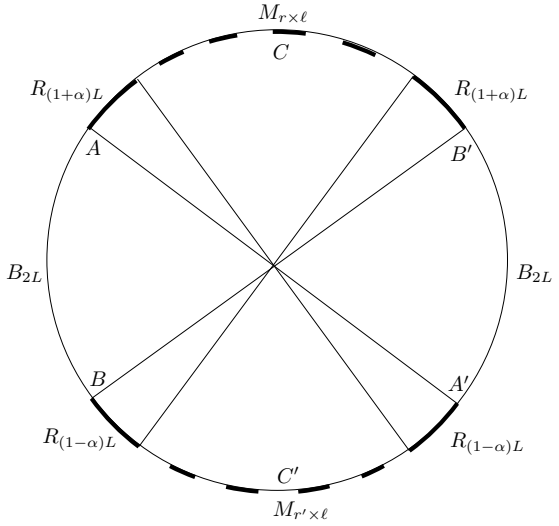


Figure 1: The construction

integer. In case $\alpha = -1, 1$ our construction coincides with the [10] construction if we set $\ell = \Theta(\sqrt{L})$.

Let $m(\mathcal{P})$ denote the number of points in a maximal separated matching on the point set $\mathcal{P}$. Now we are ready to state our theorem.

Theorem 1 $m(\mathcal{P}_{\alpha,\ell}) \leq \frac{4n}{3} + O(\ell)$.

Proof. We assume that the axe of an optimal matching divides the point set into an upper and a lower side and the matched upper and lower points are ordered from left to right. We can partition the edges of the matching into classes in such a way that on each side (upper, lower) the endpoints belong to one run or one mixed run. On each side we complete the points of a class to an arc by adding the possible intermediate points. We call these pairs of arcs *blocks*. We have constant number of blocks.

If in one of these blocks a mixed coloring is to be matched with a run, then at most half of the mixed coloring is going to participate in the matching. To be precise this is the case if the block contains an even number of alternating short runs from the mixed run. If we throw away $O(\ell)$ many matched edges we can assume that our blocks have an even number of alternating short runs from the mixed runs. The resulting separated matching we denote by M.

We consider a few different cases based on the position of the axe. If an end of the axe cuts a mixed run or it is placed between a run and a mixed run, we say that the end of the axe is *mixed*.

1st case: *Both ends of the axe are mixed.* We will change the position of the axe which will modify M. The new axe will be CC' that halves both mixed runs. In the new matching M' the two mixed runs will be completely matched. Around C and C' each mixed run is matched with itself. Observe, that the edges of M that do not involve points from any mixed run will intersect CC' but cannot cross any edge we introduced in M'. Therefore we add these edges to M'.

We will now argue that we did not decrease the size of M to get M'. As all edges of M that do not contain points from mixed runs are present in M', we only need to take care of those edges that have an endpoint in some mixed run.

If there is an edge going between the two mixed runs in M, then there can be no more points covered by M than the points of the mixed runs (see the axe) which is at most $4L = \frac{2n}{3}$ points. Consequently, we have that $|M'| \geq |M|$.

So we may assume that edges of M that have an endpoint in a mixed run have their other endpoint in a run or in the same mixed run. By a previous observation when a mixed coloring is matched to a run, then half of the mixed part is not going to participate in the matching. Therefore when in M' we match the mixed run with itself, we get at least as many points in the matching as in M.

After the previous surgery the bound on the number of matched points is straightforward. There are at most $2((1-\alpha)L + (1+\alpha)L) = 4L$ further points in M'. Hence, we get $8L = \frac{4n}{3}$ points altogether in M'.

2nd case: *None of the ends is mixed.* Observe, that the ends of the axe come from the set of $\{A, A', B, B'\}$. If the axe cuts a run to two parts P_1 and P_2, then one of P_1 and P_2 will not contain points from M. So we can shift the axe to be between two runs in a way that M does not change.

If there are at most $\frac{2n}{3}$ points on one side of the axe, then we are done. Hence, by symmetry argument we can assume that the axe is AA'. We may assume that at each end of the axe the shorter run is fully matched in the most economical way (to the other neighboring run to the axe). This gives $4L$ points to M.

On the lower side the mixed run of $(2 + 2\alpha)L$, a red run of $(1 - \alpha)L$ and a blue subrun of $(1 - \alpha)L$ points remain. On the upper side a blue subrun of $(1 + \alpha)L$, a red run of $(1 + \alpha)L$ and a mixed run of $(2 - 2\alpha)L$ points remain. We can match at most another $(2 + 2\alpha)L + (2 - 2\alpha)L = 4L$ points. Hence, we showed the desired bound of $8L = \frac{4n}{3}$ points.

3rd case: *One of the ends is mixed, the other is not.* By a previous observation if the axe cuts a run, we shift the axe to be between two runs so that M is not modified. If this end of the axe gets beside a mixed run, we use the argument in the *1st case*.

If there are at most $\frac{2n}{3}$ points on one side of the axe, we are done. Therefore, we claim by a symmetry argument that if the axe is in the lower (upper) mixed run, then the other end of the axe is at A (at B). If a point from the upper mixed run is matched to a point of the lower mixed run, then the size of M is at most $(2-2\alpha)L+(2+2\alpha)L+2(1\pm\alpha)L = 6L \pm 2\alpha L \leq 8L = \frac{4n}{3}$ points.

If there is no edge in M with endpoints from different mixed runs, then the surgery as in *1st case* can be performed. The end of the axe in the mixed run will get to the middle of the mixed run to C or C'. Notice, that one side of the axe contains at most $2n/3$ many points. Hence, the statement follows. □

If we set $\ell = \Theta(\sqrt{n})$, it is straightforward that there are at most $\frac{4n}{3} + O(\sqrt{n})$ points on the longest noncrossing alternating path. Consider the following argument. Any noncrossing alternating path consists of a separated matching and additional edges. We call these additional edges *side edges* for the reason that a line containing a side edge has the path on one of its sides. If $\ell = \Theta(\sqrt{n})$, the number of side edges is restricted to $O(\sqrt{n})$. Hence, together with Theorem 1 this concludes the case of noncrossing alternating paths and we get:

If $\ell = \Theta(\sqrt{n})$, then $\ell(\mathcal{P}_{\alpha,\ell}) = \frac{4n}{3} + O(\sqrt{n})$.

3 Conclusion

Numerous open problems remain in the area of noncrossing alternating paths and separated matchings. Regarding alternating paths the general case could yield interesting results. Researchers were mostly focusing on point sets in convex position lately.

In a bit more restricted setting one could consider two convex chains instead of the double-chain. Obviously, there would be less freedom in drawing a noncrossing alternating path than on a double-chain. But as there would be more freedom than in the convex case it is of interest and would mean a case "in between".

The lower bound $n+\Omega(\sqrt{n})$ for alternating paths is also worth to consider. By new ideas there could be an improvement. The gap is not so small till the upper bound $\frac{4n}{3} + O(\sqrt{n})$ which is believed to be asymptotically tight.

Separated matchings is a developing line of research. It rose from the area of alternating paths and turned out to be important. Investigating small discrepancy colorings promises new results. It could be a huge step forward in understanding the feature of the original Erdős problem.

Conjecture. Every equicoloring of $2n$ points in convex position in the plane admits a separated matching of size $\frac{4}{3}n$.

Acknowledgments

I would like to thank my supervisor Péter Hajnal for all his help in the research of separated matchings and alternating paths.

References

[1] M. Abellanas, J. García, G. Hernández, M. Noy and P. Ramos. *Bipartite embeddings of trees in the plane.* Discrete Appl. Math. 93 (1999), 141–148.

[2] M. Abellanas, A. García, F. Hurtado and J. Tejel. *Caminos alternantes.* X Encuentros de Geometria Computacional (in Spanish), Sevilla, 2003, 7–12.

[3] P. Brass, W. Moser and J. Pach. *Research Problems in Discrete Geometry.* Springer, New York, 2005.

[4] J. Cibulka, J. Kynčl, V. Mészáros, R. Stolař, P. Valtr. *Hamiltonian alternating paths on bicolored double-chains* I. G. Tollis, M. Patrignani (Eds.), Graph Drawing 2008, Lecture Notes in Computer Science 5417, Springer, New York, 2009, pp. 181–192.

[5] J. Cibulka, J. Kynčl, V. Mészáros, R. Stolař, P. Valtr. *Universal sets for straight-line embeddings of bicolored graphs.* To appear in Thirty Essays on Geometric Graph Theory (J. Pach editor), Springer.

[6] P. Erdős, Personal communication to J. Pach (see [10]).

[7] P. Hajnal, V. Mészáros, *A note on noncrossing path in colored convex sets.* To appear in Discrete Mathematics and Theoretical Computer Science.

[8] A. Kaneko and M. Kano. *Discrete geometry on red and blue poins in the plane — a survey.* Discrete and Computational Geometry (B. Aronov et al., eds.), Springer-Verlag, Berlin, 2004, 551–570.

[9] A. Kaneko, M. Kano and K. Suzuki. *Path coverings of two sets of points in the plane.* J. Pach (Ed.), Towards a Theory of Geometric Graphs, Contemporary Mathematics 342 (2004), 99–111.

[10] J. Kynčl, J. Pach and G. Tóth. *Long alternating paths in bicolored point sets.* in: Graph Drawing (J. Pach, ed.), Lecture Notes in Computer Science 3383, Springer-Verlag, Berlin, 2004, 340-348. Also in: Discrete Mathematics 308 (2008), 4315–4322.

[11] C. Merino, G. Salazar and J. Urrutia. *On the length of the longest alternating path for multicoloured point sets in convex position.* Discrete Mathematics Vol. 360, no. 15, pp. 1791–1797, 2006.

[12] V. Mészáros. *An upper bound on the size of separated matchings.* Electronic Notes in Discrete Mathematics (2011), pp. 633-638.

[13] V. Mészáros. *Extremal problems on planar point sets.* Ph.D. thesis, University of Szeged, Hungary, 2011.

[14] V. Mészáros. *Separated matchings and small discrepancy colorings.* Submitted.

[15] V. Mészáros. *Separated matchings on colored convex sets.* Manuscript.

[16] Richard P. Stanley. *Enumerative Combinatorics, Vol. II.* Cambridge University Press, 1999.

Compatible Matchings for Bichromatic Plane Straight-line Graphs

Oswin Aichholzer* Ferran Hurtado† Birgit Vogtenhuber*

Abstract

Two plane graphs with the same vertex set are compatible if their union is again a plane graph. We consider bichromatic plane straight-line graphs with vertex set S consisting of the same number of red and blue points, and (perfect) matchings which are compatible to them. For several different classes $\mathcal{C}$ of graphs, we present lower and upper bounds such that any given graph $G(S) \in \mathcal{C}$ admits a compatible (perfect) matching with this many disjoint edges.

1 Introduction

We consider bichromatic point sets $S = R \cup B$ where the red set R and the blue set B have the same cardinality n. An edge spanned by two points of S is called *bichromatic*, if it has one red and one blue endpoint. A graph $G(S)$ is called bichromatic, if all its edges are bichromatic. Accordingly, an edge where both endpoints have the same color is called *monochromatic*, and a graph is called monochromatic if all its vertices have the same color. Two plane graphs with the same vertex set S are called *compatible* if their union is again a plane graph and *disjoint* if their intersection does not contain any edge. Graph compatibility and graph augmentation questions are directly related to network design and graph drawing problems; see there for potential applications.

In the following, we solely consider plane straight-line graphs, and refer to them just as graphs for the sake of brevity.

There exist several results on compatible graphs, for bichromatic as well as for uncolored point sets. For example, Ishaque et al. [5] recently showed that any (uncolored) geometric matching with an even number of edges admits a disjoint compatible matching. In a similar direction, Abellanas et al. [2] showed upper and lower bounds for how many edges a compatible matching for a graph of a certain class can admit.

In a different work, Abellanas et al. [1] showed how many edges are needed at least to augment an (uncolored) connected graph to a 2-vertex or 2-edge connected graph. According results on bichromatic graphs have been obtained by Hurtado et al. [4], who also considered the question of augmenting a (disconnected) bichromatic graph to be connected. Among others, they showed how to connect a bichromatic perfect matching to a (bichromatic) spanning tree. Hoffmann and Tóth [3] extended this work to spanning trees with maximum vertex degree three.

In this work we investigate the following question. Given a bichromatic graph $G(S)$, we want to find a matching $M(S)$ (of some type) that is compatible with $G(S)$. Following the lines of [2], we call such a matching $G(S)$-compatible. Similarly, if $M(S)$ is disjoint from $G(S)$, we also say that it is $G(S)$-disjoint. We consider two classes of $G(S)$-compatible matchings: (1) bichromatic $G(S)$-disjoint / perfect matchings (Section 2) and (2) monochromatic matchings (Section 3).

For a $G(S)$-compatible matching $M(S)$, we denote the number of edges in $M(S)$ that are disjoint from $G(S)$ by $d(G(S), M(S))$. Similar to the work in [2], we focus on bounds for this number for the considered classes of matchings and the graph classes of spanning trees (*tree*), spanning paths (*path*), spanning cycles (*cycle*), and perfect matchings (*match*).

A preliminary version of this work can be found in [7], Section 3.3.

2 Bichromatic matchings

We start with bichromatic (perfect) matchings which are compatible to a given bichromatic graph. To simplify reading, we mostly omit the attribute bichromatic in this section.

It is well known that every set S with $|R| = |B|$ admits a bichromatic perfect matching [6]. On the other hand, there exists a large class of point sets with $|R| = |B|$, for which there exists only one such matching $M(S)$. Note that for any plane graph $G(S)$ that is obtained by adding edges to $M(S)$, we get $d(G(S), M(S)) = 0$. Due to these observations, and to avoid trivial bounds, we restrict considerations in this section to point sets admitting strictly more than one bichromatic perfect matching.

Let $\mathcal{S}$ be the class of point sets admitting at least two different bichromatic perfect matchings, and let $\mathcal{S}_n \subset \mathcal{S}$ be the sets with $|R| = |B| = n$. For a given class $\mathcal{C}$ of graphs, we denote by

$$b_{\mathcal{C}}(n) = \min_{S \in \mathcal{S}_n} \min_{G(S) \in \mathcal{C}} \max_{M(S)} d(G(S), M(S))$$

*Institute for Software Technology, Graz University of Technology, [oaich|bvogt]@ist.tugraz.at

†Departament de Matemàtica Aplicada IV, Universitat Politècnica de Catalunya (UPC), Ferran.Hurtado@upc.edu

the maximum number such that for every point set $S \in \mathcal{S}_n$ and every graph $G(S) \in \mathcal{C}$, we can find a *bichromatic disjoint compatible matching* $M(S)$ of cardinality at least $b_{\mathcal{C}}(n)$. Accordingly, we denote by

$$b^p_{\mathcal{C}}(n) = \min_{S\in\mathcal{S}_n} \min_{G(S)\in\mathcal{C}} \max_{M(S)} d(G(S), M(S))$$

the maximum number such that for every point set $S \in \mathcal{S}_n$ and every graph $G(S) \in \mathcal{C}$, there exists a *bichromatic compatible perfect matching* $M(S)$ with (at least) $b^p_{\mathcal{C}}(n)$ edges disjoint from $G(S)$.

Note that $b_{\mathcal{C}}(n) \geq b^p_{\mathcal{C}}(n)$, as any compatible perfect matching $M(S)$ for a given graph $G(S)$ contains a $G(S)$-disjoint matching $M'(S)$ of size $d(G(S), M'(S)) = d(G(S), M(S))$.

We start with the class of perfect matchings, and with disjoint compatible matchings for them.

Theorem 1 $\lceil \frac{n-1}{2} \rceil \leq b_{match}(n) \leq \frac{3n}{4}$.

Proof. Consider a perfect matching $PM(S)$. Using the result of Hoffmann and Tóth [3], we augment $PM(S)$ to a bichromatic spanning tree $T(S)$ with maximum vertex degree three; see Figure 1.

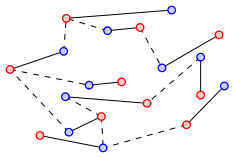


Figure 1: A perfect matching (solid edges) augmented to a spanning tree (augmenting edges are dashed).

Now consider the graph $A(S) = T(S) \backslash PM(S)$ of the augmenting edges. $A(S)$ contains exactly $n-1$ edges. Further, as every vertex is incident to exactly one edge of $PM(S)$, the maximum vertex degree in $A(S)$ is at most two. In other words, $A(S)$ is a collection of paths $\mathcal{P}$ and isolated vertices. Every path P with k_P+1 vertices has k_P edges, of which $\lceil \frac{k_P}{2} \rceil$ form a matching. Thus, $A(S)$ contains a matching with $\sum_{P\in\mathcal{P}} \lceil \frac{k_P}{2} \rceil \geq \lceil \frac{n-1}{2} \rceil$ edges, yielding a lower bound of $b_{match}(n) \geq \lceil \frac{n-1}{2} \rceil$ for the number of edges in a maximum $PM(S)$-disjoint compatible matching.

For an upper bound on $b_{match}(n)$ consider the perfect matching $PM(S)$ shown in Figure 2. Every red vertex that is incident to one of the small edges inside a triangle does not "see" any blue vertex except for the one it is matched to. Thus, in any $PM(S)$-disjoint compatible matching $M(S)$ all these red vertices must stay unmatched, implying an upper bound of $b_{match}(n) \leq d(PM(S), M(S)) \leq \frac{3n}{4}$. □

The upper bound for disjoint matchings (induced by Figure 2) directly implies an according upper bound of $b^p_{match}(n) \leq \frac{3n}{4}$ for perfect matchings that

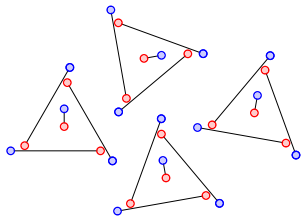


Figure 2: An upper bound example for $b_{match}(n)$.

are compatible to a given perfect matching. But for this case we can say even more.

Theorem 2 $b^p_{match}(n) \leq \frac{n}{2}$.

Proof. Consider the perfect matching $PM(S)$ illustrated in Figure 3. Every vertex that is incident to one of the short edges can only be compatibly matched in two ways; either to the other vertex of the edge it is incident to, or to the accordingly colored vertex of the long edge next to it. As matching one vertex of a short edge to the according vertex of the long edge next to it would force the other vertex of the short edge to stay unmatched, any perfect matching $M(S)$ that is compatible with $PM(S)$ must contain all short edges of $PM(S)$. □

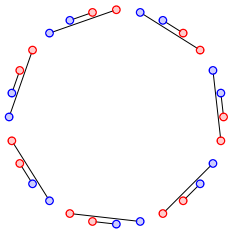


Figure 3: An upper bound example for $b^p_{match}(n)$.

For the classes of spanning trees, spanning cycles, and spanning paths, the examples illustrated in Figure 4 imply bounds of $b_{tree}(n) = b_{cycle}(n) = 0$ and $b_{path}(n) \leq 1$.

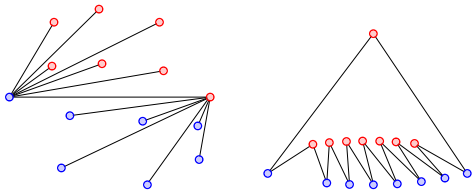


Figure 4: Upper bound examples for $b_{tree}(n)$ and $b_{cycle}(n)$.

3 Monochromatic matchings

We continue with monochromatic compatible matchings for bichromatic graphs. In the following, we denote by

$$m_{\mathcal{C}}(n) = \min_{|S|=2n} \min_{G(S)\in\mathcal{C}} \max_{M(S)} d(G(S), M(S))$$

the maximum number such that for every graph $G(S) \in \mathcal{C}$ there exists a *monochromatic compatible matching* $M(S)$ with (at least) $m_{\mathcal{C}}(n)$ edges.

We again start with the class of bichromatic perfect matchings, and with monochromatic compatible matchings for them.

Theorem 3 $\frac{n}{4} \leq m_{match}(n) \leq \frac{5n}{12}$.

Proof. Consider a perfect matching $PM(S)$. Assume w.l.o.g. that $PM(S)$ does not contain any vertical edge, and that for at least half of the edges, the left vertex is red. We augment the matching to a weakly simple polygon in the following way. First, we add a bounding box around the $PM(S)$. Next, we extend all edges with a left red vertex to the right until it hits the bounding box or (an extension of) an edge. Then we extend all other edges to the right as well, but with a slight turn. The result is a so-called weakly simple polygon, which can be transformed to a simple polygon by slightly "inflating" the edges; see Figure 5 (left). In the resulting polygon, all left endpoints of edges and all red right endpoints of edges appear as reflex vertices. All blue right endpoints of matching edges are "hidden" in the exterior of the polygon.

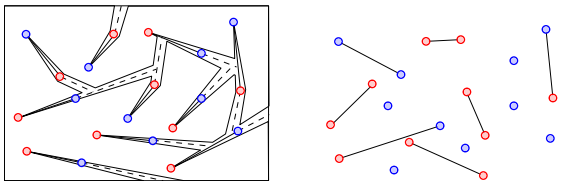


Figure 5: (left) Transforming a perfect matching to a simple polygon. (right) A resulting compatible matching.

Abellanas et al. [1] showed that for every simple polygon $P(V)$ with vertex set V and every subset $V' \subseteq V$ containing all reflex vertices of $P(V)$, there exists a perfect matching of the vertices V' where no edge is outside the boundary of $P(V)$. Applying this result to the set of reflex vertices of the constructed polygon, we obtain a matching $M(S)$ with at most $\frac{n}{2}$ bichromatic edges and thus $M(S)$ has at least $\frac{n}{4}$ red edges, implying $m_{match}(n) \geq \frac{n}{4}$. Figure 5 (right) shows a possible resulting matching.

For an upper bound on the number of edges in a monochromatic matching, we can recycle the idea from Figure 2. Inverting the colors of every second triangle construction and combining them to a closed cycle, we obtain a perfect matching $PM(S)$ where every sixth point of each color must remain unmatched in any monochromatic $PM(S)$-compatible matching; see Figure 6 for a schematic illustration. □

The principle of the lower bound part of the above proof can be reused to provide a lower bound for the

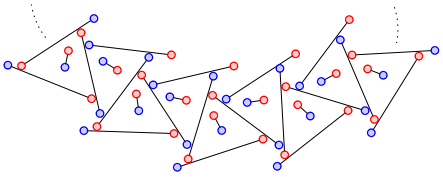


Figure 6: Scheme for a bichromatic perfect matching where any monochromatic compatible matching has at most $\frac{5n}{12}$ edges.

size of maximum monochromatic compatible matchings for trees, in dependence of the number of interior vertices (of one color) of the tree. The basic idea for this bound was developed during a research week which was also the starting point for the work [2].

Theorem 4 *Let $T(S)$ be a bichromatic tree $T(S)$ with i_r interior (non-leaf) red vertices. There exists a red matching $M(S)$ with $d(T(S), M(S)) \geq \lceil \frac{i_r - 1}{6} \rceil$.*

Proof. We generate a simple polygon for $T(S)$ by adding a bounding box, connecting $T(S)$ to the box and then inflating the whole construction. Every vertex v of $T(S)$ with vertex degree $d(v)$ corresponds to $d(v)$ vertices in the polygon P, at most one of them being reflex. We choose one of these $d(v)$ vertices for each vertex v of $T(S)$, (if v corresponds to a reflex vertex, we choose that one). Additionally, we choose a second vertex for each of the i_r red interior vertices; see Figure 7 (left) for a resulting polygon.

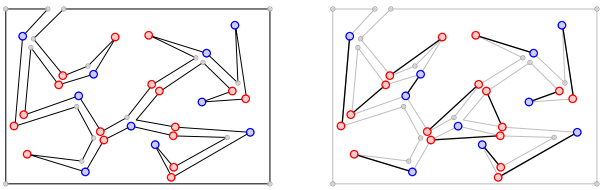


Figure 7: (left) A simple polygon generated from a bichromatic tree (non-selected vertices and vertices on the bounding box are drawn gray). (right) A (nearly) perfect matching of the selected vertex set.

Applying the result of [1] to the selected vertices, we obtain a (nearly) perfect matching with $\lfloor \frac{2n + i_r}{2} \rfloor$ edges, at least $\lfloor \frac{i_r}{2} \rfloor$ of them red. But it might happen that both red vertices corresponding to one red vertex in S are incident to such a red edge. Also, there might occur cycles of such red edges; see Figure 7 (right).

In general, the translated red edges form a set of cycles and paths. For such a path of length k, $\lceil \frac{k}{2} \rceil$ edges can simultaneously occur in a monochromatic $T(S)$-compatible matching. For a red cycle of length k, $\lfloor \frac{k}{2} \rfloor$ of these edges can be used. Assuming the worst case where (nearly) all red edges form 3-cycles, we obtain $\lceil \frac{i_r - 1}{6} \rceil$ edges for a red $T(S)$-compatible matching. □

Applying Theorem 4 to the class of spanning paths, we obtain

Corollary 5 $m_{path} \geq \lceil \frac{n-2}{6} \rceil$.

Note that in a path all vertices have degree at most two. Thus, a cycle of odd length among the red edges (translated back to S) can only occur if the path ends inside this cycle. As there might be a sequence of cycles $C_1, \ldots, C_l$ such that each C_{i+1} contains C_i in its interior, this observation does not improve the above bound; see Figure 8.

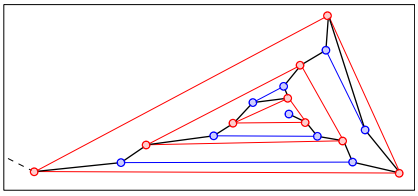


Figure 8: Multiple red 3-cycles for a path resulting from a matching of the according inflated polygon.

For spanning cycles, the proof of Theorem 4 can be slightly modified, yielding the following lower bound.

Corollary 6 $m_{cycle}(n) \geq \lceil \frac{n-1}{4} \rceil$.

Proof. For creating a simple polygon P for a given spanning cycle $C(S)$, we duplicate an extreme vertex v (w.l.o.g. a blue one), and cut the cycle between v and its duplicate v'. Then we extend one of the incident edge of v' until it hits the bounding box, and again inflate the resulting construction, by this hiding v' again. Applying the above construction, we obtain at least $\lfloor \frac{n}{2} \rfloor$ red matching edges. As the "end" vertex v of the inflated path is extreme, the red edges translated back to S cannot form any odd cycles. Thus we can use at least half of the obtained red edges for a $C(S)$-compatible monochromatic matching. □

For upper bounds, consider the examples shown in Figure 9. In Figure 9 (left), any monochromatic compatible edge is incident to one of the two high degree vertices, implying $m_{tree}(n) \leq 1$. In Figure 9 (right), the only vertex to which an "end vertex" of a "spike" can be matched is the next vertex on the big circle, implying that $m_{cycle}(n)$, $m_{path}(n) \leq \frac{5n+7}{12}$.

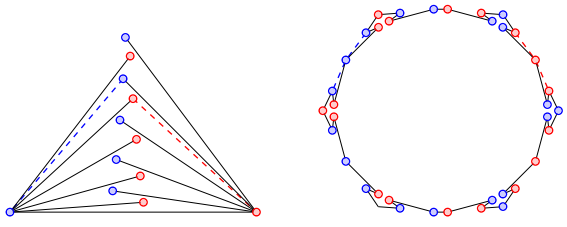


Figure 9: Upper bound examples for spanning trees (left) and spanning cycles (right).

4 Conclusion

We have shown the following bounds on the numbers of edges $m_{\mathcal{C}}(n)$ that can be reached by a monochromatic $G(S)$-compatible matching, with $G(S) \in \mathcal{C}$ and $\mathcal{C} \in \{tree, path, cycle, match\}$.

$$\begin{array}{ccccc} \frac{n}{4} & \leq & m_{match}(n) & \leq & \frac{5n}{12} \\ \lceil \frac{n-2}{6} \rceil & \leq & m_{path}(n) & \leq & \frac{5n+7}{12} \\ \lceil \frac{n-1}{4} \rceil & \leq & m_{cycle}(n) & \leq & \frac{5n+7}{12} \\ & & m_{tree}(n) & = & 1 \end{array}$$

Further, we showed that every bichromatic perfect matching admits a disjoint compatible bichromatic matching with at least $\lceil \frac{n-1}{2} \rceil$ edges, and that there exist point sets with non-unique bichromatic perfect matchings for which any compatible bichromatic (perfect) matching has at most $\frac{3n}{4}$ ($\frac{n}{2}$) disjoint edges.

We conclude with the following open problem. Given a bichromatic perfect matching $PM(S)$, can we always find a bichromatic compatible perfect matching $M(S)$ with $d(PM(S), M(S)) > 0$, supposed that $S = R \cup B$ is a point set admitting at least two different bichromatic perfect matchings?

Acknowledgments

Research partially supported by the ESF EUROCORES programme EuroGIGA - ComPoSe, Austrian Science Fund (FWF): I 648-N18 (O.A. and B.V.), and MICINN Project EUI-EURC-2011-4306 (F.H.). Research of F. H. also partially supported by projects MICINN MTM2009-07242, Gen. Cat. DGR 2009SGR1040.

References

[1] M. Abellanas, A. García, F. Hurtado, J. Tejel, and J. Urrutia. Augmenting the connectivity of geometric graphs. *Computational Geometry*, 40:220–230, 2008.

[2] O. Aichholzer, A. García, F. Hurtado, and J. Tejel. Compatible matchings in geometric graphs. In *Proc. XIV Encuentros de Geometría Computacional*, pages 145–148, Alcalá de Hernares, Spain, 2011.

[3] M. Hoffmann and C. Tóth. Pointed and colored binary encompassing trees. In *Proc. 21st ACM Symposion on Comp. Geom.*, pages 81–90, Pisa, Italy, 2005.

[4] F. Hurtado, M. Kano, D. Rappaport, and C. D. Tóth. Encompassing colored planar straight line graphs. *CGTA*, 39:14–23, 2008.

[5] M. Ishaque, D. L. Souvaine, and C. D. Toth. Disjoint compatible geometric matchings. In *Proc. 27th ACM symposium on Comp. Geom.*, pages 125–134, New York, USA, 2011.

[6] L. Larson. *Problem-Solving Through Problems*. Springer, New York, 1983.

[7] B. Vogtenhuber. *Combinatorial Aspects of [Colored] Point Sets in the Plane*. PhD thesis, Graz University of Technlolgy, Graz, Austria, 2011.

On Counting Empty Pseudo-Triangles in a Point Set

Sergey Kopeliovich* Kira Vyatkina†

Abstract

We address the problem of counting empty pseudo-triangles defined by a set of points in the plane. First, we assume that the three convex vertices are fixed, and provide algorithms, which solve the respective problem in $O(n^2)$ time using $O(n)$ space, for the cases when the pseudo-triangles can be arbitrary or must be star-shaped. The relaxation of our assumption increases the time complexity by factor n^3 in either case.

1 Introduction

The problem of counting empty polygons in a planar point set has been studied in the literature for a few decades. A polygon is *contained* in a set P of points if all its vertices reside at the points from P, and is *empty* if no point from P falls inside it. The goal is usually to determine or to bound the number of empty polygons contained in P, or to answer some related questions.

In 1978, Erdös [7] asked for the smallest number $n(k)$, such that any set of $n(k)$ points in the plane contains a convex empty k-gon. The results obtained in this respect are surveyed in [1]. Certain bounds on the number of convex empty polygons in a point set can be found in [5, 6, 9].

Analogous questions can be stated for non-convex polygons. They were first addressed by van Kreveld and Speckmann [8], and applied to empty pseudo-triangles in a planar point set.

A *pseudo-triangle* is a planar polygon with precisely three convex vertices, which are often referred to as *corners*.

In 2007, van Kreveld and Speckmann [8] provided tight asymptotic bounds on the minimum and maximum number of arbitrary empty pseudo-triangles in a point set, and of star-shaped ones. Two years later, Ahn et al. [3] examined the problem of generating such pseudo-triangles optimal in a certain sense—and namely, those with the minimum perimeter, the maximum area, or the minimum longest convex chain. (As a matter of fact, Ahn et al. [3] used a more general definition of a pseudo-triangle, allowing its corners to be connected by concave *curves*, and not necessarily polygonal chains. However, the respective optimization problems immediately reduce to those of finding an optimal *polygonal* pseudo-triangle, defined as above.) A year ago, Aichholzer et al. [2] investigated non-convex *4-holes* in a point set, which essentially represent non-convex quadrilaterals, and can be viewed as a special case of pseudo-triangles. Finally, very recently, Ahn et al. [4] have further elaborated the results from [8] and [3] and summarized them in a joint paper.

When looking for an empty pseudo-triangle with the minimum perimeter or the maximum area, the algorithms by Ahn et al. [3] actually generate all the empty pseudo-triangles defined by P. In addition, the asymptotic bounds by van Kreveld and Speckmann [8] imply that those algorithms fulfil the mentioned generation task in worst-case optimal time.

The aim of the present research is to develop simple and maximally efficient algorithms for counting empty pseudo-triangles in a point set. Of course, the method by Ahn et al. [3, 4] allows accomplishing this task; however, we demonstrate that the counting problem can be solved at a lower computational cost than the generating one.

We shall follow the scheme suggested by van Kreveld and Speckmann [8] and further adopted by Ahn et al. [3, 4], and first restrict our attention to the case when some three points from P are selected, and we are interested only in the empty pseudo-triangles with the corners at those points. Then the worst-case number of general and of star-shaped empty pseudo-triangles is known to be $\Theta(n^3)$ and $\Theta(n^2)$, respectively [8], and the method by Ahn et al [3, 4] adapted for solving the corresponding counting problems will require $O(n^3)$ time and $O(n)$ space in either case. In this work, we propose algorithms that run in $O(n^2)$ time and linear space in either case. Through the arguments provided in [8, 4], our results immediately extend to the general case, when the pseudo-triangle corners are not fixed; the computational time is thereby slowed down by a factor of n^3 in each case, while the working storage always remains linear. All our algorithms exploit only the most common data structures like arrays or linked lists.

In the next section, we introduce main observations and ideas needed to develop and justify our methods. In Section 3, the algorithms for counting empty pseudo-triangles with fixed corners are provided, for

*Department of Mathematics and Mechanics, Saint Petersburg State University, burunduk30@gmail.com

†Algorithmic Biology Laboratory, Saint Petersburg Academic University, Russian Academy of Sciences, kira@math.spbu.ru

the cases when the pseudo-triangles can be arbitrary or must be star-shaped, respectively. Subsequently, we eliminate the restrictions on the corner location. We conclude with a few brief remarks.

2 Preliminaries

Let P be a set of n points in the plane. To simplify the exposition, let us assume that no three points from P are collinear. In addition, we shall first assume that some three points $a, b, c \in P$ have been selected, and restrict our attention to empty pseudo-triangles, the corners of which reside at a, b, and c. Clearly, only the points from P lying inside $\triangle abc$ may become the vertices of such an empty pseudo-triangle. Since we are mainly interested in the worst-case behavior of our algorithms, we shall suppose that a, b, and c constitute the set of vertices of the convex hull of P, which is thus the triangle $\triangle abc$.

Without loss of generality, let us assume that the segment ab is horizontal, a is its leftmost endpoint, and c lies above ab (Fig. 1a).

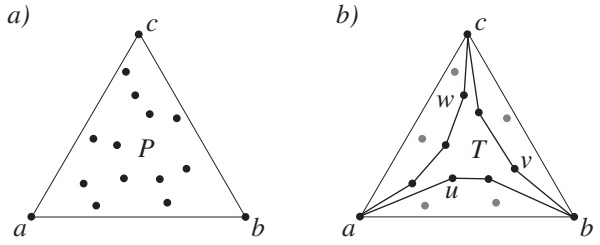


Figure 1: a) The convex hull of the set P of points is the triangle $\triangle abc$. b) In the boundary of the empty pseudotriangle T, the vertices u, v, and w follow the corners a, b, and c, respectively, in the counterclockwise order of vertices.

Let T be an empty pseudo-triangle, and let u, v, and w be its three vertices immediately following a, b, and c, respectively, in the counterclockwise order or vertices (Fig. 1b). Note that u, v, and w may coincide with b, c, and a, respectively; yet all the three coincidences occur simultaneously if and only if $P = \{a, b, c\}$.

We shall use the following fact proved it [8].

Lemma 1 *To each empty pseudo-triangle with the corners at a, b, and c, a distinct triple of vertices (u,v,w) corresponds.*

The converse is obviously not true: for example, if some two of the segments au, bv, and cw—which are supposed to be pseudo-triangle edges—intersect, then the triple (u, v, w) does not correspond to any pseudo-triangle.

For any two points p and q in the plane, we shall denote by l_{pq} the line through p and q.

We shall make use of the following observation [8, 3]. Suppose that a triple (u, v, w) does define a (unique) empty pseudo-triangle T. Then the lower concave boundary chain of T, which connects a and b, can be obtained as follows: compute the convex hull of the set of points consisting of a, b, u, and all the points from P lying below l_{au} and l_{bv}, take its boundary, and remove from it the edge ab (Fig. 2).

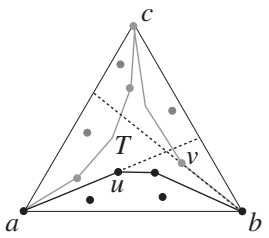


Figure 2: The concave chain connecting a and b in the boundary of T represents the boundary of the convex hull of the black points, from which the edge ab has been eliminated.

Since u and v fully determine the concave boundary chain between a and b, we shall further denote it by C_{uv}.

Our algorithms for counting empty pseudo-triangles will examine all the (valid) candidate pairs (u, v) of points from P, and for each of them determine the number of choices for w, which would result in a triple (u, v, w) defining either an arbitrary or a star-shaped empty pseudo-triangle. To obtain such a pair (u, v), we shall first choose u, and then appropriately select v.

3 Counting Problems

3.1 General Empty Pseudo-Triangles

Obviously, any point from $P \setminus \{a, c\}$ can be selected as u, while a and c cannot (Fig. 3a). Suppose that u and some two points v and w form a triple defining an empty pseudo-triangle T. Since C_{uv} must lie strictly below l_{bv}, we conclude that v must lie strictly above l_{bu}. On the other hand, any point from P residing above l_{bu} can be chosen as v (Fig. 3b).

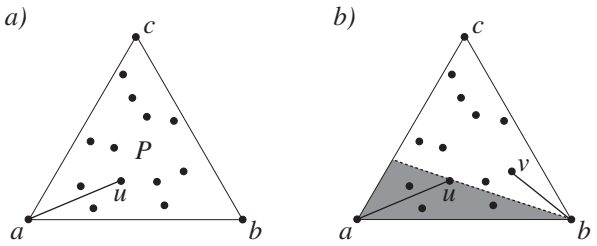


Figure 3: a) Any point from $P \setminus \{a, c\}$ can be chosen as u. b) Any point lying above l_{bu} can be chosen as v.

It remains to select w. Since C_{vw} must lie strictly to the right of l_{cw}, w must lie strictly to the left of cv. Moreover, C_{wu} must lie above l_{au}, and share with it a single point a; therefore, w must either be a or lie strictly above l_{au} (Fig. 4a). Finally, our choice of w must ensure that the triangle bounded by the lines l_{au}, l_{bv}, and l_{cw} is empty (see Fig. 4b), since this triangle would be contained in the pseudo-triangle having the edges au, bv and cw, and thus, any point from P lying inside the former would also fall inside the latter.

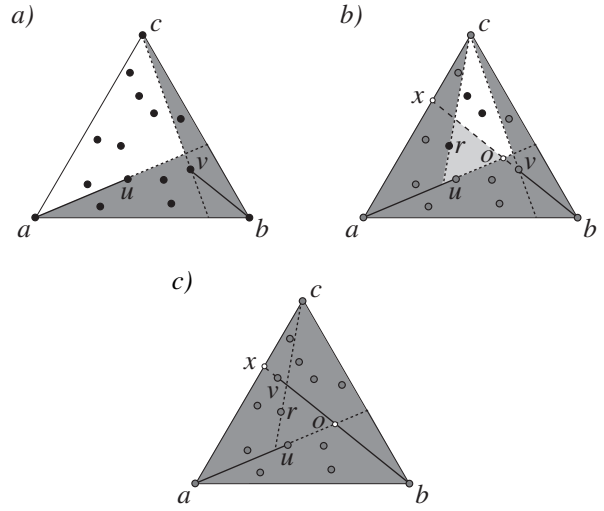


Figure 4: a) A candidate point w cannot lie in the gray area. b) Since the interior of the light-gray triangle bounded by the lines l_{au}, l_{bv}, and l_{cw} must be empty, w must be one of the three black points (here, $w = r$). c) If v lies to the left of l_{cr}, no point can be selected as w.

Let $o = l_{au} \cap l_{bv}$, and let $x = l_{bv} \cap ac$ (Fig. 4b). Moreover, let r be the point from P lying inside $\triangle aox$, which maximizes the angle $\angle acr$. First, suppose that v lies to the right of l_{cr}. Then the valid candidate points to be selected as w are represented by r and the subset P' of points from P lying between l_{cr} and l_{cv} and above l_{bv} (in the white triangle, according to Fig. 4b). Note that P' can be alternatively viewed as the subset of points from P lying strictly between l_{cr} and l_{cv} and above l_{au} (due to the fact that the interior of the light-gray triangle must be empty, according to Fig. 4b).

Observe that v may happen to lie to the left of l_{cr} (Fig. 4c). If so, no point from P is a valid candidate for w.

Now let us outline an algorithm, which allows to efficiently compute the number of empty general pseudo-triangles in P.

Algorithm EmptyGeneral(P, a, b, c)
Input: P – set of points with a convex hull $\triangle abc$
Output: Num – the number of empty general pseudo-triangles in P with the corners at a, b, c

1. Let P_b be a set of points composing $P \setminus \{b\}$ sorted in a cw order w.r.t. b.
   Let P_c be a set of points composing $P \setminus \{c\}$ sorted in a ccw order w.r.t. c.
2. $Num \leftarrow 0$
3. **for each** point $u \in P \setminus \{a, c\}$
4. Let $P_u \subset P_c$ consist of the point a and all points from P_c lying strictly above l_{au}.
5. **for each** point $q \in P_u$
6. Let $left[q]$ be the number of points from P_u lying strictly to the left from l_{cq}.
7. $r \leftarrow a$
8. **for each** point $v \in P_b$
9. **if** v lies above l_{bu} and to the right of l_{cr}
10. $Num \leftarrow Num + left[v] - left[r]$
11. **if** v lies above l_{au} **and** $\angle acv > \angle acr$
12. $r \leftarrow v$
13. **return** Num

Correctness of the proposed algorithm is guaranteed by the discussion carried out above. The time complexity of the preprocessing stage (Steps 1 and 2) is $O(n \log n)$. It is easy to see that for each point u, the respective iteration of the cycle "for" (Steps 4–12) is accomplished in linear time. Obviously, linear storage is sufficient for any auxiliary data structures. We conclude that the algorithm **EmptyGeneral** runs in $O(n^2)$ and $O(n)$ space.

Theorem 2 *Let P be a set of n points in the plane, such that its convex hull represents a triangle $\triangle abc$, and no three its points are collinear. The number of empty general pseudo-triangles in P with the corners at a, b, c can be computed in $O(n^2)$ time using $O(n)$ space.*

3.2 Empty Star-Shaped Pseudo-Triangles

To count the empty star-shaped pseudo-triangles in P, we follow essentially the same scheme. The only distinction is that w now needs to be chosen more carefully.

Suppose we have selected the points u and v as described in Section 3.1 (see Fig. 3). Consider the triangle $\triangle aox$ and the point r defined as above (see Fig. 4b). As in the case of general empty pseudo-triangles, if v lies to the left of l_{cr}, no point can be selected as w (see Fig. 4c). Otherwise, w again must either be r or lie strictly between l_{cr} and l_{cv}, and above l_{bv}. However, if $o \notin bv$, this does not yet guarantee that the pseudo-triangle defined by the triple (u, v, w) would be star-shaped: to this end, we must require w to lie strictly to the left of the line l_{co}, which is a stronger condition (Fig. 5a). Observe that the set of points lying strictly inside the triangle bounded by l_{cr}, l_{co} and l_{bv} is identical to that lying strictly inside the triangle bounded by l_{cr}, l_{co} and l_{au}. Consequently, in this case, w can be either r or any point

lying strictly inside the triangle bounded by the lines l_{cr}, l_{co} and l_{au}.

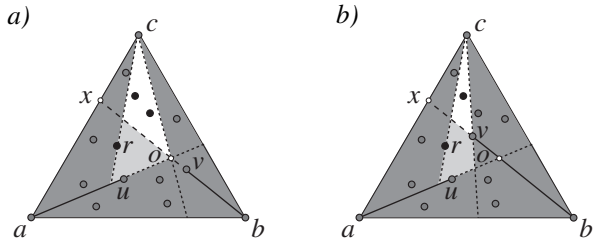


Figure 5: The set of valid candidates for w (black points) consists of r and the points lying strictly inside the triangle bounded by a) l_{cr}, l_{co} and l_{au}, if $o \notin bv$; b) l_{cr}, l_{cv} and l_{au}, if $o \in bv$ but $cr \cap bv = \emptyset$.

If $o \in bv$ but v lies to the right of l_{cr} (that is, $cr \cap bv = \emptyset$), then the condition that w lies strictly to the left of the line l_{cv} is sufficient, and w can be either r or any point lying strictly inside the triangle bounded by the lines l_{cr}, l_{cv} and l_{au} (Fig. 5b).

Now it should become clear that the method for counting empty star-shaped pseudo-triangles in P is fully similar to the one for counting general ones. The most important distinction is that throughout the algorithm, we should maintain the number of points from P lying strictly to the left of l_{co}. However, since the point o moves along the l_{au} constantly to the right, this is easy to incorporate without affecting the complexity.

Theorem 3 *Let P be a set of n points in the plane, such that its convex hull represents a triangle $\triangle abc$, and no three its points are collinear. The number of empty star-shaped pseudo-triangles in P with the corners at a, b, c can be computed in $O(n^2)$ time using $O(n)$ space.*

4 Empty Pseudo-Triangles with Arbitrary Corners

To relax the assumptions that the convex hull of P is a triangle, and the corners of the pseudo-triangles being counted reside at its three vertices, we simply note that in a general set of points, there are $O(n^3)$ distinct triples of those. To count all the empty general or star-shaped triangles for a given set of points, it is sufficient to handle each triple separately using the algorithms described in the previous section. Consequently, the running time in both cases will increase up to $O(n^5)$, while the amount of storage will remain linear.

5 Conclusion

An open question that arises immediately is whether it is possible to develop faster algorithms for solving the counting problems addressed in this paper. Another potential direction for future research is generalizing the results obtained for empty pseudo-triangles in a point set to the case of empty pseudo-quadrilaterals and other polygonal figures.

Acknowledgments

The second author highly appreciates discussions with Hee-Kap Ahn and Iris Reinbacher carried out when she was visiting Geometric Computing Lab at Pohang University of Science and Technology (POSTECH) in July 2009. The authors thank two anonymous referees for helpful comments.

References

[1] O. Aichholzer, [Empty] [colored] k-gons - Recent results on some Erdos-Szekeres type problems. *In Proc. XIII Encuentros de Geometria Computacional, Zaragoza, Spain*, 43–52, 2009.

[2] O. Aichholzer, R. Fabila-Monroy, H. González-Aguilar, T. Hackl, M. A. Heredia, C. Huemer, J. Urrutia, and B. Vortenhuber, 4-Holes in Point Sets. *In Proc. the 27th European Workshop Comp. Geom. (EuroCG 2011)*, 115–118, 2011.

[3] H.-K. Ahn, S. W. Bae, and I. Reinbacher, Optimal empty pseudo-triangles in a point set. *In Proc. the 21st Canadian Conf. Comp. Geom. (CCCG 2009)*, 5–8, 2009.

[4] H.-K. Ahn, S. W. Bae, M. van Kreveld, I. Reinbacher, and B. Speckmann, Empty pseudo-triangles in point sets. *Discr. Appl. Math.*, 159(18):2205–2213, 2011.

[5] I. Bárány and P. Valtr, Planar point sets with a small number of empty convex polygons. *Studia. Sci. Math. Hungar.*, 41:243-266, 2004.

[6] A. Dumitrescu, Planar sets with few empty convex polygons. *Studia. Sci. Math. Hungar.*, 36:93-109, 2000.

[7] P. Erdös, Some more problems on elementary geometry. *Austral. Math. Soc. Gaz.*, 5:52–54, 1978.

[8] M. J. van Kreveld and B. Speckmann, On the number of empty pseudo-triangles in point sets. *In Proc. the 19st Canadian Conf. Comp. Geom. (CCCG 2007)*, 37–40, 2007.

[9] R. Pinchasi, R. Radoicic, and M. Sharir, On empty convex polygons in a planar point set. *J. Comb. Theory, Ser. A*, 113:385-419, 2006.

Computer Generation of Triply-Crossing Tile Patterns

Kokichi Sugihara*

Abstract

This paper presents an algorithm for generating new tiling patterns where three streams of tiles cross without overlap or gaps. Copies of three tiles are placed in three different directions, gradually changing their shapes until they cross at the center. The algorithm is based on the periodic tiling of hexagons, morphing between two tiles, and mixing two or more curves. The proposed method might provide a new direction for the extension of the tiling art started by M. C. Escher.

1 Introduction

For many years, tiling patterns have been used to decorate the walls and floors of buildings. Tiling patterns were based on simple geometric figures such as triangles and rectangles for most of that time.

However, the situation changed in the 20th century. The Dutch artist M. C. Escher developed a new form of tiling art in his series of artworks titled "Regular Division of the Plane" [2, 10], where periodic patterns were generated using a few complicated tiles, such as animal shapes, instead of simple polygons. A typical group contains tiling patterns composed of copies of one tile, such as "Regular Division of the Plane No. 56 (Lizard, 1946)," while another group consists of copies of two tiles, such as "Regular Division of the Plane No. 111 (Flying Fish and Bird, 1962)."

Escher developed another new direction for tiling art by combining tiling with a gradual deformation of the tiles and figure–ground reversal. A typical example is "Sky and Water I" (1938), where a bird at the top is gradually deformed downward to melt into the background, while fish tiles emerge from the background and become clearest at the bottom.

With these artworks, Escher showed that tiling can be used to create a variety of new artistic patterns [1, 8]. However, the creation of such patterns requires hard work and trial and error, because the possible tile shapes that can be used in this art form are very limited and it is not easy to find a good combination of tiles.

In contrast, the mathematical structures of periodic tiling have been well studied, for example, see Grünbaum and Shepard [3], and Martin [7]. The automatic generation of tiling patterns has also been studied. Kaplan and Salesin [4, 5] proposed methods for finding tilable shapes close to a given shape, to automatically generate periodic tiling. Koizumi and Sugihara [6] proposed a different approach to the same problem, and they provided an efficient method. Sugihara [9] also proposed a method for generating Sky-and-Water-type tiling patterns using a given pair of tiles.

In this paper, we propose a further extension for the automatic generation of tiling patterns. For any given set of three different tiles, our method generates a tiling pattern where the three groups of tiles are aligned in different directions until they cross at the center.

We present our basic idea in Section 2, details of the method with an example in Section 3, and concluding remarks in Section 4.

2 Basic Idea

We aim to generate tiling patterns where three groups of tiles cross at the center. For this purpose, we use the hexagonal grid shown in Fig. 1 as the underlying structure. As shown in the figure, we assigned label 1 to hexagons from the upper left point to the lower right point in three diagonal lines. Similarly, label 2 was assigned to hexagons from the upper right point to the lower left point in three lines, with label 3 forming three horizontal lines.

This assignment of the three labels is homogeneous in the following sense. Any hexagon labeled i ($i = 1, 2, 3$) in Fig. 1 is surrounded by six hexagons in the same manner as shown in Fig. 2, where if $i = 3$, $i + 1$ is read as 1, and if $i = 1$, $i - 1$ is read as 3. As shown in this figure, we number the six edges of a hexagon as $1, 2, \ldots, 6$ starting at the upper right edge. We denote the jth edge of a hexagon labeled i as edge (i, j) for $i = 1, 2, 3$ and $j = 1, 2, \ldots, 6$. Then, each edge has the same counterpart whenever it contacts another edge. For example, edge $(1, 1)$ contacts edge $(3, 4)$, and edge $(1, 2)$ contacts edge $(2, 5)$. In this sense, the assignment of labels 1, 2, and 3 in Fig. 1 is homogeneous.

This property of homogeneity is suitable for the generation of our tiling pattern, because if each contacting pair of edges share the same deformation, the resulting pattern would be tiling, i.e., we can place

*Graduate School of Advanced Mathematical Sciences, Meiji University/ JST, CREST, kokichis@isc.meiji.ac.jp

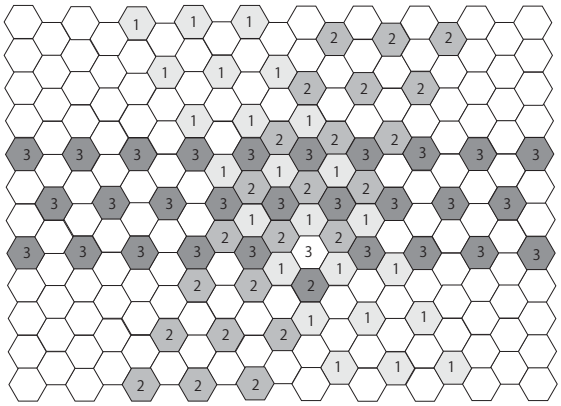


Figure 1: Hexagonal grid used for generating triply-crossing tile patterns.

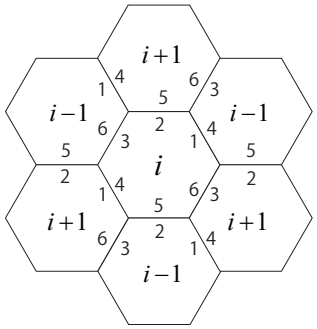


Figure 2: A tile and six surrounding tiles.

the deformed hexagons without overlap or gaps.

Suppose that we are given three different tiles. Our goal is to place copies of these tiles on the hexagonal grid with the minimum deformation so that every pair of mutually neighboring tiles makes contact without overlap or gaps. We use the following steps to achieve this goal.

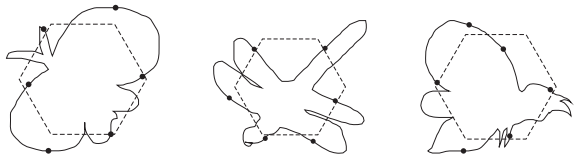
Outline of the procedure

1. Adjust the sizes of the three tiles by fixing a common regular hexagon, which is referred to as the reference hexagon, to the three tiles.
2. Divide the boundary of each tile into six boundary pieces that correspond to the six sides of the reference hexagon.
3. Deform the three tiles by taking averages along the common edges of neighboring reference hexagons.
4. Place the deformed tiles at the center of the hexagonal grid, then put the original tiles on hexagons that are sufficiently distant from the center, before placing morphed tiles in between.

A detailed description of these steps will be provided in the following section.

3 Generation of Tile Patterns

Suppose that we are given three different tile shapes. Let us call them as tile 1, tile 2, and tile 3. Fig. 3 shows an example of a set of three tiles, represented by the solid lines. Tile 1 is a "butterfly," tile 2 is a "dragonfly," and tile 3 is a "bee." We explain our procedure using this set of tiles as an example.



(a) tile 1 (b) tile 2 (c) tile 3

Figure 3: Three tiles.

3.1 Reference Hexagons

Suppose that the boundary of a tile is represented as a closed curve $c(s)$ with arc-length parameter s, $0 \le s \le S$. First, we find the center of gravity, say g, and we then compute the average of the distances from g to the boundary points:

$$r = \frac{1}{S}\int_0^S d(g, c(s))\mathrm{d}s, \qquad (1)$$

where $d(p, q)$ denotes the Euclidean distance between two points p and q. The value r can be considered as the average radius of the tile.

We generate a regular hexagon with center g and radius r, where the radius of a regular hexagon is the distance from the center to a vertex of the hexagon. We refer to this hexagon as the *reference hexagon*.

We generate reference hexagons for all three tiles and adjust the tile size until the reference hexagons are congruent.

In Fig. 3, the broken lines indicate the reference hexagons of the three tiles.

3.2 Decomposition of Tile Boundaries

Let $c_i(s)$, $0 \le s \le S_i$, be the counterclockwise boundary curve of tile i. For simplicity, we assume that $c_i(0) = c_i(S_i)$ is the rightmost point of the tile. Let h_j, $j = 1, 2, \ldots, 6$, be the vertices of the reference hexagon numbered from the rightmost vertex counterclockwise. Let $c_i(s_j)$, $j = 1, 2, \ldots, 6$, be the points on the curve that are closest to the vertices h_j. We refer to these points as *break points*. In Fig. 3, the break points are represented as dots.

We assume that:

$$s_1 = 0 < s_2 < s_3 < \cdots < s_6 < S_i. \qquad (2)$$

In other words, we assume that the six points $c_i(s_j)$ are distinct and they are placed counterclockwise in

this order. Next, we divide the boundary curve of the tile into six pieces at these points and change the parameter from s to u so $0 \leq u \leq 1$ for each piece. The six pieces of the boundary curve are represented as:

$$c_{ij}(u), \quad 0 \leq u \leq 1,\ j = 1, 2, \ldots, 6, \tag{3}$$

where

$$c_{ij}(0) = c_i(s_j),\ c_{ij}(1) = c_i(s_{j+1}),\ j = 1, 2, \ldots, 6 \tag{4}$$

(s_7 should be read as s_1). We also define the reverse curve $\overline{c}_{ij}(u)$ as

$$\overline{c}_{ij}(u) = c_{ij}(1-u),\ 0 \leq u \leq 1, j = 1, 2, \ldots, 6. \tag{5}$$

Note that the argument above assumes that the six break points $c_i(s_j),\ j = 1, 2, \ldots, 6$ lie counterclockwise on the curve. However, this is not always true because when the shape of a tile is complicated, such as a waving snake, the six points nearest to the vertices of the hexagon might be located in a random order along the boundary. In that case, we have to locate the six boundary points using another method, say manually or so on.

3.3 Mixing Boundary Pieces

The next task is to deform the boundary pieces of the tiles until the tiles can be placed without overlap or gaps. As we observed, there is a homogeneous neighbor relationship between the sides of the three labeled hexagons because there is a unique counterpart for each edge.

For example, the edge $(1, 1)$, i.e., the first edge of the hexagon labeled 1 is adjacent to the edge (3,4), i.e., the fourth edge of the hexagon labeled 3. This generates a spatial relationship between the two tiles if we draw the tiles and the reference hexagons as shown in Fig. 4.

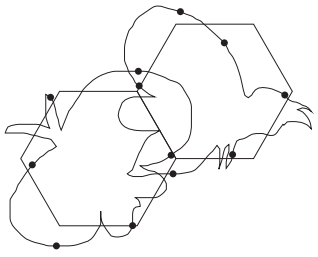


Figure 4: Tile 1 and tile 3 are adjacent along the first edge of the reference hexagon fixed to tile 1.

Suppose that edge (i, j) is adjacent to edge (i', j'). This means that the jth edge of tile i is adjacent to the j'th edge of tile i'. We need to deform the tiles so that they make contact without overlap or gaps. A natural solution might be to take the average of the two curves at each corresponding value of the parameter. We can generate the average curve $a_{ij}(u)$ by:

$$a_{ij}(u) = \frac{1}{2}(c_{ij}(u) + \overline{c}_{i'j'}(u)). \tag{6}$$

However, this does not work well, because the deformed curves become inconsistent at the point where the three reference hexagons meet. To avoid this difficulty, we locate the deformed points that correspond to the vertices of the hexagons, before taking the average of the boundary pieces.

For example, consider the start point of the first boundary piece of tile 1. This point corresponds to the start point of edge (1,1). The neighbor relationships in Fig. 2 show that this point coincides with the start point of edge (2,3) and the start point of edge (3,5). Thus, we take the average of these three start points and consider it as the location of the start points of the deformed boundary pieces. Formally, we perform the following process.

Let v_{ij} be the start point of the jth boundary piece of tile i, and v'_{ij} and v''_{ij} be the start points of the boundary pieces of two other tiles where the corresponding vertices of their reference hexagons occupy the same location. We take the average:

$$w_{ij} = \frac{1}{3}(v_{ij} + v'_{ij} + v''_{ij}) \tag{7}$$

and consider it the common start point of the associated deformed boundary pieces of the three tiles.

We adjust the size and orientation of each boundary piece until the start and end points coincide with the average points defined by eq. (7). More formally, we apply an orthogonal affine transformation to the boundary piece $c_{ij}(u)$ so $c_{ij}(0)$ and $c_{ij}(1)$ are transformed to w_{ij} and w_{ij+1}, respectively. This can be done by first scaling $c_{ij}(u)$ until the length between $c_{ij}(0)$ and $c_{ij}(1)$ coincides with the length between w_{ij} and $w_{i,j+1}$. Then, we translate $c_{ij}(u)$ so $c_{ij}(0)$ moves to w_{ij}, before finally rotating it around w_{ij} until the other terminal point $c_{ij}(1)$ coincides with $w_{i,j+1}$. Let us denote the resulting boundary piece as $c_{ij}(u)$.

After this adjustment, we take the average of the two corresponding curves using eq. (6). This procedure allows us to obtain three deformed tiles that can be placed without overlap or gaps.

3.4 Morphing between the Original Tiles and Deformed Tiles

The final task is to place the original, deformed, and intermediate tiles on the hexagonal grid. This is achieved using the following method.

Consider a hexagon with label 1, 2, or 3 in the hexagonal grid in Fig. 1. We refer to the hexagon as *connected* if it is adjacent to at least one labeled hexagon, and *isolated* if it is not.

Let d_1 be the maximum distance from the center of the grid to the center of the connected hexagons, while d_2 is a real value greater than d_1.

Let H be a hexagon labeled i in the hexagonal grid shown in Fig. 1, and d be the distance from the center of the grid to the center of H. We consider the following three cases.

First, suppose that H is connected. We place the deformed version of tile i so the reference hexagon coincides with H.

Second, suppose that H is isolated and $d_2 \leq d$. We then place the original tile i on H.

Third, suppose that H is isolated and $d < d_2$. We then make an intermediate shape by mixing pieces of the original tile $c_{ij}(u)$ and the deformed tile $a_{ij}(u)$ by:

$$b_{ij}(u) = \frac{(d - d_1)c_{ij}(u) + (d_2 - d)a_{ij}(u)}{d_2 - d_1}. \quad (8)$$

Using the three tiles shown in Fig. 3, we can generate the tile pattern shown in Fig. 5.

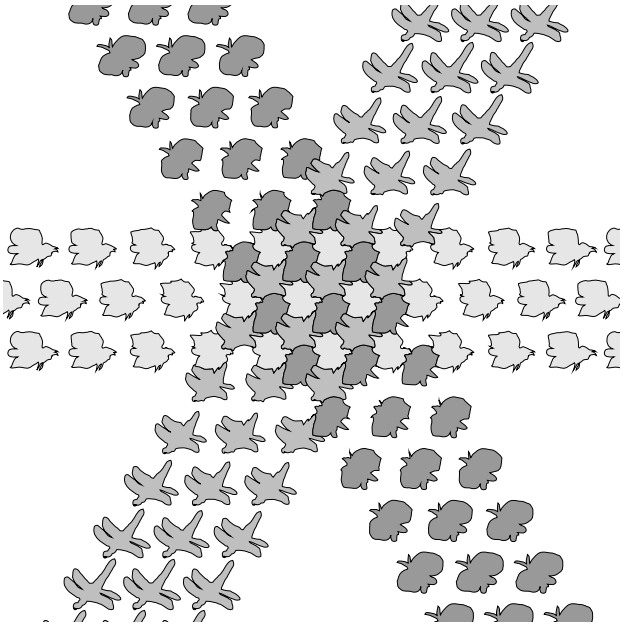


Figure 5: Triply-crossing tile pattern.

4 Concluding Remarks

We proposed a method for generating a triply-crossing tile pattern using three given tiles. This may be a new direction for extending Escher's tiling art.

There is much scope for future work.

First, we used many heuristics to identify parameters such as reference hexagons and break points. However, we cannot be sure that these heuristics are the best. Thus, we need to investigate these heuristics and search for better methods.

Second, our method can generate tiling patterns for any set of three tiles, but the deformation sometimes appears unnatural. To guarantee a high-quality output, we need to characterize suitable sets of three tiles and construct a method of finding them.

Other future work includes: (i) extension of triply-crossing to n-tuple-crossing for $n = 4, 5, 6$, etc.; (ii) extension to other crossing types such as a combination of three duple-crossing areas; and (iii) extension of triply-crossing to a combination of crossing and figure–background reversal similar to Escher's "Sky and Water I."

Acknowledgments

This work was partly supported by a Grant-in-Aid for Scientific Research (B) (No. 20360044) from the Japanese Society for the Promotion of Science, and Meiji University Global COE Program on Formation and Development of Mathematical Sciences Based on Modeling and Analysis.

References

[1] B. Ernst (J. E. Brigham, trans.). *The Magic Mirror of M. C. Escher.* Taschen America, New York, 1976.

[2] M. C. Escher. *The Graphic Work of M. C. Escher.* London, 1961.

[3] B. Grünbaum and G. C. Shepard. *Tilings and Patterns — An Introduction.* W. H. Freeman and Company, New York, 1989.

[4] C. S. Kaplan and D. H. Salesin. *Escherization.* Proceedings of SIGGRAPH 2000, New Orleans, LA, pp. 499–510.

[5] C. S. Kaplan and D. H. Salesin. *Dihedral Escherization.* Proceedings of Graphics Interface 2004, W. Heidrich, R. Balakrishnan and A. K. Peters, eds., 17–19 May, 2004, London, pp.255–262.

[6] H. Koizumi and K. Sugihara. *Maximum Eigenvalue Problem for Escherization.* Graphs and Combinatorics, vol. 27, pp. 431–439, 2011.

[7] G. E. Martin. *Transformation Geometry — An Introduction to Symmetry.* Springer-Verlag, New York, 1982.

[8] D. Schattschneider. *M. C. Escher — Vision of Symmetry, New Edition.* Harry N. Abrams Inc., New York, 2004.

[9] K. Sugihara. Computer-aided Generation of Escher-like Sky and Water Tiling Patterns. Journal of Mathematics and the Arts, vol. 3, no. 4, pp. 195–207, 2009.

[10] K. Yasuda (ed.). *The Collection of Huis Ten Bosch, M. C. Escher.* Nagasaki, Huis Ten Bosch, 1994.

New separation theorems and sub-exponential time algorithms for packing and piercing of fat objects

Farhad Shahrokhi
Department of Computer Science and Engineering, UNT
P.O.Box 13886, Denton, TX 76203-3886, USA farhad@cs.unt.edu

Abstract

For $\mathcal{C}$ a collection of n objects in R^d, let the packing and piercing numbers of $\mathcal{C}$, denoted by $Pack(\mathcal{C})$, and $Pierce(\mathcal{C})$, respectively, be the largest number of pairwise disjoint objects in $\mathcal{C}$, and the smallest number of points in R^d that are common to all elements of $\mathcal{C}$, respectively. When elements of $\mathcal{C}$ are fat objects of arbitrary sizes, we derive sub-exponential time algorithms for the NP-hard problems of computing $Pack(\mathcal{C})$ and $Pierce(\mathcal{C})$, respectively, that run in $n^{O_d(Pack(\mathcal{C})^{\frac{d-1}{d}})}$ and $n^{O_d(Pierce(\mathcal{C})^{\frac{d-1}{d}})}$ time, respectively, and $O(n \log n)$ storage. Our main tool which is interesting in its own way, is a new separation theorem. The algorithms readily give rise to polynomial time approximation schemes (PTAS) that run in $n^{O((\frac{1}{\epsilon})^{d-1})}$ time and $O(n \log n)$ storage. The results favorably compare with many related best known results. Specifically, our separation theorem significantly improves the splitting ratio of the previous result of Chan, whereas, the sub-exponential time algorithms significantly improve upon the running times of very recent algorithms of Fox and Pach for packing of spheres.

1 Introduction and Summary

An effective tool in the design of divide and conquer graph algorithms is separation. The separator theorem of Lipton and Tarjan [13],[14] asserts that any n vertex planar graph can be separated into two subgraphs with a splitting ratio of $1/3 - 2/3$, that is, each subgraph having at most $2n/3$ vertices, by removing $O(\sqrt{n})$ vertices. Additional results for graphs and geometric objects have been obtained by Alon et al [3], Alber and Fiala [2], Miller et al [15], Smith and Wormald [16], Chan [4], Fox and Pach [6] [7],[8],[10], Fox, Pach and Toth [9], and Shahrokhi [17].

1.1 Past Related Results

Throughout this paper $\mathcal{C}$ denotes a finite collection of subsets of R^d of cardinality n. Miller et al [15] proved that given $\mathcal{C}$ a set of spheres in R^d, where d is fixed and each point is common to only a constant number of spheres, there is a sphere S in R^d, so that at most $\frac{d+1}{d+2}$ of the spheres in $\mathcal{C}$ are entirely inside of S, at most $\frac{d+1}{d+2}$ are entirely outside, and at most $O_d(n^{\frac{d-1}{d}})$ intersect the boundary of S. Smith and Wormald [16], among other results, derived variations of this fundamental result, where the separator is a box, and hence, reduced the splitting ratio to $1/3-2/3$ in any dimension d. Nonetheless, their result required "disjointness" assumption of the spheres, which could be weakened to the assumption that there is a very small overlapping among the objects. Since the intersection graphs of many geometric objects exhibit a large vertex connectivity, it is impossible to separate them with the removal of a small number of vertices. Consequently, researchers, have focused on separating intersection graphs of geometric objects, including spheres, with respect to other measures.

Alber and Fiala [2] studied the separation properties of unit discs in R^2 with respect to the area which coincides with the packing number, in case of unit discs. For a set $\mathcal{C}$ of unit discs in R^2, they derived a separator theorem with a constant splitting ratio, used it to compute the $Pack(\mathcal{C})$ in $n^{O(\sqrt{Pack(\mathcal{C})})}$ time, and also derived a PTAS for computing the packing number.

Chan [4] studied the packing and piercing numbers of fat objects of arbitrary sizes in R^d. He introduced the concept of a measure on fat objects that coincides with the packing and piercing numbers, and generalized the separation result of Smith and Wormald for this measure. In simple words, Chan's beautiful separation theorem asserts that given a set $\mathcal{C}$ of n fat objects in R^d, with a sufficiently large measure $\mu(\mathcal{C})$, there is a cube R so that the measure of objects inside of R is at most $(1-\alpha)\mu(\mathcal{C})$, the measure of objects outside of R is at most $(1-\alpha)\mu(\mathcal{C})$, and the measure of objects intersecting R is $n^{O_d(\mu(\mathcal{C})^{\frac{d-1}{d}})}$, where α is a constant whose value is about $\frac{1}{2^d+1}$. Chan then used his separation theorem to design Polynomial time Approximation Schemes (PTAS) for packing and piercing problems that run in $n^{O((\frac{1}{\epsilon})^d)}$ time and $O(n)$ space. Specifically, he improved the running time of the best previous PTAS for packing problem that was due to Erlebach et al [5].

This is an extended abstract of a presentation given at EuroCG 2012. 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 in a conference with formal proceedings and/or in a journal.

In [17], we obtained a combinatorial measure separation theorem for a class of graphs containing the intersection graphs of nearly fat objects in R^d, and obtained sub-exponential algorithms and PTAS for packing and piercing number of unit hight rectangles, unit discs, and other related problems.

Fox and Pach [6], [7], [8] have developed a series of extremely powerful separator theorems for planar curves and string graphs. These results highly generalize the planar separator theorem and have striking applications in combinatorial geometry. Very recently, Fox and Pach discovered the potential of their methods for deriving the sub-exponential algorithms [10]. Among other results, they announced that their methods combined with the separation theorem of Miller et al can be compute the packing number of spheres in $2^{O_d\left(n^{\frac{d}{d+1}}polylogn\right)}$ time, or, in $n^{O_d\left(n^{\frac{d}{d+1}}\right)}$ time.

1.2 Our Results

We present several main results. First, we present an improvement to Chan's separation theorem which is obtained by combining some of his ideas with the original work of Smith and Wormald. Specifically, using a box of aspect ratio ratio 2 in R^d, as the separator, we obtain a parametric splitting ratio of $1/3 - 2/3(1+\epsilon)$, for any $0 < \epsilon \leq \frac{1}{2}$, for separating the measure, independent of the dimension. Moreover, we provide an explicit (constant) lower bound for $\mu(\mathcal{C})$, in terms of d and ϵ, for obtaining such a suitable separation, contrasting the result in [4] that was obtained for *sufficiently large values*. We use this separation theorem to derive sub-exponential time algorithms for packing and piercing problems, running in $n^{O_d(Pack(\mathcal{C})^{\frac{d-1}{d}})}$ and $n^{O_d(Pierce(\mathcal{C})^{\frac{d-1}{d}})}$ time, respectively, and $O(n \log n)$ storage. Finally, we convert these algorithms to PTAS that run in $n^{O((\frac{1}{\epsilon})^{d-1})}$ time and $O(n \log n)$ storage.

2 Preliminaries

For a closed set B in R^d, let δ_B denote the boundary of B, and note that $\delta_B \subseteq B$. Let $\bar{B}$ denote $R^d - B$, that is, $\bar{B}$ is the set of points "outside" of B. A collection $\mathcal{C}$ of objects in R^d is fat, if for every r and size r box R, we can choose a constant number c of points such that every object in $\mathcal{C}$ that intersects R and has size at least r contains one of these points [4]. It should be noted that the class of fat objects contains spheres, cubes, and boxes with bounded aspect ratios.

Let $\mathcal{C}$ be a collection of subsets of R^d, and let μ be a mapping that assigns non-negative values to subsets of $\mathcal{C}$. Chan [4] calls μ a measure, if for any $\mathcal{A}, \mathcal{B} \subseteq \mathcal{C}$ the following hold.

(i) $\mu(\mathcal{A}) \leq \mu(\mathcal{B})$, if $\mathcal{A} \subseteq \mathcal{B}$.

(ii) $\mu(\mathcal{A} \cup \mathcal{B}) \leq \mu(\mathcal{A}) + \mu(\mathcal{B})$.

(iii) $\mu(\mathcal{A} \cup \mathcal{B}) = \mu(\mathcal{A}) + \mu(\mathcal{B})$, if no object in $\mathcal{A}$ intersects an object in $\mathcal{B}$.

(iv) Given any $r > 0$ and any size$-r$ box R in R^d, if every object in $\mathcal{A}$ has size at least R and intersects R, then $\mu(\mathcal{A}) \leq c$, for a constant c.

(v) A constant-factor approximation to $\mu(\mathcal{A})$ can be computed in $|\mathcal{A}|^{O(1)}$ time. Moreover, if $\mu(\mathcal{A}) \leq b$, then, $\mu(\mathcal{A})$ can be computed exactly in $|\mathcal{A}|^{O(b)}$ time.

Note that feasible solutions to the packing and piercing problems gives rise to measures.

Let $\mathcal{A} \subseteq \mathcal{C}$, and let B be a closed subset of R^d. Let $\mathcal{A}_{B-\delta_B}$ and $\mathcal{A}_{\bar{B}}$ denote, respectively, the set of all objects in $\mathcal{A}$ that are contained in $B - \delta_B$, or are completely inside of B, and the set of all objects in $\mathcal{A}$ that are contained in $\bar{B}$, or are completely outside of B, respectively. Let $\mathcal{A}_B$, and $\mathcal{A}_{\delta_B}$, denote the set of all objects in $\mathcal{A}$ that have their centers in B, and the set of all objects in $\mathcal{A}$ that have a point in common with δ_B.

Aspect ratio of a box in R^d is the ratio of its longest side to its shortest side. Chan [4] proved the following separation theorem.

Theorem 1 *Given a measure μ satisfying $(i) - (iv)$ and a collection $\mathcal{C}$ of n objects in R^d with sufficiently large $\mu(\mathcal{C})$ there is a box R with $\mu(\mathcal{C}_{R-\delta_R}), \mu(\mathcal{C}_{\bar{R}}) \geq \alpha\mu(\mathcal{C})$, and $\mu(\mathcal{C}_{\delta R}) = O_d(n^{\mu(\mathcal{C})^{\frac{d-1}{d}}})$, where α is some fixed constant. Moreover, if (v) is satisfied then, R can be computed in polynomial time and linear space.*

3 The Separation Theorem

Theorem 2 *Let $\mathcal{C}$ be a set of n objects in R^d, $d \geq 2$ and let μ be a measure on $\mathcal{C}$ satisfying $(i) - (iv)$, and let $0 < \epsilon \leq \frac{1}{2}$. If $\mu(\mathcal{C}) \geq \left(\frac{3.c.d^2.8^d}{\epsilon}\right)^d$, then, there is a box R so that*

$$\mu(\mathcal{C}_{R-\delta R}) \leq \frac{2}{3}(1+\epsilon)\mu(\mathcal{C}), \mu(\mathcal{C}_{\bar{R}}) \leq \frac{2}{3}(1+\epsilon)\mu(\mathcal{C}),$$

and

$$\mu(\mathcal{C}_{\delta R}) = O_d(n^{\mu(\mathcal{C})^{\frac{d-1}{d}}}).$$

Proof. Let B be a minimum volume box with aspect ratio at most 2 with $\mu(\mathcal{C}_B) \geq (\frac{1+\epsilon}{3})\mu(\mathcal{C})$, whose side lengths are $l_1 \leq l_2 \leq .... \leq l_d, l_d \leq 2l_1$. Let s denote the center of B. For any m with $1 \leq m < 2^{\frac{1}{d}}$, let B_m be the box that is the magnified version of B by the magnification factor m. Thus, B_m is a box of side lengths $l_1^m \leq l_2^m \leq ... \leq l_d^m, l_d^m \leq 2l_1^m$, that has center s, and contains B. Note that $l_i^m \leq ml_i < 2^{\frac{1}{d}}l_i$, for $i = 1, 2, ..., d$, and hence the volume of B_m is strictly

less that 2 times the volume of B. By cutting B_m, in the middle of its longest side, we can decompose B into two boxes $B_m^i, i = 1, 2$, of aspect ratio at most two, each having a volume strictly smaller than volume of B. Thus, $\mu(\mathcal{C}_{B_m^i}) < (\frac{1+\epsilon}{3})\mu(\mathcal{C})$, $i = 1, 2$, since B has the minimum volume. Consequently, using (ii), we deduce that $\mu(\mathcal{C}_{B_m}) < \frac{2}{3}(1+\epsilon)\mu(\mathcal{C})$. Therefore, $\mu(\mathcal{C}_{B_m - \delta B_m}) < \frac{2}{3}(1+\epsilon)\mu(\mathcal{C})$. Next, let C_m be a cube of side length l_d^m having center s, and note that area of δ_{C_m} is $2.d{l_d^m}^{d-1} < 2^{\frac{2d-1}{d}}.d.{l_d}^{d-1} < 4.d.{l_d}^{d-1}$. Now, let $l = \frac{l_d}{8\mu(C)^{\frac{1}{d}}}$, and note that δ_{C_m}, and hence δ_{B_m} can be covered with at most $d.8^d.\mu(\mathcal{C})^{\frac{d-1}{d}}$ cubes of size l. Let $\mathcal{C}^1$ denote the set of all objects in $\mathcal{C}$ of size at least l, and let $1 \leq m < 2^{\frac{1}{d}}$. Then, by (iv), $\mu(\mathcal{C}^1_{\delta B_m}) \leq c.d.8^d.\mu(C)^{\frac{d-1}{d}}$. Similarly, let $\mathcal{C}^2$ denote the set of all objects in $\mathcal{C}$ of size strictly smaller than l. We need the following claim to finish the proof.

Claim. Let $1 \leq m_1 < m_2 < 2^{\frac{1}{d}}$, with $m_2 - m_1 \geq \frac{1}{\mu(\mathcal{C})^{\frac{1}{d}}}$, let $A_1 \in \mathcal{C}^2_{\delta B_{m_1}}$, and let $A_2 \in \mathcal{C}^2_{\delta_{B_{m_2}}}$. Then, $A_1 \cap A_2 = \emptyset$.

Justification. For any $x, y \in R^d$, let $distance(x, y)$ denote the distance between x and y. Note that $m_2 - m_1 \geq \frac{1}{\mu(\mathcal{C})^{\frac{1}{d}}}$ implies that for any two points $a \in \delta B_{m_2}$ and $b \in \delta B_{m_1}$, we must have $distance(a, b) \geq \frac{l_1}{2\mu(\mathcal{C})^{\frac{1}{d}}} \geq \frac{l_d}{4\mu(\mathcal{C})^{\frac{1}{d}}} \geq 2.l$. Assume to the contrary that $A_1 \cap A_2 \neq \emptyset$, and let $x \in A_1 \cap A_2$. Let $x_1 \in A_1 \cap \delta B_{m_1}$, and let $x_2 \in A_2 \cap \delta_{B_{m_2}}$. Note that $distance(x, x_1) < l$ and $distance(x, x_2) < l$, and thus, $distance(x_1, x_2) < 2.l$ which is a contradiction. □

Now use the claim and employ (ii) to conclude that

$$\sum_{j=0}^{\lfloor (2^{\frac{1}{d}}-1)\mu(\mathcal{C})^{\frac{1}{d}} \rfloor} \mu(\mathcal{C}^2_{\delta B_{1+j/\mu(\mathcal{C})^{\frac{1}{d}}}}) \leq \mu(\mathcal{C}).$$

It follows that there is a j so that for $m^* = 1 + \frac{j}{\mu(\mathcal{C})^{\frac{1}{d}}}$, we have $\mu(\mathcal{C}^2_{\delta B_{m^*}}) \leq \frac{\mu(C)^{\frac{d-1}{d}}}{2^{\frac{1}{d}}-1} \leq d^2\mu(C)^{\frac{d-1}{d}}$, since, $\frac{1}{2^{\frac{1}{d}}-1} \geq \frac{1}{d^2}$. We conclude that $\mu(\mathcal{C}_{\delta B_{m^*}}) \leq c.d^2 8^d.n^{\mu(\mathcal{C})^{\frac{d-1}{d}}}$. Now let $R = B_{m^*}$, and to finish the proof note that for $\mu(\mathcal{C}) \geq (\frac{3c.d^2.8^d}{\epsilon})^d$, we have $\mu(\mathcal{C}_{\delta R}) \leq \frac{\epsilon\mu(\mathcal{C}_R)}{3}$. □.

4 Sub-Exponential Time Algorithms

Our next result is the following.

Theorem 3 *Let $\mathcal{C}$ be a set of n fat objects in R^d, $d \geq 2$, then $Pack(\mathcal{C})$ and $Pierce(\mathcal{C})$ can be computed in $n^{O_d(Pack(\mathcal{C})^{\frac{d-1}{d}})}$ and $n^{O_d(Pierce(\mathcal{C})^{\frac{d-1}{d}})}$, respectively, and $O(n \log n)$ storage.*

Proof. We provide the details for computing packing number; The details relevant to the computation of piercing number are similar, but slightly different. We use the following recursive algorithm adopted from [14] and tailored to our needs, which we previously utilized in [17].

Step 1. Determine an approximate solution μ to the packing number using (iv). Choose an ϵ, let $\alpha = \frac{2}{3}(1+\epsilon)$ for Theorem 2, and let $C = C(d)$ be the Lower bound on $\mu(\mathcal{C})$ in Theorem 2. Test using (v) to determine if $Pack(\mathcal{C}) \leq C$. If so, then compute a solution in $n^{O(C)}$ time and return it. If not, proceed to the recursive step.

Step 2 (Recursive Step). Find box R by applying Theorem 2 to μ. For any independent set of objects $\mathcal{I}$ in $\mathcal{C}_{\delta B}$ compute $Pack(\mathcal{C}_B - N(\mathcal{I}))$, $Pack(\mathcal{C}_{\bar{B}} - N(\mathcal{I}))$, and return

$$\max_{\mathcal{I}}\{Pack(\mathcal{C}_B - N(\mathcal{I}) + Pack(\mathcal{C}_{\bar{B}} - N(\mathcal{I})) + |\mathcal{I}|\}$$

where the maximum is taken overall independent sets of objects $\mathcal{I}$ in $\mathcal{C}_{\delta B}$.

It is easy to verify that the algorithm computes $Pack(\mathcal{C})$ correctly. For the running time, let $T(n, p)$ denote the execution time of the algorithm on an instance of the problem with $Pack(\mathcal{C}) = p$. Note that, $T(n, p) \leq n^{O(p^{\frac{d-1}{d}})}T(n, (\alpha p))$, if $p > C$; Otherwise $T(n, p) \leq n^{O(C)}$. It is not difficult to verify that $T(n, p) = n^{O_d(p^{\frac{d-1}{d}})}$ as claimed. Similarly, one can verify the claim concerning the storage. □

5 Approximation Algorithms

The algorithms in the previous section gives rise to polynomial time approximation schemes (PTAS) for both of stated problems with $n^{O((\frac{1}{\epsilon})^{d-1})}$ time and $O(n \log n)$ storage. For the PTAS, one can slightly modify the original divide and conquer approach in [13].

Specifically, one must first obtain a constant time approximation solution to the problem using (v), use it to define the measure μ, and apply the separation theorem, or Theorem 2, to this μ. In the recursive step, the divide and conquer algorithm stops when the value of the measure is *small*. That is, if the value of the measure is $O((\frac{1}{\epsilon})^2)$, then an exact solution is computed by the application of our sub-exponential time algorithm. The claims concerning the running time, storage, and quality of the approximation are not difficult to verify.

References

[1] Agarwal, P.K., Kreveld, M., Suri, S., Label placement by maximum independent sets in rectangles, Comput. Geometry:Theory and Appl. 11(3-4) 209-218, 1998.

[2] Alber J., Fiala J., Geometric Separation and Exact Solutions for the Parameterized Independent Set Problem on Disk Graphs,Journal of Algorithms, Volume 52 , Issue 2, 134 - 151, 2004

[3] N. Alon, P. Seymour, and R. Thomas, A separator theorem for graphs with an excluded minor and its applications, Proceedings 22nd ACM Symposium on the Theory of Computing, STOC90, 1990, 293-299.

[4] Chan T., Polynomial-time approximation schemes for packing and piercing fat objects , Journal of Algorithms, 46(2), 178 - 189, 2003.

[5] Erlebach T., Jansen K. and Seidel E., Polynomial-time approximation schemes for geometric graphs. Proc. 12th ACM-SIAM Symposium on Discrete Algorithms (SODA'01), 671-679, 2001.

[6] Fox J., Pach J., A separator theorem for string graphs and its applications, Combinatorics, Probability and Computing 19(2010), 371-390.

[7] Fox J., Pach J., Separator theorems and Turn-type results for planar intersection graphs, Advances in Mathematics 219, 1070-1080, 2008.

[8] Fox J., Pach J., Coloring kk-free intersection graphs of geometric objects in the plane, Proceedings of the twenty-fourth annual symposium on Computational geometry, 346-354, 2008.

[9] Fox J, Pach J, Tóth C.D, Intersection patterns of curves, J. London Math. Soc. (2008)

[10] J. Fox and J. Pach, Computing the independence number of intersection graphs. SODA 2011, 1161-1165.

[11] Harry B. Hunt III, Marathe M. V., Radhakrishnan V., Ravi S. S, Rosenkrantz D. J., and Stearns R. E.. A Unified Approach to Approximation Schemes for NP- and PSPACE-Hard Problems for Geometric Graphs. Journal of Algorithms, 26, 135-149, 1996.

[12] Hochbaum D.S., Maass W., Approximation Schemes for Covering and Packing Problems in Image Processing and VLSI. Journal of the Association for Computing Machinery, 32(1), 130-136, 1985.

[13] Lipton, R.J., Tarjan, R.E., Applications of a planar separator theorem, SIAM J. Comput., 9(3),615-628, 1980.

[14] Lipton, R.J., Tarjan R.E, A separator theorem for planar graphs, SIAM Journal on Applied Mathematics 36, 1979, 177-189. .

[15] Miller, G.L., Teng, S., Thurston W., and Vavasis, S. A., Separators for Sphere-Packings and Nearest Neighborhood graphs, JACM, 44(1), 1-29, 1997

[16] Smith and N.C. Wormald, Geometric separator theorems and applications. In 39th Annual Symposium on Foundations of Computer Science: FOCS '98, pages 232-243, Palo Alto, CA, 1998.

[17] Shahrokhi F., A New Separation Theorem with Geometric Applications, EuroCG2010.

Refold Rigidity of Convex Polyhedra

Erik D. Demaine* Martin L. Demaine† Jin-ichi Itoh‡ Anna Lubiw§ Chie Nara¶
Joseph O'Rourke‖

Abstract

We show that every convex polyhedron may be unfolded to one planar piece, and then refolded to a different convex polyhedron. If the unfolding is restricted to cut only edges of the polyhedron, then we show that many regular and semi-regular polyhedra are "edge-unfold rigid" in the sense that each of their unfoldings may only fold back to the original. For example, all of the 43,380 edge unfoldings of a dodecahedron may only fold back to the dodecahedron. We begin the exploration of which polyhedra are edge-unfold rigid, demonstrating infinite rigid classes through perturbations, and identifying one infinite nonrigid class: tetrahedra.

(The full version of this paper is available.[1])

1 Introduction

It has been known since [5] and [3] that there are convex polyhedra, each of which may be unfolded to a planar polygon and then refolded to different convex polyhedra. For example, the cube may be unfolded to a "Latin cross" polygon, which may be refolded to 22 distinct non-cube convex polyhedra [4, Figs. 25.32-6]. But there has been only sporadic progress on understanding which pairs of convex polyhedra[2] have a common unfolding. A notable recent exception is the discovery [7] of an unfolding of a cube that refolds to a regular tetrahedron, partially answering Open Problem 25.6 in [4, p. 424].

Here we begin to explore a new question, which we hope will shed light on the unfold-refold spectrum of problems: Which polyhedra $\mathcal{P}$ are *refold-rigid* in the sense that any unfolding of $\mathcal{P}$ may only be refolded back to $\mathcal{P}$? The answer we provide here is: NONE—Every polyhedron $\mathcal{P}$ has an unfolding that refolds to an incongruent $\mathcal{P}'$. Thus every $\mathcal{P}$ may be *transformed* to some $\mathcal{P}'$.

This somewhat surprising answer leads to the next natural question: Suppose the unfoldings are restricted to *edge unfoldings*, those that only cut along edges of $\mathcal{P}$ (rather than permitting arbitrary cuts through the interior of faces). Say that a polyhedron $\mathcal{P}$ whose every edge unfolding only refolds back to $\mathcal{P}$ is *edge-unfold rigid*, and otherwise is an *edge-unfold transformer*. It was known that four of the five Platonic solids are edge-unfold transformers (e.g., [2] and [6]). Here we prove that the dodecahedron is edge-unfold rigid: all of its edge unfoldings only fold back to the dodecahedron. The proof also demonstrates edge-unfold rigidity for 11 of the Archimedean solids. We also establish the same rigidity for infinite classes of slightly perturbed versions of these polyhedra. In contrast to this, we show that every tetrahedron is an edge-unfold transformer: at least one among a tetrahedron's 16 edge unfoldings refolds to a different polyhedron.

This work raises many new questions, summarized in Section 6.

2 Notation and Definitions

We will use $\mathcal{P}$ for a polyhedron in $\mathbb{R}^3$ and P for a planar polygon. An *unfolding* of a polyhedron $\mathcal{P}$ is development of its surface after cutting to a single (possibly overlapping) polygon P in the plane. The surface of $\mathcal{P}$ must be cut open by a spanning tree to achieve this. An *edge-unfolding* only includes edges of $\mathcal{P}$ in its spanning cut tree. Note that we do not insist that unfoldings avoid overlap.

A *folding* of a polygon P is an identification of its boundary points that satisfies the three conditions of Alexandrov's theorem: (1) The identifications (or "gluings") close up the perimeter of P without gaps; (2) The resulting surface is homeomorphic to a sphere; and (3) Identifications result in $\leq 2\pi$ angle glued at every point. Under these three conditions, Alexandrov's theorem guarantees that the folding produces a convex polyhedron, unique once the gluing is specified. See [1] or [4]. Note that there is no restriction that whole edges of P must be identified to whole edges, even when P is produced by an edge unfolding. We call a gluing that satisfies the above conditions an *Alexandrov gluing*.

A polyhedron $\mathcal{P}$ is *refold-rigid* if every unfolding of $\mathcal{P}$ may only refold back to $\mathcal{P}$. Otherwise, $\mathcal{P}$ is

*MIT, edemaine@mit.edu
†MIT, mdemaine@mit.edu
‡Kumamoto Univ., j-itoh@kumamoto-u.ac.jp
§Univ. Waterloo, alubiw@uwaterloo.ca
¶Tokai Univ., cnara@ktmail.tokai-u.jp
‖Smith College, orourke@cs.smith.edu

[1] http://cs.smith.edu/~orourke/Papers/RefoldRigidEuroCGFull.pdf.

[2] All polyhedra in this paper are convex, and the modifier will henceforth be dropped.

a *transformer*. A polyhedron is *edge-unfold rigid* if every edge unfolding of $\mathcal{P}$ may only refold back to $\mathcal{P}$, and otherwise it is an *edge-unfold transformer*.

3 Polyhedra Are Transformers

The proof that no polyhedron $\mathcal{P}$ is refold rigid breaks naturally into two cases. We first state a lemma that provides the case partition. Let $\kappa(v)$ be the curvature at vertex $v \in \mathcal{P}$, i.e., the "angle gap" at v: 2π minus the total incident face angle $\alpha(v)$ at v. By the Gauss-Bonnet theorem, the sum of all vertex curvatures of $\mathcal{P}$ is 4π.

Lemma 1 *For every polyhedron $\mathcal{P}$, either there is a pair of vertices with $\kappa(a) + \kappa(b) > 2\pi$, or there are two vertices each with at most π curvature: $\kappa(a) \le \pi$ and $\kappa(b) \le \pi$.*

Proof. Suppose there is no pair with curvature sum more than 2π. So we have $\kappa(v_1) + \kappa(v_2) \le 2\pi$ and $\kappa(v_3) + \kappa(v_4) \le 2\pi$ for four distinct vertices. Suppose neither of these pairs have both vertices with at most π curvature. If $\kappa(v_2) > \pi$, then $\kappa(v_1) \le \pi$; and similarly, if $\kappa(v_4) > \pi$, then $\kappa(v_3) \le \pi$. Thus we have identified two vertices, v_1 and v_3, both with at most π curvature. □

We can extend this lemma to accommodate 3-vertex doubly covered triangles as polyhedra, because then every vertex has curvature greater than π.

Lemma 2 *Any polyhedron $\mathcal{P}$ with a pair of vertices with curvature sum more than 2π is not refold-rigid: There is an unfolding that may be refolded to a different polyhedron $\mathcal{P}'$.*

Proof. Let $\kappa(a) + \kappa(b) > 2\pi$, and so the incident face angles satisfy $\alpha(a) + \alpha(b) < 2\pi$. Let γ be a shortest path on $\mathcal{P}$ connecting a to b. Cut open $\mathcal{P}$ with a cut tree T that includes γ as an edge. How T is completed beyond the endpoints of $\gamma = ab$ doesn't matter.

Let γ_1 and γ_2 be the two sides of the cut γ, and let m_1 and m_2 be the midpoints of γ_1 and γ_2. Reglue the unfolding by folding γ_1 at m_1 and gluing the two halves of γ_1 together, and likewise fold γ_2 at m_2. All the remaining boundary of the unfolding outside of γ is reglued back exactly as it was cut by T.

The midpoint folds at m_1 and m_2 have angle π (because γ is a geodesic). The gluing draws the endpoints a and b together, forming a point with total angle $\alpha(a) + \alpha(b) < 2\pi$. Thus this gluing is an Alexandrov gluing, producing some polyhedron $\mathcal{P}'$. Generically $\mathcal{P}'$ has one more vertex than $\mathcal{P}$: it gains two vertices at m_1 and m_2, and a and b are merged to one. $\mathcal{P}'$ could only have the same number of vertices as $\mathcal{P}$ if $\alpha(a) + \alpha(b) = 2\pi$, which is excluded in this case. □

Lemma 3 *Any polyhedron $\mathcal{P}$ with a pair of vertices each with curvature at most π is not refold-rigid: There is an unfolding that may be refolded to a different polyhedron $\mathcal{P}'$.*

Proof. Let a and b be a pair of vertices with $\kappa(a) \le \pi$ and $\kappa(b) \le \pi$, and so $\alpha(a) \ge \pi$ and $\alpha(b) \ge \pi$. Let $\gamma = ab$ be a shortest path from a to b on $\mathcal{P}$. Because the curvature at each endpoint is at most π, there is at least π surface angle incident to a and to b. This permits identification of a rectangular neighborhood R on $\mathcal{P}$ with midline ab, whose interior is vertex-free.

Now we select a cut tree T that includes ab and otherwise does not intersect R. This is always possible because there is at least π surface angle incident to both a and b. So we could continue the path beyond ab to avoid cutting into R. Let T unfold $\mathcal{P}$ to polygon P. We will modify T to a new cut tree T'.

Replace ab in T by three edges ab', $b'a'$, $a'b$, forming a zigzag 'Z'-shape, $Z = ab'a'b$, with $Z \subset R$. We will illustrate with an unfolding of a cube, shown in Fig. 1, with ab the edge cut between the front (F) and top (T) faces of the cube.

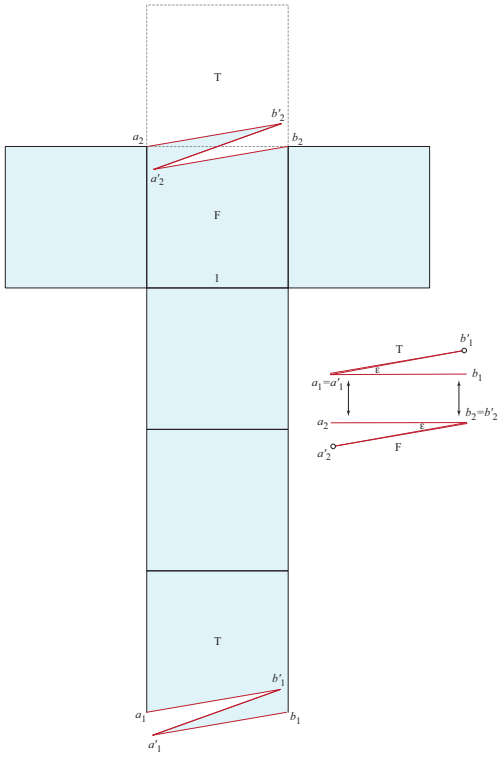


Figure 1: Unfolding of a unit cube. The cut edge ab is replaced by $Z = ab'a'b$. The unfolding P' is illustrated. The insert shows the gluing in the vicinity of ab in the refolding to $\mathcal{P}'$.

We select an angle ε determining the Z according to two criteria. First, ε is smaller than either $\kappa(a)$ and $\kappa(b)$. Second, ε is small enough so that the following construction sits inside R. Let $R' \subset R$ be a rectangle

whose diagonal is ab; refer to Fig. 2. Trisect the left

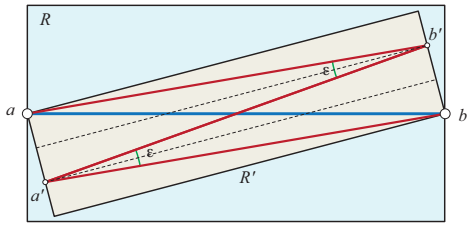


Figure 2: Construction of zig-zag path Z.

and right sides of R', and place a' and b' two-thirds away from a and b respectively. The angle of the Z at a' and at b' is ε. $\triangle ab'a'$ and $\triangle ba'b'$ are congruent isosceles triangles; so $|ab'| = |b'a'| = |a'b|$.

The turn points a' and b' have curvature zero on $\mathcal{P}$ (because $Z \subset R$ and R is vertex-free). Let P' be the polygon obtained by unfolding $\mathcal{P}$ by cutting T', and label the pair of images of each Z corner $a_1, a'_1, \ldots, b_2$, as illustrated in Fig. 1. Now we refold it differently, to obtain a different polyhedron $\mathcal{P}'$. "Zip" P' closed at the reflex vertices a'_2 and b'_1. Zipping at a'_2 glues $a'_2b'_2$ to a'_2b_2, so that now $b_2 = b'_2$; zipping at b'_1 glues $b'_1a'_1$ to b'_1a_1, so that now $a_1 = a'_1$. (See the insert of Fig. 1.) Finally, the two "halves" of the new $a'_1b'_1 = a_2b_2$ are glued together, and the remainder of T' is reglued just as it was in T.

This gluing produces new vertices near a' and b', each of curvature $\kappa(a') = \kappa(b') = \varepsilon$. An extra ε of surface angle is glued to both a and b, so their curvatures each decrease by ε (and so maintain the Gauss-Bonnet sum of 4π). By the choice of ε, these curvatures remain positive. Alexandrov's theorem is satisfied everywhere: the curvatures at a, b, a', b' are all positive, and the lengths of the two halves of the new $a'_1b'_1 = a_2b_2$ edge are the same (and note this length is not the original length of ab on $\mathcal{P}$, but rather the side-length of the isosceles triangles: $|ab'| = |b'a'| = |a'b|$). So this refolding corresponds to some polyhedron $\mathcal{P}'$. It is different from $\mathcal{P}$ because it has two more vertices at a' and b' (vertices at a and b remain with some positive curvature by our choice of ε). □

Putting Lemmas 2 and 3 together yields the claim:

Theorem 4 *Every polyhedron has an unfolding that refolds to a different polyhedron, i.e., no polyhedron is refold-rigid.*

4 Many (Semi-)Regular Polyhedra are Edge-Unfold Rigid

Our results on edge-unfold rigidity rely on this theorem:

Theorem 5 *Let $\theta_{\min}$ be the smallest angle of any face of $\mathcal{P}$, and let $\kappa_{\max}$ be the largest curvature at any vertex of $\mathcal{P}$. If $\theta_{\min} > \kappa_{\max}$, then $\mathcal{P}$ is edge-unfold rigid.*

Proof. Let T be an edge-unfold cut tree for $\mathcal{P}$, and P the resulting unfolded polygon. No angle on the boundary of P can be smaller than $\theta_{\min}$. Let x be a leaf node of T and y the parent of x. The exterior angle at x in the unfolding P is at most $\kappa_{\max}$. Because every internal angle of P is at least $\theta_{\min}$, which is larger than $\kappa_{\max}$, no point of P can be glued into x, leaving the only option to "zip" together the two cut edges deriving from $xy \in T$. Let $T' = T \setminus xy$ be the cut tree remaining after this partial gluing, and P' the partially reglued manifold.

If T' is not empty, it is a tree, with at least two leaves, one of which might be y (if x was the only child of y). Any leaf $z \in T'$ corresponds to some vertex $v \in \mathcal{P}$, with all but one incident edge already glued. Because P' has not gained any new angles beyond those available in P, we have returned to the same situation: no angle of P' is small enough to fit into the angle gap at z, which is at most $\kappa_{\max}$ at any v. Thus again the edge of T' incident to z must be zipped in the gluing. Continuing in this manner, we see that T may only be reglued by exactly identifying every cut-edge pair, reproducing $\mathcal{P}$. □

Corollary 6 *The regular and semi-regular solids that satisfy Theorem 5, listed in Table 4, are all edge-refold rigid.*

Corollary 7 *Any polyhedron $\mathcal{P}$ that satisfies Theorem 5, may be "perturbed" by moving its vertices slightly to create an uncountable number of edge-refold rigid polyhedra.*

Proof. Proof omitted. □

5 Tetrahedra are edge-unfold transformers

The goal of this section is to prove this theorem:

Theorem 8 *Every tetrahedron may be edge-unfolded and refolded to a different polyhedron.*

There are 16 distinct edge unfoldings of a tetrahedron $\mathcal{T}$. The spanning cut trees that determine these unfoldings fall into just two combinatorial types: The cut tree is a star, a Y-shaped "trident" with three leaves, or the cut tree is a path of three edges. There are 4 different tridents, and $2 \cdot \binom{4}{2} = 12$ different paths. In all these unfoldings, the polygon P that constitutes the unfolded surface is a hexagon: the three cut edges becomes three pairs of equal-length edges of the hexagon. Our goal is to show that, for any $\mathcal{T}$, at least one of the 16 unfoldings P may be refolded to a polyhedron $\mathcal{P}'$ not congruent to $\mathcal{T}$.

Polyhedron Name	$\theta_{\min}$	$\kappa_{\max}$	$\theta_{\min} > \kappa_{\max}$
Dodecahedron	$\frac{3}{5}$	$\frac{1}{5}$	✓
Trunc. Cube	$\frac{1}{3}$	$\frac{1}{6}$	✓
Rhombicuboctahedron	$\frac{1}{3}$	$\frac{1}{6}$	✓
Trunc. Cuboctahedron	$\frac{1}{2}$	$\frac{1}{12}$	✓
Snub Cube	$\frac{1}{3}$	$\frac{1}{6}$	✓
Icosidodecahedron	$\frac{1}{3}$	$\frac{2}{15}$	✓
Trunc. Dodecahedron	$\frac{1}{3}$	$\frac{1}{15}$	✓
Trunc. Icosahedron	$\frac{3}{5}$	$\frac{1}{15}$	✓
Rhomb-icosidodecahedron	$\frac{1}{3}$	$\frac{1}{15}$	✓
Trunc. Icosidodecahedron	$\frac{1}{2}$	$\frac{1}{30}$	✓
Snub Dodecahedron	$\frac{1}{3}$	$\frac{1}{15}$	✓
Pseudo-rhombicuboctahedron	$\frac{1}{3}$	$\frac{1}{6}$	✓

Table 1: Inventory of minimum face angles and maximum vertex curvatures, for selected regular and semi-regular polyhedra. All angles expressed in units of π.

Proof. (of Theorem 8). The proof classifies tetrahedra by their four curvatures, and then establishes the claim for each of the resulting four classes. A concrete example of one of the classes, a *2r-tetrahedron* with $\kappa_1 \geq \kappa_2 \geq 1 > \kappa_3 \geq \kappa_4$, is shown in Fig. 3. Proof omitted. □

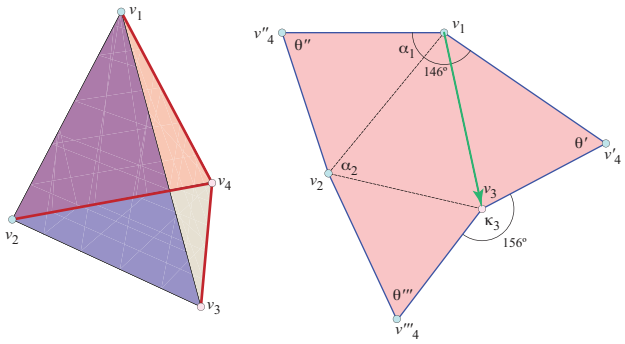


Figure 3: A tetrahedron with v_1, v_2 "convex" and v_3, v_4 "reflex," cut open with a trident rooted at v_4, producing a hexagon with one reflex vertex with exterior angle κ_3. The proof shows that the convex angle α_1 derived from v_1 fits within κ_3.

6 Open Problems

Our work so far just scratches the surface of a potentially rich topic. Here we list some questions suggested by our investigations.

1. Star unfoldings (e.g., [4, Sec. 24.3]) are natural candidates for rigidity. Is it the case that almost every star unfolding of (almost?) every polyhedron is refold-rigid?
2. Which (if either) of the following is true? (a) Almost all polyhedra are edge-unfold rigid. (b) Almost all polyhedra are edge-unfold transformers.
3. Characterize the polygons P that can fold in two different ways (have two different Alexandrov gluings) to produce the exact same polyhedron $\mathcal{P}$. We have only sporadic examples of this phenomenon (among the foldings of the Latin cross).
4. Do our transformer results extend to the situation where the unfoldings are required to avoid overlap? We can extend Lemma 2 to ensure nonoverlap, but extending Lemma 3 seems more difficult.
5. One could view an edge-unfold and refold operation as a directed edge between two polyhedra in the space of all convex polyhedra. Thm. 5 and Cor. 7 show neighbors of some (semi-)regular polyhedra have no outgoing edges. What is the connected component structure of this space?

References

[1] A. D. Alexandrov. *Convex Polyhedra*. Springer-Verlag, Berlin, 2005. Monographs in Mathematics. Translation of the 1950 Russian edition by N. S. Dairbekov, S. S. Kutateladze, and A. B. Sossinsky.

[2] E. Demaine, M. Demaine, A. Lubiw, A. Shallit, and J. Shallit. Zipper unfoldings of polyhedral complexes. In *Proc. 22nd Canad. Conf. Comput. Geom.*, pages 219–222, Aug. 2010.

[3] E. D. Demaine, M. L. Demaine, A. Lubiw, J. O'Rourke, and I. Pashchenko. Metamorphosis of the cube. In *Proc. 15th Annu. ACM Sympos. Comput. Geom.*, pages 409–410, 1999. Video and abstract.

[4] E. D. Demaine and J. O'Rourke. *Geometric Folding Algorithms: Linkages, Origami, Polyhedra*. Cambridge University Press, July 2007. `http://www.gfalop.org`.

[5] A. Lubiw and J. O'Rourke. When can a polygon fold to a polytope? Technical Report 048, Dept. Comput. Sci., Smith College, June 1996. Presented at *Amer. Math. Soc.* Conf., 5 Oct. 1996.

[6] J. O'Rourke. Flat zipper-unfolding pairs for Platonic solids. `http://arxiv.org/abs/1010.2450`, Oct. 2010.

[7] T. Shirakawa, T. Horiyama, and R. Uehara. Construction of common development of regular tetrahedron and cube. In *EuroCG*, 2011.

How Many Potatoes are in a Mesh?

Marc van Kreveld* Maarten Löffler* János Pach†

Abstract

We consider the combinatorial question of how many convex polygons can be made at most by using the edges taken from a fixed triangulation. For general triangulations there can be exponentially many: $\Omega(1.5028^n)$ and $O(1.62^n)$ in the worst case. If the triangulation is fat (every triangle has its angles lower-bounded by a constant $\delta > 0$), then there can only be polynomially many: $\Omega(n^{\frac{1}{2}\lfloor\frac{2\pi}{\delta}\rfloor})$ and $O(n^{\lceil\frac{\pi}{\delta}\rceil})$ in the worst case.

1 Introduction

In combinatorial geometry, a common objective is to search for the minimum and maximum count of some geometric substructure within a geometric structure of a given type. Well-known examples are the number of vertices in the lower envelope or single face in an arrangement of line segments, the number of triangulations that have a given set of points as their vertices, the number of unit-distance pairs in a set of points, etc. [8]. In this paper we analyze how many convex polygons (potatoes) can be constructed by taking unions of triangles taken from a fixed triangulation (mesh) with n vertices. Equivalently, we analyze how many convex polygon boundaries can be made using the edges of a fixed triangulation, see Figure 1. For general triangulations there can be exponentially many. However, the lower-bound examples use many triangles with very small angles. When $n \to \infty$, the smallest angles tend to zero. To understand whether this is necessary, we also study the number of convex polygons in a triangulation where all angles are bounded from below by a fixed constant. It turns out that the maximum number of convex polygons is always polynomial in this case.

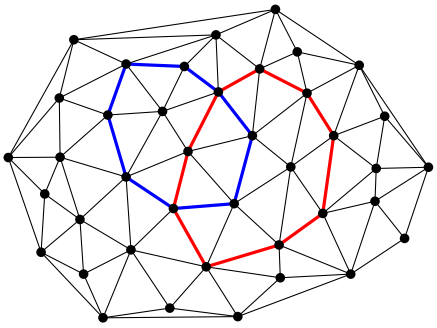


Figure 1: A triangulation T. Two convex polygons that respect T are marked.

In computational geometry, *realistic input models* have received considerable attention in the last decades. By making assumptions on the input, many computational problems can be solved provably faster than what is possible without these assumptions. One of the early examples concerned fat triangles: a triangle is δ-fat if each of its angles is at least δ, for some fixed constant $\delta > 0$. Matousek *et al.* [6] show that the union of n δ-fat triangles has complexity $O(n \log\log n)$, while for n general triangles this is $\Omega(n^2)$. As a consequence, the union of fat triangles can be computed more efficiently as well. Several other realistic input models have been introduced since then.

In [1, 4, 5, 7], *fat triangulations* were used as a realistic input model motivated by polyhedral terrains, sometimes with extra assumptions. Fat triangulations are related to the meshes computed in the area of high-quality mesh generation. The smallest angle of the elements of the mesh is a common quality measure [3]. In graph drawing, the concept of *angular resolution* is also directly related to fatness.

In this paper we show that the maximum number of convex polygons in a triangulation can be as large as $\Omega(1.5028^n)$, and we give an upper bound of $O(1.62^n)$. If the triangulation is δ-fat, there can be $\Omega(n^{\frac{1}{2}\lfloor\frac{2\pi}{\delta}\rfloor})$ convex polygons, and we show an upper bound of $O(n^{\lceil\frac{\pi}{\delta}\rceil})$. It is interesting (albeit not surprising) to see a case where fatness reduces the combinatorial complexity from exponential to polynomial. This paper is also loosely motivated by Aronov et al. [2], where the algorithmic problem of finding a largest-area convex polygon in a given triangulation is studied.

*Department of Information and Computing Sciences, Utrecht University, the Netherlands, m.j.vankreveld@uu.nl, m.loffler@uu.nl. M.L. is supported by the Netherlands Organisation for Scientific Research (NWO) under grant no. 639.021.123.

†Ecole Polytechnique Féderale de Lausanne and City College of New York, pach@cims.nyu.edu. Supported by NSF Grant CCF-08-30272, by NSA, by OTKA under EUROGIGA project GraDR 10-EuroGIGA-OP-003, and by Swiss National Science Foundation Grant 200021-125287/1.

2 Preliminaries

A *triangulation* is a maximal plane straight-line graph with a finite set of vertices. All bounded faces are triangles, the interiors of all triangles are disjoint and the intersection of any pair of triangles is either a vertex or a shared edge. We also denote the set of vertices of a graph G by $V(G)$ and the set of edges by $E(G)$, and say that the *size* of G is $n = |V(G)|$. A polygon P is said to *respect* a graph G if its boundary uses edges from G only.

A triangulation T is called δ-*fat*, for some $\delta \in (0, \frac{2}{3}\pi]$, if every angle of every triangle of T is at least δ.

Let $S = [0, 2\pi)$. We define cyclic addition and subtraction $(+, -) : S \times \mathbb{R} \to S$ in the obvious way. We call elements of S *directions* and implicitly associate an element $s \in S$ with the vector $(\sin s, \cos s)$.

3 Lower bound in general triangulations

We place m points evenly spaced on the upper half of a circle. Assume $m = 2^k + 1$ for some integer k, and let the points be $v_0, \ldots, v_{m-1}$, clockwise. To triangulate this point set we add the convex hull edges, then connect v_0 and v_{m-1} to $v_{(m-1)/2}$, and recursively triangulate the subpolygons by always connecting the furthest pair to the midpoint. The dual of the triangulation is a perfectly balanced binary tree.

Let $N(k)$ be the number of different convex paths from v_0 to v_{m-1}. Then we have

$$N(k) = 1 + (N(k-1))^2, \quad N(1) = 2$$

because we can combine every path from v_0 to $v_{(m-1)/2}$ with every path from $v_{(m-1)/2}$ to v_{m-1}, and the extra path is $\overline{v_0, v_{m-1}}$ itself. Using this recurrence we can relate the number m of vertices used to the number $P(m)$ of convex paths obtained: $P(3) = 2$; $P(5) = 5$; $P(9) = 26$; $P(17) = 677$.

Now we place n points evenly spaced on the upper half of a circle. We triangulate $v_0, \ldots, v_{16}$ as above, and also $v_{16} \ldots, v_{32}$, and so on. We can make $n/16$ groups of 17 points where the first and last point of each group are the same. Each group is triangulated to give 677 convex paths; the rest is triangulated arbitrarily. In total we get $677^{n/16} = \Omega(1.5028^n)$ convex paths from v_0 to v_{n-1}. We omit the one from v_0 directly to v_{n-1}, and use this edge to complete every convex path to a convex polygon instead. The number of convex polygons is $\Omega(1.5028^n)$.

Theorem 1 *There exists a triangulation T with n vertices such that the number of convex polygons that respect T is $\Omega(1.5028^n)$. This is true even if T is the Delaunay triangulation of its vertices.*

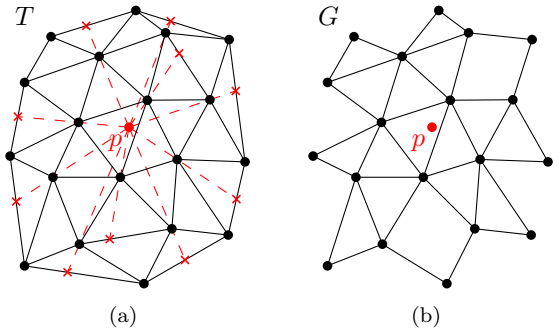


Figure 2: (a) We project each interior vertex of T from p onto the next edge. (b) The graph G obtained by removing the marked edges.

4 Upper bound for general triangulations

Let T be any triangulation with n vertices. First, fix a point p inside some triangle of T. We will count only the polygons that contain p for now.

For every vertex v not on the convex hull of T, let e_v be the first edge that is hit by a ray leaving v on the line through p and v, away from p. (Assume general position of p w.r.t. the vertices.) Let G be the graph obtained from T by removing all such edges. Figure 2 shows an example. G now has exactly $2n - 3$ edges (because every vertex not on the convex hull causes one edge to disappear).

A cycle in G is *monotone around* p if it is star-shaped with p in its kernel.

Lemma 2 *The number of convex polygons in T that contain p is bounded from above by the number of monotone cycles around p in G.*

Proof. With each convex polygon, we associate a monotone cycle by replacing any edges e_v that were removed by the two edges via v, recursively. This results in a proper cycle because the convex polygon was a monotone cycle in T, and this property is maintained. Each convex polygon results in a different cycle because the angle from the vertices of e_v via v is always concave. □

We turn G into a directed graph by orienting every edge such that p lies to the left of its supporting line. We are interested in the number of simple cycles that respect G; these are the monotone cycles around p.

Claim 1 *The complement of the outer face of G is star-shaped with p in its kernel.*

Claim 2 *Let e be an edge on the outer face of G from u to v. Then either u has outdegree 1, or v has indegree 1 (or both).*

If $F \subset E(G)$ is a subset of the edges of G, we also consider the subproblem of counting all simple cycles in G that use all edges in F, the *fixed* edges. For a tuple (T, G, F), we define the *potential* ρ to be the number of vertices of T (or G) minus the number of edges in F, i.e., $\rho(T, G, F) = |V(G)| - |F|$. Clearly, the potential of a subproblem is an upper bound on the number of edges that can still be used in any simple cycle.

We will now show that the number of cycles in a subproblem can be expressed in terms of subproblems of smaller potential. Let $Q(k)$ be the maximum number of simple cycles in any subproblem with potential k.

Lemma 3 *The function $Q(\cdot)$ satisfies*

$$Q(k) \leq Q(k-1) + Q(k-2), \quad Q(0) = Q(1) = 1\,.$$

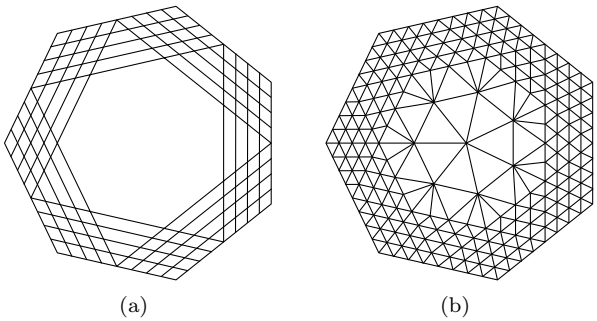
Proof. Let (T, G, F) be a subproblem and let $k = \rho(T, G, F)$. If $k = 1$ then $|F| = |V(G)| - 1$, so the number of fixed edges on the cycle is one less than the number of vertices available. Therefore the last edge is also fixed, if any cycle is possible. If $k = 0$, all edges are fixed.

For the general case, suppose all edges on the outer face of G are fixed. Then there is only one possible cycle. If any vertex on the outer face has degree 2 and only one incident edge fixed, we fix the other incident edge too. Suppose there is at least one edge, $e = \overline{uv}$, on the outer face that is not fixed. By Claim 2 one of its neighbours must have degree 1 towards e. Assume without loss of generality that this is v. We distinguish two cases.

(i) The degree of v is 2. Any cycle in G either uses v or does not use v. If it does not use v we have a subproblem of potential $k-1$. If it uses v, it must also use its two incident edges, so we can include these edges in F to obtain a subproblem of potential $k-2$. So, the potential $\rho(T, G, F) \leq Q(k-1) + Q(k-2)$.

(ii) The degree of v is larger than 2. Any cycle in G either uses e or does not use e. If it uses e, we can add e to F to obtain a subproblem of potential $k-1$. If it does not use e, then consider v and the edge $e' = \overline{vw}$ that leaves v on the outer face. Since v has indegree 1 but total degree greater than 2, it must have outdegree greater than 1. Therefore, by Claim 2, w must have indegree 1. Therefore, also w will not be used by any cycle in G that does not use e, and we can remove v and w to obtain a smaller graph. Again, we also remove all incident edges; if any of them was fixed we have no solutions. We obtain a subproblem of potential $k-2$ in this case. Again, the potential $\rho(T, G, F) \leq Q(k-1) + Q(k-2)$. □

This expression grows at a rate of the root of $x^2 - x - 1 = 0$, which is approximately 1.618034.



(a) (b)

Figure 3: (a) Essential part of the construction, allowing l^k choices for a convex polygon. (b) Final triangulation.

Because every convex polygon must contain at least one triangle of T, we just place p in each triangle and multiply the bound multiply by $2n$. Since 1.62 is a slight overestimate (by rounding) of the root, we can ignore the factor $2n$ in the bound.

Theorem 4 *Any triangulation T with n vertices has $O(1.62^n)$ convex polygons that respect T.*

5 Lower bound in fat triangulations

Let $k = \lfloor \frac{2\pi}{\delta} \rfloor$, and let $l = \sqrt{\frac{n}{2k}}$. Let Q be a regular k-gon, and for each edge e of Q consider the intersection point of the supporting lines of the neighbouring edges. Let Q' be a scaled copy of Q that goes through these points. Now, consider a sequence $Q = Q_1, Q_2, \ldots, Q_l = Q'$ of l scaled copies of Q such that the difference between the radii of consecutive copies is always equal. We extend the edges of each copy until they touch Q'. Figure 3(a) illustrates the construction so far.

We claim that the graph constructed so far allows at least l^k different convex polygons.

We now add vertices and edges to turn the construction into a δ-fat triangulation. We use $\binom{l-1}{2}$ more vertices per sector, placing $l - i$ vertices on each edge of Q_i to ensure all angles are bounded by δ. We need $O(lk)$ vertices to triangulate the interior using some adaptive mesh generation method. The final triangulation can be seen in Figure 3(b).

The construction uses $\frac{3}{2}kl^2 + O(kl)$ vertices, and since we have $l = \sqrt{\frac{n}{2k}}$, there are $\frac{3}{2}kl^2 + O(kl) = \frac{3}{2}k\frac{n}{2k} + O(k\sqrt{\frac{n}{2k}}) = \frac{3}{4}n + O(\sqrt{nk}) \leq n$ vertices in total.

Observe that the triangles of the outer ring are Delaunay triangles. The inner part can also be triangulated with Delaunay triangles, since the Delaunay triangulation maximises the smallest angle of any triangle.

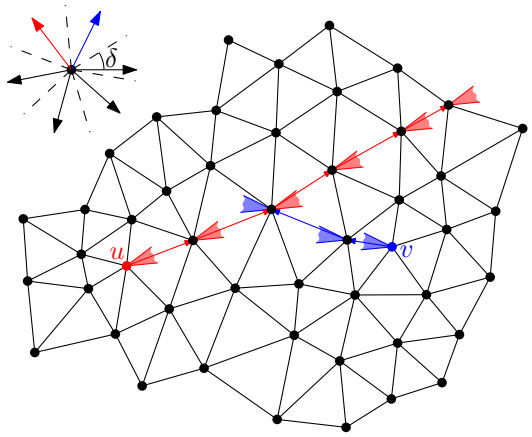


Figure 4: Two vertices u and v that need to be extreme in two directions that differ by at most 2δ (indicated by red and blue) define a unique potential convex chain since there can be at most one edge in each sector. In this figure, the two paths do intersect, but do not form a convex chain.

Theorem 5 *There exists a δ-fat triangulation T of size n such that the number of convex polygons that respect T is $\Omega(n^{\frac{1}{2}\lfloor\frac{2\pi}{\delta}\rfloor})$. This is true even if T is required to be the Delaunay triangulation of its vertices.*

6 Upper bound in fat triangulations

Lemma 6 *Let $u, v \in V(T)$ be two vertices, and let $c, d \in S$ be two directions such that $d - c \leq 2\delta$. Then there is at most one convex path in T from u to v that uses only directions in $[c, d)$.*

Proof. Let $m = c + \frac{1}{2}(d-c)$ be the direction bisecting c and d. Because T is δ-fat, for any vertex in $V(T)$ there is at most one incident edge with outgoing direction in $[c, m)$, and also at most one with direction in $[m, d)$. Because the path needs to be convex, it must first use only edges from $[c, m)$ and then switch to only edges from $[m, d)$. We can follow the unique path of edges with direction in $[c, m)$ from u and the unique path of edges with direction in $[m + \pi, d + \pi)$ from v. If these paths intersect the concatenation may be a unique convex path from u to v as desired (clearly, the path is not guaranteed to be convex). Figure 4 illustrates this. □

Given a convex polygon P that respects T, a vertex v of P is *extreme* in direction $s \in S$ if there are no other vertices of P further in that direction, i.e. if P lies to the left of the line through v with direction $s + \frac{1}{2}\pi$.

Let $\Gamma_\delta = \{0, 2\delta, 4\delta, \ldots, 2\pi\}$ be a set of directions spaced 2δ. As an easy corollary of Lemma 6, the vertices of a convex polygon P respecting T that are extreme in the directions of Γ_δ uniquely define P. There are at most n choices for each extreme vertex, so the number of convex polygons is at most $n^{|\Gamma_\delta|}$. Substituting $|\Gamma_\delta| = \lceil\frac{\pi}{\delta}\rceil$ we obtain the following theorem.

Theorem 7 *Any δ-fat triangulation T of size n has at most $O(n^{\lceil\frac{\pi}{\delta}\rceil})$ convex polygons that respect T.*

7 Discussion

We investigated the maximum number of convex polygons that can be formed using the edges of a given triangulation only. We provided a construction for a general triangulation with $\Omega(1.5028^n)$ such convex polygons, and showed that there is an upper bound of $O(1.62^n)$. The upper and lower bounds for fat triangulations match up to a constant factor (depending on δ) if $\frac{\pi}{\delta}$ is an integer. If not, they differ by a factor $\sqrt{n}$ if the remainder is smaller than $\frac{1}{2}$, or by a factor n if the remainder is at least $\frac{1}{2}$.

Acknowledgements

We thank Stefan Langerman and John Iacono for detecting an error in an earlier version of this paper.

References

[1] B. Aronov, M. de Berg, and S. Thite. The complexity of bisectors and voronoi diagrams on realistic terrains. In *Proc. 16th ESA*, volume 5193 of *LNCS*. Springer, 2008.

[2] B. Aronov, M. van Kreveld, M. Löffler, and R. I. Silveira. Peeling meshed potatoes. *Algorithmica*, 60(2):349–367, 2011.

[3] M. Bern, D. Eppstein, and J. Gilbert. Provably good mesh generation. *J. Comput. Syst. Sci.*, 48(3):384–409, 1994.

[4] M. de Berg, O. Cheong, H. J. Haverkort, J. G. Lim, and L. Toma. The complexity of flow on fat terrains and its i/o-efficient computation. *Comput. Geom.*, 43(4):331–356, 2010.

[5] M. de Berg, A. F. van der Stappen, J. Vleugels, and M. J. Katz. Realistic input models for geometric algorithms. *Algorithmica*, 34(1):81–97, 2002.

[6] J. Matoušek, J. Pach, M. Sharir, S. Sifrony, and E. Welzl. Fat triangles determine linearly many holes. *SIAM J. Comput.*, 23(1):154–169, 1994.

[7] E. Moet, M. van Kreveld, and A. F. van der Stappen. On realistic terrains. *Comput. Geom.*, 41(1-2):48–67, 2008.

[8] J. Pach and M. Sharir. *Combinatorial Geometry and Its Algorithmic Applications: The Alcala Lectures*. Mathematical Surveys and Monographs. AMS, 2009.

On the reconstruction of convex sets from random normals measurements

Hiba Abdallah* Quentin Mérigot†

Abstract

We study the problem of reconstructing convex sets using only a finite number of measurements of normal vectors. More precisely, we suppose that the normal vectors we are measuring come from independent random points uniformly distributed along the boundary of our convex sets. Given a target Hausdorff error ϵ, we are interested in upper bounds on the number of probes that one has to perform in order to obtain an ϵ-approximation of this convex set with high probability. Our results make use of the stability theory related to Minkowski's Theorem.

1 Introduction

Surface reconstruction is now a classical and rather well-understood topic in computational geometry. The input of this question is a finite set of points P measured on (or close to) an underlying unknown surface S, and the goal is to reconstruct a Hausdorff approximation of this surface.

In this article, we deal with a similar reconstruction question. However, our input is not a set of points, but a set of (unit outer) normal vectors measured at various unknown locations on S. This question stemmed from a collaboration with CEA-Leti. The Leti has developed sensors that embed an accelerometer and a magnetometer, and can return their own orientation, but not their position. Since these sensors are small and rather inexpensive it is possible to use many of them to monitor the deformations of a known surface [14]. Can such sensors be used for surface reconstruction ?

In the case where the underlying surface is convex the situation is relatively simple as Minkowski's Theorem asserts that given a set of normal vectors $\mathbf{n}_1, \dots, \mathbf{n}_N$ in the unit sphere, and a set of positive numbers $a_1, \dots, a_N$, there exists a unique (up to translation) convex polytope with exactly N faces and such that the area of the face with normal $\mathbf{n}_i$ is a_i [12]. The computational issues related to Minkowski's Theorem offer several aspects that are of interest in computational geometry [11, 10, 7].

*Laboratoire Jean Kuntzmann, Université Grenoble I, Hiba.Abdallah@ujf-grenoble.fr

†Laboratoire Jean Kuntzmann, Université Grenoble I and CNRS, Quentin.Merigot@imag.fr

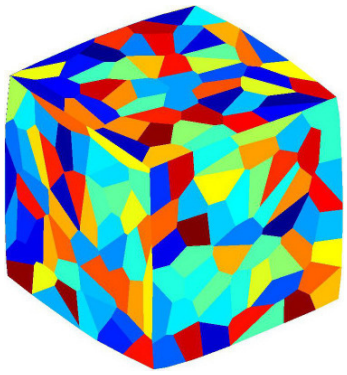


Figure 1: *Reconstruction of a unit cube using Minkowski Theorem from 300 random normal measurements to which a uniform noise of radius 0.05 has been added.*

Stability theory Minkowski's Theorem can be generalized using the notion of surface area measure. Any convex body K in $\mathbb{R}^d$ has a well-defined unit outer normal $\mathbf{n}_K(x)$ at almost every point in the boundary ∂K. The surface area measure of K is the non-negative measure μ_K defined by the relation:

$$\mu_K(B) := \mathcal{H}^{d-1}\left(\{x \in \partial K; \mathbf{n}_K(x) \in B\}\right) \qquad (1)$$

for every (Borel) subset B of $\mathcal{S}^{d-1}$, where $\mathcal{H}^{d-1}$ is the $(d-1)$-Hausdorff measure. Note that by definition, the total mass of the surface area measure $\mu_K(\mathcal{S}^{d-1})$ coincides with the $(d-1)$ area of the boundary ∂K. The following result by Alexandrov [1] generalizes Minkowski's Theorem:

Theorem 1 (Alexandrov) *Let μ be a non-negative measure on $\mathcal{S}^{d-1}$ with zero mean i.e. $\int_{\mathcal{S}^{d-1}} u d\mu(u) = 0$. Then there exists a convex body K, which is unique up to translation, whose surface area measure μ_K coincides with μ.*

Minkowski and Alexandrov Theorems can be interpreted as reconstruction theorems, and with every such theorem comes a corresponding stability question. The results that are of interest in our case are those that assert that if the surface area measures μ_K and μ_L of two convex sets are close (in a sense that will be made precise), then up to a translation, those sets are also Hausdorff-close. Stability results for the Minkowski problem have been obtained using the

This is an extended abstract of a presentation given at EuroCG 2012. 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 in a conference with formal proceedings and/or in a journal.

theory surrounding the Brunn-Minkowski inequality, starting from two articles by Diskant [3, 4, 5, 6, 13, 9].

Contributions In the present paper, we suppose the existence of an underlying convex set K. Our input data is a set made of N unit normals $\mathbf{n}_1, \ldots, \mathbf{n}_N$, measured at N independent random locations $x_1, \ldots, x_N$ uniformly distributed on the boundary of K. The question we consider is the following: given a positive number ϵ, what is the minimum value of N needed to be able to reconstruct a convex set L_N which is ϵ Hausdorff-close to K with high probability?

This problem is closely related to the question of stability in Minkowski's Theorem. In Section 2, we show how the proof techniques of [9] imply a stability result for Minkowski's Theorem for a very weak (pseudo-)distance between measures on the unit sphere:

Definition 1 *Recall that the* support function *of a convex body K is defined by $h_K(u) := \sup_{x\in K}\langle x|u\rangle$. We let Conv_1^{d-1} denote the set of support functions of convex sets contained in the unit ball. Given two measures μ, ν on $\mathcal{S}^{d-1}$, one defines*

$$\mathrm{d_C}(\mu,\nu) = \sup_{f\in \mathrm{Conv}_1^{d-1}} \left| \int_{\mathcal{S}^{d-1}} f \mathrm{d}\mu - \int_{\mathcal{S}^{d-1}} f \mathrm{d}\nu \right|. \quad (2)$$

Theorem (Corollary 5) *Let K, L be two convex sets in $\mathbb{R}^d$ with unit volume and such that K is enclosed between two balls of radii $r < R$ not necessarily centered at the same points, i.e. $\mathrm{B}_r \subseteq K \subseteq \mathrm{B}_R$. Then,*

$$\min_{x\in\mathbb{R}^d} \mathrm{d_H}(K+x, L) \leq \mathrm{const}(r, R, d)\mathrm{d_C}(\mu_K, \mu_L)^{1/d}.$$

The main reason for using the pseudo-distance $\mathrm{d_C}$ is that the space of convex functions on the sphere is in some sense "small". This implies a bound on the $\mathrm{d_C}$-approximation of μ_K given by the random sampling scheme for normals described at the beginning of this paragraph (see Theorem 6).

2 Stability in Minkowski Theorem

In this section, we modify the proof of [9] to obtain a stability theorem with respect to the distance $\mathrm{d_C}$. The techniques we employ are drawn from the theory of mixed volumes and Brunn-Minkowski's theory. In this section, all convex bodies are compact and belong to the d dimensional Euclidean space $\mathbb{R}^d$.

2.1 Background and notations

We let $\mathcal{K}(r, R)$ denote the set of convex bodies that contain a ball of radius r and contained in a another ball of radius R.

We make use of the following representations for the *volume* of K and the *mixed volume* $\mathrm{V}_1(K, L) := \mathrm{V}(K[d-1], L)$ of K and another body L. The reader unfamiliar with mixed volumes can consider these formulas as definitions (cf [13, Chapter 5]):

$$\mathrm{V}(K) = \frac{1}{d}\int_{\mathcal{S}^{d-1}} h_K(u) d\mu_K(u)$$
$$\mathrm{V}_1(K, L) = \frac{1}{d}\int_{\mathcal{S}^{d-1}} h_L(u) d\mu_K(u).$$

Note in particular that $\mathrm{V}_1(K, B)$, where B is the unit ball, is the $(d-1)$-area of ∂K, also denoted by $\mathrm{A}(\partial K)$.

Minkowski's inequality. The following isoperimetric inequality is due to Minkowski:

$$\mathrm{V}_1^d(K, L) \geq \mathrm{V}^{d-1}(K)\mathrm{V}(L). \quad (3)$$

Equality holds if and only if K and L are homothetic. Inequality (3) make it interesting to study the following isoperimetric difference

$$\Delta(K, L) = \mathrm{V}_1^d(K, L) - \mathrm{V}_d^{d-1}(K)\mathrm{V}(L). \quad (4)$$

This isoperimetric difference plays an important role in stability results, especially in the two following theorems of Diskant. We denote by $r(K, L)$ be the inradius of K with respect to L, i.e.

$$r(K, L) = \max\{\lambda > 0; \exists x \in \mathbb{R}^d,\ \lambda L + x \subseteq K\}$$

Theorem 2 *(Diskant [3]) The inradius $r(K, L)$ of K relative to L is lower bounded by:*

$$\left[\frac{\mathrm{V}_1(K, L)}{\mathrm{V}(L)}\right]^{\frac{1}{d-1}} - \frac{[\mathrm{V}_1^{\frac{d}{d-1}}(K, L) - \mathrm{V}(K)\mathrm{V}^{\frac{1}{d-1}}(L)]^{\frac{1}{d}}}{\mathrm{V}^{\frac{1}{d-1}}(L)}.$$

Theorem 3 *(Diskant [3]) There exists two constants C, ϵ_0 depending on r, R and d only such that for any $K, L \in \mathcal{K}(r, R)$ such that if $\epsilon := \Delta(K, L) \leq \epsilon_0$, then*

$$r(K, sL) \geq 1 - C\epsilon^{\frac{1}{d}} \text{ where } s = \left(\frac{\mathrm{V}(K)}{\mathrm{V}(L)}\right)^{\frac{1}{d}}.$$

2.2 Stability in Minkowski's theorem for $\mathrm{d_C}$

Theorem 4 *Let K, L be two convex bodies with unit volume, and suppose that $K \in \mathcal{K}(r, R)$. Then there exists two constants ϵ_0 and c depending on r, R and d only such that if $\epsilon := \mathrm{d_C}(\mu_K, \mu_L) \leq \epsilon_0$, we have*

$$L \in \mathcal{K}((1 - c\epsilon^{\frac{1}{d}})r, (1 + c\epsilon^{\frac{1}{d}})R).$$

Proof. The proof is based on the proof of Theorem 2 in [3], and the Lemma 3 in [4]. We show in two steps that there exists c_1, c_2 depending only on R, d and the surface area of the ball B_R, $A_n(R)$ such that

$$r(L, K) \geq 1 - c_1\epsilon^{\frac{1}{d}} \text{ and } r(K, L) \geq 1 - c_2\epsilon^{\frac{1}{d}} \quad (5)$$

provided that ϵ is smaller than a certain ϵ_0.

(a) Let h_K be the support function of the convex K. We have

$$|\mathrm{V}_1(L,K)| = \left|\frac{1}{d}\int_{\mathcal{S}^{d-1}} h_K(u)d\mu_L(u)\right| \leq \frac{4R}{d}A_n(R)$$

provided that $\epsilon \leq A_n(R)$, and

$$|\mathrm{V}_1(L,K) - \mathrm{V}(K)| \leq \left|\frac{1}{d}\int_{\mathcal{S}^{d-1}} h_K(u)(d\mu_L - d\mu_K)\right| \leq \frac{2R}{d}\epsilon.$$

The second inequality follows from $\mathrm{d_C}(\mu_L,\mu_K) \leq \epsilon$. Since $x^p - y^p \leq px^{p-1}(x-y)$ for $x \geq y \geq 0$ and $p \geq 1$ (see Hardy, Little-wood, Polya [8], p.39), we obtain

$$\begin{aligned}\Delta(L,K) &\leq d\mathrm{V}_1(L,K)|\mathrm{V}_1(L,K) - \mathrm{V}(K)| \\ &\leq \frac{2}{d}(2R)^2 A_n(R)\epsilon.\end{aligned}$$

On the other hand, the isoperimetric inequality (3) gives us $\mathrm{V}_1(L,K) \geq 1$, then using inequality $x^p - y^p \geq px^{p-1}(x-y)$ for $x \geq y \geq 0$ and $p \leq 1$, we have

$$\begin{aligned}\frac{[\mathrm{V}_1^{\frac{d}{d-1}}(L,K) - \mathrm{V}(L)\mathrm{V}^{\frac{1}{d-1}}(K)]^{\frac{1}{d}}}{\mathrm{V}^{\frac{1}{d-1}}(K)} &\leq \frac{[\Delta(L,K)]^{\frac{1}{d}}}{\mathrm{V}^{\frac{1}{d-1}}(K)\mathrm{V}_1^{\frac{d-2}{d-1}}(L,K)} \\ &\leq \left(\frac{2}{d}(2R)^2 A_n(R)\epsilon\right)^{\frac{1}{d}}.\end{aligned}$$

By Theorem 2, we prove the first inequality in (5).

(b) Let $q = 1 - c_1\epsilon^{\frac{1}{d}}$, then

$$\begin{aligned}\Delta(K,L) &= \mathrm{V}_1^d(\frac{q}{q}K, L) - \mathrm{V}^d(L) \\ &= \frac{1}{q^{d(d-1)}}\mathrm{V}_1^d(qK,L) - \mathrm{V}^d(L) \leq \frac{1}{q^{d(d-1)}} - 1\end{aligned}$$

This follows from $qK \subseteq L$, and consequently $\mathrm{V}_1^d(qK,L) \leq \mathrm{V}_1^d(L,L) = \mathrm{V}^d(L) = 1$. Moreover,

$$\frac{1}{q^{d(d-1)}} - 1 \leq c_1(d^2-d-1)\left(\frac{1}{q}\right)^{d(d-1)}\epsilon^{\frac{1}{d}},$$

implies $\Delta(K,L) \leq c_1(d^2-d-1)\left(\frac{1}{q}\right)^{d(d-1)}\epsilon^{\frac{1}{d}}$. Hence,

$$\frac{[\mathrm{V}_1^{\frac{d}{d-1}}(K,L) - \mathrm{V}(L)\mathrm{V}^{\frac{1}{d-1}}(K)]^{\frac{1}{d}}}{\mathrm{V}^{\frac{1}{d-1}}(L)} \leq c_3\epsilon^{\frac{1}{d^2}}$$

where $c_3 = (c_1(d^2-d-1)q^{-d(d-1)})^{\frac{1}{d}}$. By Theorem 2, $r(K,L) \geq 1 - c_3\epsilon^{\frac{1}{d^2}}$ i.e.

$$|h_L(u)| \leq \frac{2R}{(1 - c_3\epsilon^{\frac{1}{d^2}})} \text{ for all } u \in \mathcal{S}^{d-1},$$

provided that $\epsilon \leq \left(\frac{1}{c_3}\right)^{\mathrm{d}^2}$. It follows that

$$|\mathrm{V}_1(K,L) - \mathrm{V}(L)| \leq \frac{1}{d}\frac{2R}{\left(1 - c_3\epsilon^{\frac{1}{d^2}}\right)}\epsilon. \qquad (6)$$

We use the same techniques as in part (a) to prove the second inequality of (5). We finish the proof by setting $c := \min\{c_1, c_2\}$ and $\epsilon_0 = \left(\frac{1}{2c}\right)^{\frac{1}{d}}$. □

Remark. $\epsilon \leq \epsilon_0$ guarantees that $L \in \mathcal{K}(\frac{r}{2}, \frac{R}{2})$.

Corollary 5 *Let K, L be two convex bodies with unit volume and suppose that $K \in \mathcal{K}(r,R)$. There exists two constants c, ϵ_0 depending on r and R only, such that if $\epsilon := \mathrm{d_C}(\mu_K,\mu_L) \leq \epsilon_0$, then*

$$\min_{x\in\mathbb{R}^d} \mathrm{d_H}(K+x, L) \leq c\epsilon^{\frac{1}{d}}.$$

Proof. This is an immediate consequence of inequalities in (5) (see proof of Theorem 7.2.2 in [13]). We also use the fact that all the function used in the integrals are support function of convex sets. □

Remark. The optimal exponent in Corollary 5, while perhaps smaller than $\frac{1}{d}$, cannot be lower than $\frac{1}{d-1}$. This can be seen by choosing for K a right circular cone in $\mathbb{R}^3$, and for L the convex body obtained by cutting off the apex in such a way that the height of the removed part is ϵ.

3 Random sampling

Let K be a convex body in $\mathcal{K}(r,R)$, and μ_K its surface area measure. We consider N vectors $\{\mathbf{n}_i\}_{i=1}^N$ obtained by measuring the outer normal of K at N random and independent points x in ∂K (note that by definition, the distribution of each $\mathbf{n}_i$ is μ_K). The measure $\mu_N := \frac{1}{N}\sum_{i=1}^N \delta_{n_i}$ is called the *empirical measure associated to* μ_K. The main result in this section is the following theorem.

Theorem 6 *Let K be a convex body such that $A(\partial K) = 1$. The empirical measure μ_N constructed from μ_K converges to μ_K as N grows to infinity with high probability:*

$$\mathbb{P}\left[\mathrm{d_C}(\mu_N,\mu_K) \geq \epsilon\right] \leq 2\exp\left(C\epsilon^{\frac{1-d}{2}} - N\epsilon^2/2\right) \qquad (7)$$

where C depends only on the dimension d.

In order to prove Theorem 6, we will need the following result due to Bronshtein. Recall that the ϵ-covering number of a metric space X, denoted by $\mathcal{N}(X,\epsilon)$, is the minimal number of balls of radius ϵ needed to cover X.

Theorem 7 *(Bronshtein [2]) Let $\mathcal{M}_d$ be the set of closed convex subsets contained in the unit ball of $\mathbb{R}^d$, endowed with the Hausdorff distance. The following inequality hold as soon as $\epsilon \leq \epsilon_d := 10^{-12}/(d-1)$*

$$\log_2(\mathcal{N}(\mathcal{M}_d,\epsilon)) \leq 10^6 d^{5/2}\log 12(\sqrt{\epsilon})^{1-d}.$$

Proof of Theorem 6. By Theorem 7 we have

$$\mathcal{N}(\mathcal{M}_d, \epsilon) \leq 2^{C\epsilon^{\frac{1-d}{2}}} \leq \exp\left(C\epsilon^{\frac{1-d}{2}}\right)$$

as soon as $\epsilon \leq \epsilon_d$, where C depends only on the dimension d. We recall that for two convex sets, $\mathrm{d_H}(K, L) = \|h_K - h_L\|_\infty$. This implies the following bound on the covering numbers of Conv_1^{d-1} with respect to the norm of uniform convergence:

$$\mathcal{N}(\mathrm{Conv}_1^{d-1}, \epsilon) \leq \exp\left(C\epsilon^{\frac{1-d}{2}}\right). \tag{8}$$

Hoeffding's inequality asserts that

$$\mathbb{P}\left(\left|\frac{1}{N}\sum_{i=1}^{N} f(n_i) - \int_{\mathcal{S}^{d-1}} f d\mu_K\right| \geq \epsilon\right) \leq 2\exp(-2N\epsilon^2) \tag{9}$$

for all function $f \in \mathrm{Conv}_1^{d-1}$. Inequalities (8), (9) and the union bound imply inequality (7). □

Note that it is possible that the empirical measure μ_N does not satisfy the equality $\int_{\mathcal{S}^{d-1}} u d\mu_N(u) = 0$, which is a necessary condition to the existence of a convex polytope L such that $\mu_L = \mu_N$. The following proposition shows that this equality can be enforced without perturbing μ_N too much in the $\mathrm{d_C}$-sense.

Proposition 8 *If $\mathrm{d_C}(\mu_K, \mu_N) \leq \epsilon$, then there exists another measure $\nu = \sum_{i=1}^{N} a_i \delta_{m_i}$ on the sphere such that $\mathrm{d_C}(\mu_N, \nu) \leq \epsilon$ and*

$$\int_{\mathcal{S}^{d-1}} u \mathrm{d}\mu(u) = \sum_{i=1}^{N} a_i m_i = 0.$$

Proof. Let v be a vector in $\mathcal{S}^{d-1}$ and π be the function defined by $\pi(u) := \langle u|v\rangle = h_{\{v\}}(u)$. Letting $m = \frac{1}{N}\sum_{i=1}^{N} \mathbf{n}_i$, we have

$$\mathrm{d_C}(\mu_K, \mu_N) \geq \left|\int_{\mathcal{S}^{d-1}} \langle u|v\rangle \mathrm{d}(\mu_K - \mathrm{d}\mu_N)\right| = |\langle m|v\rangle|.$$

Since this hold for every v, the norm of m is at most ϵ. It is then easy to see that the measure $\nu = \frac{1}{N}\sum_{i=1}^{N} a_i \delta_{m_i}$, with $a_i = \|\mathbf{n}_i - m\|$ and $m_i = (\mathbf{n}_i - m)/a_i$, satisfies the two requirements. □

Acknowledgements. The authors would like to acknowledge the support of a grant from Université Grenoble I/Joseph Fourier (MSTIC GEONOR), and the associated team ECR "Géométrie et Capteurs" CEA/UJF between LJK-UJF and CEA-LETI.

References

[1] Alexandrov, A. On the theory of mixed volumes of convex bodies. *Mat. Sb 3*, 45 (1938), 227–251.

[2] Bronshtein, E. M. ϵ-entropy of convex sets and functions. *Mat. Sb 17*, 3 (1976), 508–514.

[3] Diskant, V. Bounds for the discrepancy between convex bodies in terms of the isoperimetric difference. *Sib. Math. J. 13*, 4 (1972), 529–532.

[4] Diskant, V. Bounds for the discrepancy between convex bodies in terms of the isoperimetric difference. *Sib. Math. J. 13*, 4 (1972), 529–532.

[5] Gardner, R. *Geometric tomography*. Cambridge University Press, 1995.

[6] Goodey, P., and Groemer, H. Stability results for first order projection bodies. In *Proc. Amer. Math. Soc* (1990), vol. 109.

[7] Gritzmann, P., and Hufnagel, A. On the algorithmic complexity of minkowski's reconstruction theorem. *J. Lond. Math. Soc. 59*, 3 (1999), 1081–1100.

[8] Hardy, G., Littlewood, J., and Pólya, G. *Inequalities*. Cambridge Univ Press, 1988.

[9] Hug, D., and Schneider, R. Stability results involving surface area measures of convex bodies. *Rend. Circ. Mat. Palermo (2) Suppl.*, 70, part II (2002), 21–51.

[10] Lachand-Robert, T., and Oudet, É. Minimizing within convex bodies using a convex hull method. *SIAM J. Optim. 16*, 2 (2005), 368–379.

[11] Little, J. Extended gaussian images, mixed volumes, shape reconstruction. In *Proc. Symposium on Computational Geometry* (1985), ACM, pp. 15–23.

[12] Minkowski, H. Volumen und oberfläche. *Math. Ann. 57*, 4 (1903), 447–495.

[13] Schneider, R. *Convex bodies: the Brunn-Minkowski theory*. Cambridge Univ Prss, 1993.

[14] Sprynski, N., Szafran, N., Lacolle, B., and Biard, L. Surface reconstruction via geodesic interpolation. *Computer-Aided Design 40*, 4 (2008), 480–492.

On Ellipsoidal Approximations of Zonotopes

Michal Černý Miroslav Rada*

Abstract

We adapt Goffin's Algorithm for construction of the Löwner-John ellipsoid for a full-dimensional zonotope given by the generator description.

1 Introduction

The aim of this paper is twofold.

First, we describe the following algorithm. Let $\varepsilon > 0$ be fixed. Given a full-dimensional zonotope $\mathcal{Z} \subseteq \mathbb{R}^n$, represented as a set of rational generators, the algorithm constructs an ellipsoidal approximation of $\mathcal{Z}$ satisfying

$$\mathcal{E}(n^{-2} \cdot E, s) \subseteq \mathcal{Z} \subseteq \mathcal{E}((1+\varepsilon) \cdot E, s) \qquad (1)$$

in time polynomial in the bit-size of the generator description. Here $\mathcal{E}(E, s)$ is the ellipsoid $\{x : (x-s)^{\mathrm{T}} E^{-1}(x-s) \leq 1\}$, where E is positive definite.

The approximation (1) is called *ε-approximate Löwner-John ellipsoid* for $\mathcal{Z}$. (The name comes from the well-known Löwner-John Theorem: for every full-dimensional bounded convex set $A \subseteq \mathbb{R}^n$, there exists an ellipsoid $\mathcal{E}(E, s)$ satisfying $\mathcal{E}(n^{-2} \cdot E, s) \subseteq A \subseteq \mathcal{E}(E, s)$.)

Second, we show how possible improvements of the factor n^{-2} in (1) are related to the following problem:

> *given a zonotope centered at zero, represented as a set of rational generators, and a rational number $\gamma > 0$, does $K_\gamma \subseteq \mathcal{Z}$ hold?* (2)

Here, $K_\gamma = \mathcal{E}(\gamma^2 \cdot I, 0)$ is the euclidian ball centered at zero with radius γ. The problem (2) is likely to be in *co-**NP***, but we do not have a conjecture whether or not it is *co-**NP***-complete.

We also construct several polynomial-time algorithms for testing geometric properties of zonotopes given by rational generators.

Remark. The well-known Goffin's Algorithm [6, 7] constructs the ε-approximate Löwner-John ellipsoid for a general full-dimensional bounded polyhedron $\mathcal{F}$ given by the facet description (A, b). (The *facet description* of $\mathcal{F}$ consists of a matrix A and a vector b such that $\mathcal{F} = \{x : Ax \leq b\}$.) Of course, Goffin's method can be applied to the zonotope $\mathcal{Z}$ as well. The problem is that the conversion of the generator description to the facet description can take superpolynomial time. The reason is that there are zonotopes with a superpolynomial number of facets compared to the number of generators ([3, 11]; see also [12] and [1, 5]). Hence, if we want to preserve polynomial time in the size of input, which is the bit-size of the generator description, we cannot use Goffin's method directly. We will take the advantage of the fact that a zonotope given by the generator description is a kind of an implicitly defined polyhedron in the sense of [7].

Remark. Zonotopes are centrally symmetric polyhedra. By Jordan's Theorem, every centrally symmetric, full-dimensional bounded convex set $A \subseteq \mathbb{R}^n$ can be approximated with a better factor than n^{-2}: we can even achieve $\mathcal{E}(n^{-1} \cdot E, s) \subseteq A \subseteq \mathcal{E}(E, s)$. But the Theorem is non-constructive and does not suggest an algorithmic method. We show that the algorithmic improvement of (1) to the form $\mathcal{E}(n^{-1} \cdot E, s) \subseteq \mathcal{Z} \subseteq \mathcal{E}((1+\varepsilon) \cdot E, s)$ is related to the complexity of the problem (2).

2 Basic definitions and the main theorem

Let $A \subseteq \mathbb{R}^n$ be a set and $x \in \mathbb{R}^n$. We define $A \oplus x := \mathit{convexhull}\{A \cup (A+x)\}$, where $A+x = \{a+x : a \in A\}$. The operation $\oplus$ can be seen as a special case of the Minkowski sum. A *zonotope* $\mathcal{Z} := \mathcal{Z}(s; g_1, \ldots, g_m)$ is the set $\{s\} \oplus g_1 \oplus \cdots \oplus g_m$. The vectors $g_1, \ldots, g_m$ are called *generators* and the vector s is called *shift*. The $(m+1)$-tuple $(s, g_1, \ldots, g_m)$ is called *generator description* of $\mathcal{Z}$. If the vectors $s, g_1, \ldots, g_m$ are rational, we define the *bit-size* of (the generator description of) the zonotope $\mathcal{Z}$ as $\mathit{size}(\mathcal{Z}) := \mathit{size}(s) + \sum_{i=1}^{m} \mathit{size}(g_i)$, where $\mathit{size}(\cdot)$ denotes the bit-size of a rational number/vector/matrix.

Our aim is to prove:

Theorem 1 *For every $\varepsilon > 0$ there is a polynomial-time algorithm that computes the ε-approximate Löwner-John ellipsoid (1) for a given full-dimensional zonotope represented by a rational generator description.*

Remark. Polynomial time means 'weak polynomial time' in the sense that the number of iterations of the algorithm depends on the sizes of rational numbers occurring in the generator description.

*Both authors: Department of Econometrics, University of Economics, Prague; Náměstí Winstona Churchilla 4, CZ13067 Prague 3, Czech Republic, cernym@vse.cz, miroslav.rada@vse.cz.

This is an extended abstract of a presentation given at EuroCG 2012. 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 in a conference with formal proceedings and/or in a journal.

3 Some properties of zonotopes

Let a full-dimensional zonotope $\mathcal{Z} = \mathcal{Z}(s; g_1, \ldots, g_m)$ be given. Let $\partial\mathcal{Z}$ denote the boundary of $\mathcal{Z}$.

The zonotope $\mathcal{Z}$ is a centrally symmetric set; its center is $\mathcal{Z}^c := s + \frac{1}{2}\sum_{i=1}^{m} g_i$. Without loss of generality we can assume that (a) $\mathcal{Z}^c = 0$, and (b) *if g is a generator, then also $-g$ is a generator.* Using (b), the sequence $g_1, \ldots, g_m$ can be ordered into the form $g_1, \ldots, g_{m/2}, -g_1, \ldots, -g_{m/2}$. Then we can state the following lemma.

Lemma 2 *The matrix $G := (g_1, \ldots, g_{m/2})$ satisfies $\mathcal{Z} = \{G\alpha : -1 \le \alpha \le 1, \alpha \in \mathbb{R}^{m/2}\}$.*

Let F be a k-dimensional face of $\mathcal{Z}$ and let A be its affine hull. A set of linearly independent generators $g'_1, \ldots, g'_k$ which form a basis of A is called *basis* of the face F. We write $bas(F) = \{g'_1, \ldots, g'_k\}$. [*Remark.* The basis need not be unique. Whenever we talk about $bas(F)$, we mean that $bas(F)$ is some basis from set of all bases. It will be apparent that it is not important which particular basis is chosen if more bases exist.] Given a point $x \in \mathcal{Z}$, we define the *degree* of x as $deg(x) := \min\{dim(F) : F \text{ is a face of } \mathcal{Z} \text{ containing } x\}$. The face F, for which the minimum is attained, is denoted as $\mathcal{F}(x)$.

Theorem 3 *Let $\mathcal{Z}$ denote a zonotope given by a rational generator description and let x denote a rational vector.*

(a) The relation $x \in \mathcal{Z}$ is polynomial-time decidable.

(b) The relation $x \in \partial\mathcal{Z}$ is polynomial-time decidable.

(c) The number $deg(x)$ is polynomial-time computable.

(d) The set $bas(\mathcal{F}(x))$ is polynomial-time computable.

Proof-sketch. Let G be the matrix from Lemma 2. The symbol α stands for a vector in $\mathbb{R}^{m/2}$.

(a) By Lemma 2 we have $x \in \mathcal{Z}$ iff the linear system $x = G\alpha, -1 \le \alpha \le 1$ is feasible. This is a linear programming problem.

(b) We know that $\mathcal{Z}$ is centered at 0. Now we have that $x \in \partial\mathcal{Z}$ iff $\max\{\delta : \delta x = G\alpha, -1 \le \alpha \le 1\} = 1$, which is a linear programming problem.

(c) Given a generator g, set $\delta_1 = \max\{\delta : x + \delta g = G\alpha,\ -1 \le \alpha \le 1\}$, $\delta_2 = \max\{\delta : x - \delta g = G\alpha,\ -1 \le \alpha \le 1\}$ and $\delta = \min\{\delta_1, \delta_2\}$. If $\delta > 0$ we say that the generator g *can move x in both directions.*

Let H be the set of generators which can move x in both directions. Using linear programming, H can be computed in polynomial time. The dimension of *linearhull*(H) is equal to $deg(x)$.

(d) Let H be as above. Any basis (i.e., maximal linearly independent subset) of H is a basis of $\mathcal{F}(x)$. □

Corollary 4 *Let $x \in \mathcal{Z}$ be a point of degree $\le n-1$. Then, a point x^* with the following property can be found in polynomial time: there is a facet F such that $\{x, x^*\} \subseteq F$ and x^* is in the interior of F.*

Said loosely: if the point x is in a low-dimensional face, then x can be "perturbed" into a position x^* which is the interior of a facet.

4 Sketch of Goffin's method

First, we sketch the important ingredients of the traditional Goffin's method applicable for a bounded full-dimensional polyhedron $\mathcal{P}$ given by the facet description $Ax \le b$. Then, in Section 5, we restate the algorithm for zonotopes. All the propositions stated here can be found in [7, 10].

Goffin's Algorithm is a form of the Ellipsoid Method with shallow cuts. It constructs a finite sequence of ellipsoids $\mathcal{E}(E_0, s_0), \mathcal{E}(E_1, s_1), \ldots$ of shrinking volumes satisfying $\mathcal{P} \subseteq \mathcal{E}(E_i, s_i)$.

The work in one iteration is as follows. Let the ellipsoid $\mathcal{E}(E_i, s_i) \supseteq \mathcal{P}$ be available; we either terminate or construct $\mathcal{E}(E_{i+1}, s_{i+1})$.

By shift we can assume that $s_i = 0$. We apply the transformation $\Phi : \xi \mapsto E_i^{-1/2}\xi$; under this transformation, the ellipsoid $\mathcal{E}(E_i, s_i = 0)$ is mapped to the unit ball $B = \mathcal{E}(I, 0)$ and the polyhedron $\mathcal{P} = \{x : Ax \le b\}$ is mapped to the polyhedron $\mathcal{P}' = \{x : A'x \le b\}$ with $A' = AE_i^{1/2}$. We shrink the unit ball B slightly more than by a factor n, say by a factor $n \cdot \sqrt{1+\varepsilon}$, where $\varepsilon > 0$ is a small number: we set $B' := \mathcal{E}(\frac{1}{n^2(1+\varepsilon)} I, 0)$. We test whether

$$B' \subseteq \mathcal{P}'. \tag{3}$$

If the answer is positive, then we terminate — we have found an approximate Löwner-John ellipsoid.

The test (3) can be performed easily. We know the facet description (A', b); say that $a_1^{\mathrm{T}} x \le b_1, \ldots, a_k^{\mathrm{T}} x \le b_k$ are the inequalities of the system $A'x \le b$. Assume further that they are normalized in the way that $\|a_1\| = \cdots = \|a_k\| = 1$. We test whether the following condition holds:

$$b_j \ge \frac{1}{n \cdot \sqrt{1+\varepsilon}} \quad \text{for all } j = 1, \ldots, k. \tag{4}$$

If (4) holds, then the test (3) is successful. If (4) does not hold, there is an index j_0 such that $b_{j_0} < \frac{1}{n \cdot \sqrt{1+\varepsilon}}$. Then we have found a violated inequality $a_{j_0}^{\mathrm{T}} x \le b_{j_0}$ of $\mathcal{P}'$ which proves that the test (3) fails.

If the test (3) fails, we use the vector a_{j_0} for a cut, called a_{j_0}*-cut*: we construct the smallest-volume ellipsoid $\mathcal{E}'$ containing the set $B \cap \left\{x : a_{j_0}^{\mathrm{T}} x \le \frac{1}{n \cdot \sqrt{1+\varepsilon}}\right\}$ and start a new iteration with $\mathcal{E}'$.

5 The version for zonotopes given by generator descriptions

Now we restate Goffin's method for zonotopes given by generator descriptions. Given a zonotope $\mathcal{Z}$, we use the symbol L for the bit-size of its generator description. Let $vol(\cdot)$ denote volume. For a positive definite matrix E, $vol(E)$ is a shorthand for $vol(\mathcal{E}(E,0))$.

The initial ellipsoid. We need an initial ellipsoid $\mathcal{E}(E_0,0) \supseteq \mathcal{Z}$. Using Lemma 2 we can set $\mathcal{E}(E_0,0) := \mathcal{E}(\frac{m}{2} \cdot GG^{\mathrm{T}},0)$. The expression also shows that the matrix E_0 can be computed in time polynomial in L. And moreover:

Lemma 5 *There exists a polynomial p_1 such that $vol(E_0) \le 2^{p_1(L)}$.*

The lower bound on volume. As the zonotope $\mathcal{Z}$ is full-dimensional, we can choose $j_1,\ldots,j_n$ such that the generators $g_{j_1},\ldots,g_{j_n}$ are linearly independent. Setting $G := (g_{j_1},\ldots,g_{j_n})$ we have $vol(\mathcal{Z}) \ge |\det G| > 0$. As the positive number $|\det G|$ can be computed by a polynomial time algorithm, we have $size(|\det G|) \le p_2(L)$ with some polynomial p_2. Hence we have $vol(\mathcal{Z}) \ge |\det G| \ge 2^{-p_2(L)}$.

Lemma 6 *There exists a polynomial p_2 such that $vol(\mathcal{Z}) \ge 2^{-p_2(L)}$.*

Parallel cuts. We take the advantage of the fact that a zonotope is a centrally symmetric body centered at zero. Central symmetry implies that whenever we know that $\mathcal{Z} \subseteq \{x : c^{\mathrm{T}}x \le \gamma\}$, then also $\mathcal{Z} \subseteq \{x : c^{\mathrm{T}}x \ge -\gamma\}$. It follows that we can use parallel cuts. The following lemma on parallel cuts comes from [2]; see also [7].

Lemma 7 *Let c be a vector satisfying $\|c\| = 1$, let $B = \mathcal{E}(I,0)$ be an n-dimensional unit ball and let $\gamma \in (0, \frac{1}{\sqrt{n}})$. The smallest-volume n-dimensional ellipsoid containing the set $B \cap \{x : -\gamma \le c^{\mathrm{T}}x \le \gamma\}$ is the ellipsoid $\mathcal{E}(E,0)$ with $E = \frac{n(1-\gamma^2)}{n-1}\left(I - \frac{1-n\gamma^2}{1-\gamma^2} \cdot cc^{\mathrm{T}}\right)$. We say that E **results from** B **with a cut** (c,γ).*

Lemma 8 *Let $\varepsilon > 0$. Then there exists a constant $\kappa_\varepsilon \in (0,1)$, depending only on ε, such that the following holds: whenever a vector c satisfying $\|c\| = 1$ is given and the ellipsoid $\mathcal{E}(E,0)$ results from the unit ball B with a cut $(c, \gamma := \frac{1}{\sqrt{n(1+\varepsilon)}})$, then $vol(E) \le \kappa_\varepsilon \cdot vol(B)$.*

Testing whether $\mathcal{Z}$ contains a ball. The next crucial step is the test (3). In Section 4 we could perform the test (3) in the form (4) using the fact that the facet description of the polyhedron under consideration was available. However, now we cannot lean on that description.

At the moment we cannot design a polynomial-time algorithm for testing whether a given zonotope $\mathcal{Z}$, centered at zero, satisfies $K_\gamma \subseteq \mathcal{Z}$, where $K_\gamma = \mathcal{E}(\gamma^2 \cdot I, 0)$ is a ball with radius γ.

Problem. Let T be the problem "given a rational generator description of a full-dimensional zonotope $\mathcal{Z}$ centered at zero and a rational number γ, does $K_\gamma \subseteq \mathcal{Z}$ hold?". We conjecture that the problem T is in *co-**NP*** but we do not have a conjecture whether or not it is *co-**NP***-complete.

Remark. The complement of T is likely to be in ***NP***: the fact $K_\gamma \not\subseteq \mathcal{Z}$ can be witnessed by a point x satisfying $x^{\mathrm{T}}x \le \gamma^2$ and $x \notin \mathcal{Z}$. The former fact is verified in polynomial time easily; the latter fact can be verified in polynomial time by Theorem 3(a). However, a proof of $T \in$ *co-**NP*** would require showing that the witness x of the fact $K_\gamma \not\subseteq \mathcal{Z}$ can always be chosen in the way that it is guaranteed that $size(x) \le p(size(\mathcal{Z}) + size(\gamma))$ holds, where p is a fixed polynomial.

If the problem T is computationally hard, it seems to be a serious obstacle. We overcome it for a certain price: we construct a smaller inscribed ellipsoid. With Theorem 3(a) we can use essentially the same trick as in [7]: instead of testing $K_\gamma \subseteq \mathcal{Z}$ we test whether

$$\gamma e_i \in \mathcal{Z} \quad \text{for all } i = 1,\ldots,n. \tag{5}$$

(Here e_i is the i-th column of the unit matrix.) If the test is successful, by central symmetry we know that all the points $\pm\gamma e_1,\ldots,\pm\gamma e_n$ are in $\mathcal{Z}$; then also

$$\mathcal{Z} \supseteq convexhull\{\pm\gamma e_i : i = 1,\ldots,n\} \supseteq \mathcal{E}(\tfrac{\gamma^2}{n},0). \tag{6}$$

We will perform the test with $\gamma = \frac{1}{\sqrt{n(1+\varepsilon)}}$. Then: if the test (5) is successful, we have $\mathcal{E}(\frac{1}{n^2(1+\varepsilon)}I,0) \subseteq \mathcal{Z}$; if the test (5) is unsuccessful, we know an index j_0 such that $\frac{1}{\sqrt{n(1+\varepsilon)}} \cdot e_{j_0} \notin \mathcal{Z}$.

The separation procedure. If the test (5) is unsuccessful, we know a point $x_0 = \frac{1}{\sqrt{n(1+\varepsilon)}} \cdot e_{j_0}$ satisfying $x_0 \notin \mathcal{Z}$. Then we would like to perform a parallel a-cut of the unit ball $B = \mathcal{E}(I,0)$ with some suitable a. In Section 4 we selected a as the normal vector of the found violated inequality — but this is not possible here because the facet description of $\mathcal{Z}$ is not available.

We will construct a separator a of x_0 from $\mathcal{Z}$, which will be used for the (parallel) a-cut.

Let G be the matrix from Lemma 2. Recall that we assume that the zonotope $\mathcal{Z}$ is centered at zero.

Step 1. We set $\beta^* := \max\{\beta \in \mathbb{R} : \beta x_0 = G\alpha,\ -1 \le \alpha \le 1\}$ (using linear programming). It follows that $x^* := \beta^* x_0 \in \partial\mathcal{Z}$; hence, $deg(x^*) \le n-1$.

Step 2. If $deg(x^*) < n-1$, we replace x^* by a new point x^* with $deg(x^*) = n-1$ using Corollary 4.

Step 3. We know that $deg(x^*) = n-1$. Hence we have $|bas(\mathcal{F}(x^*))| = n-1$. We compute $\{h_1, \ldots, h_{n-1}\} = bas(\mathcal{F}(x^*))$. Observe that $\{x^* + \sum_{i=1}^{n-1} \lambda_i h_i : \lambda_1, \ldots, \lambda_{n-1} \in \mathbb{R}\}$ is a hyperplane separating x_0 from $\mathcal{Z}$.

Step 4. We find a vector orthogonal to $h_1, \ldots, h_{n-1}$: set $H := (h_1, \ldots, h_{n-1})$ and define $a := (I - H(H^{\mathrm{T}}H)^{-1}H^{\mathrm{T}})x_0$. The vector a is the output of the algorithm.

By the theory of Section 3, all tests and operations can be performed in polynomial time.

Remark. Observe that we constructed the separator without using the Yudin-Nemirovskii Theorem (see Sect. 4.3 of [7]).

The algorithm. Let $\varepsilon > 0$ be fixed. Let a rational generator description of a full-dimensional zonotope $\mathcal{Z}$ be given. We have the initial ellipsoid $\mathcal{E}(E_0, 0) \supseteq \mathcal{Z}$ as described above.

Let us describe the work in one iteration. We have $\mathcal{E}(E_i, 0) \supseteq \mathcal{Z}$ from the previous iteration; we either terminate or construct E_{i+1}. We apply the mapping $\Phi : \xi \mapsto E_i^{-1/2}\xi$ under which the ellipsoid $\mathcal{E}(E_i, 0)$ is projected to the unit ball $\mathcal{E}(I, 0)$ and the zonotope $\mathcal{Z}$ generated by $g_1, \ldots, g_m$ is projected to a zonotope $\mathcal{Z}'$ generated by $\Phi(g_1), \ldots, \Phi(g_m)$.

We set $\gamma := \frac{1}{\sqrt{n(1+\varepsilon)}}$ and we perform the test (5) with γ and $\mathcal{Z}'$. If the test passes, we can finish — by (6) we know that $\mathcal{E}(\frac{1}{n^2(1+\varepsilon)}I, 0) \subseteq \mathcal{Z}' \subseteq \mathcal{E}(I, 0)$, and hence $\mathcal{E}(\frac{1}{n^2(1+\varepsilon)}E_i, 0) \subseteq \mathcal{Z} \subseteq \mathcal{E}(E_i, 0)$. It follows that $\mathcal{E}(\frac{1}{1+\varepsilon}E_i, 0)$ is the ε-approximate Löwner-John ellipsoid for $\mathcal{Z}$.

If the test (5) with γ and $\mathcal{Z}'$ fails, we determine the vector a using the separation procedure and perform a cut $(c := \frac{a}{\|a\|}, \gamma)$ using Lemma 7. We get a matrix E from the Lemma. We set $E_{i+1} := \Phi^{-1}(E)$ and the iteration is finished.

Recall that $L = size(\mathcal{Z})$. It is easy to show that the algorithm terminates after no more than $-\frac{1}{\log_2 \kappa_\varepsilon} \cdot [1 + p_1(L) + p_2(L)]$ iterations, where p_1 is the polynomial of Lemma 5, p_2 is the polynomial of Lemma 6 and κ_ε is the constant of Lemma 8. (By Lemma 8 we know that the volumes of the ellipsoids $\mathcal{E}(E_i, 0)$ are decreasing exponentially fast, and if the algorithm does not terminate in N iterations, we get an ellipsoid $\mathcal{E}(E_N, 0) \supseteq \mathcal{Z}$ with volume $< 2^{-p_2(L)}$, which contradicts Lemma 6.)

6 Conclusion

The basic question is whether the statement of Theorem 1 can be improved. In (6) we have lost the factor n^{-1} (or, in terms of lengths of semiaxes, a factor $n^{-1/2}$) not being able to test whether a zonotope contains a ball. If that test could be implemented, then we could strengthen the theorem and find an ellipsoid satisfying $\mathcal{E}(n^{-1} \cdot E, 0) \subseteq \mathcal{Z} \subseteq \mathcal{E}((1+\varepsilon) \cdot E, 0)$. Even if the test couldn't be implemented, it would be challenging to try to adapt the algorithm for finding an approximation $\mathcal{E}(n^{-\lambda} \cdot E, 0) \subseteq \mathcal{Z} \subseteq \mathcal{E}((1+\varepsilon) \cdot E, 0)$ with some $\lambda \in (1, 2)$.

Moreover, the ellipsoids provide a lower bound and an upper bound on volume of a zonotope; this is interesting for the reason that computing the volume exactly is a $\sharp\boldsymbol{P}$-hard problem [4].

Acknowledgments

The work of the first author was supported by Grant No. P403/12/1947 of the Czech Science Foundation. The work of the second author was supported by I.G.A. Project F4/18/2011 of University of Economics, Prague, Czech Republic.

References

[1] D. Avis, K. Fukuda. *Reverse Search for Enumeration.* Discrete Applied Mathematics 65, 1996, 21–46.

[2] R. G. Bland, D. Goldfarb, M. J. Todd. *The Ellipsoid Method: A Survey.* Operations Research 29, 1981, 1039–1091.

[3] R. C. Buck. *Partion of Space.* The American Mathematical Monthly 50, 1943, 541–544.

[4] M. Dyer, P. Gritzmann, A. Hufnagel. *On the Complexity of Computing Mixed Volumes.* SIAM Journal on Computing 27, 1998, 356–400.

[5] J.-A. Ferrez, K. Fukuda, T. Liebling. *Solving the Fixed Rank Convex Quadratic Maximization in Binary Variables by a Parallel Zonotope Construction Algorithm.* European Journal of Operational Research 166, 2005, 35–50.

[6] J.-L. Goffin. *Variable Metric Relaxation Methods. Part II: The Ellipsoid Method.* Mathematical Programming 30, 1984, 147–162.

[7] M. Grötschel, L. Lovász, A. Schrijver. *Geometric Algorithms and Combinatorial Optimization.* Springer, Heidelberg, 1993.

[8] L. J. Guibas, A. Nguyen, L. Zhang. *Zonotopes as Bounding Volumes.* Proceeding SODA '03 Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms. SIAM Pennsylvania, 2003.

[9] S. Schön, H. Kutterer. *Using Zonotopes for Overestimation-Free Interval Least-Squares — Some Geodetic Applications.* Reliable Computing 11, 2005, 137–155.

[10] A. Schrijver. *Theory of Linear and Integer Programming.* Wiley, New York, 2000.

[11] T. Zaslavsky. *Facing up to Arrangements: Face-count Formulas for Partitions of Space by Hyperplanes.* Memoirs of the American Mathematical Society 154, 1975.

[12] G. Ziegler. *Lectures on Polytopes.* Springer, Heidelberg, 2004.

A Uniform Approach to Enumeration of Facets, Enumeration of Vertices and Computation of Volume of a Zonotope

Miroslav Rada Michal Černý*

Abstract

A uniform approach is proposed for solving three fundamental problems related to zonotopes given by generator descriptions—enumeration of facets, enumeration of vertices and computation of volume. A common algorithmic frame is designed, allowing us to solve all three problems simultaneously. The proposed algorithm is compact for enumeration of facets and computation of volume.

1 Introduction

1.1 Zonotopes

We define the *Minkowski sum* of a set $A \subseteq \mathbb{R}^d$ and a vector $b \in \mathbb{R}^d$ as $A \dot{+} b := \{a + \alpha b : a \in A, 0 \leq \alpha \leq 1\}$. A *zonotope* is a polytope defined as the Minkowski sum of a finite number of line segments. Formally: given $s, g_1, \ldots, g_m \in \mathbb{R}^d$, a *zonotope* is the set

$$\mathcal{Z}(s; g_1, \ldots, g_m) := \{s\} \dot{+} g_1 \dot{+} \cdots \dot{+} g_m.$$

The $(m+1)$-tuple $(s; g_1, \ldots, g_m)$ is called *generator description (of the zonotope)*, the vector s is called *shift* and the vectors $g_1, \ldots, g_m$ are called *generators*.

The *dimension* of the zonotope $\mathcal{Z}(s; g_1, \ldots, g_m)$, denoted $dim(\mathcal{Z})$, is the affine dimension of the set $\mathcal{Z}(s; g_1, \ldots, g_m)$.

Zonotopes have some structural properties, summarized in Lemma 1, which will be useful later.

Lemma 1 *(a) For every zonotope $\mathcal{Z} = \mathcal{Z}(s; g_1, \ldots, g_m) \subset \mathbb{R}^d$ there exist a matrix $G \in \mathbb{R}^{m \times d}$ and vectors $y^d, y^h \in \mathbb{R}^m$ such that $\mathcal{Z} = \{x \in \mathbb{R}^d : x = Gy, y^d \leq y \leq y^h, y \in \mathbb{R}^m\}$.*

(b) A zonotope $\mathcal{Z}(s; g_1, \ldots, g_m)$ is a centrally symmetric set with center $s + \frac{1}{2}\sum_{i=1}^{m} g_i$;

(c) Given $\mathcal{Z} = \mathcal{Z}(s; g_1, \ldots, g_m)$, let $g_i = \alpha g_j$ for distinct $i, j \in \{1, \ldots, m\}$ and $\alpha \in \mathbb{R}$. Then

$$\mathcal{Z} = \begin{cases} \mathcal{Z}(s; g_1, \ldots, g_{i-1}, g_i + g_j, \ldots, g_{j-1}, g_{j+1}, \\ \quad \ldots, g_m) \qquad \text{for } \alpha \geq 0 \\ \mathcal{Z}(s + g_j; g_1, \ldots, g_{i-1}, g_i - g_j, \ldots, g_{j-1}, \\ \quad g_{j+1}, \ldots, g_m) \qquad \text{for } \alpha < 0. \end{cases}$$

The generator g_j from Lemma 1(c) is called *redundant*. The process of removal of redundant generators can be iterated until all are removed. Then we obtain a new representation $(s', g'_1, \ldots, g'_{m'})$ with $m' < m$ of the same zonotope. It will be denoted

$$red(s; g_1, \ldots, g_m) := (s', g'_1, \ldots, g'_{m'}). \qquad (1)$$

1.2 The algorithm and some of its properties

In this paper we deal with three combinatorial problems related to zonotopes given by generator descriptions—enumeration of facets, enumeration of vertices and computation of volume.

There is an important result yielding a lower bound on complexity of any algorithm enumerating facets and vertices:

Theorem 2 ([7]) *Given a zonotope in $\mathbb{R}^d$ with m generators, it holds $f_0 \leq 2\sum_{i=0}^{d-1} \binom{m-1}{i}$ and $f_{d-1} \leq 2\binom{m}{d-1}$, where f_0 and f_{d-1} stand for the number of vertices and the number of facets, respectively. Furthermore, there exist zonotopes such that the inequalities are satisfied with equality.*

It follows that it is not possible to design an algorithm which would compute the vertex or facet representation of a zonotope given by generator representation in time polynomial in the number of generators and dimension.

We propose algorithms computing volume, the facet representation and the vertex representation of a given zonotope. Facet representation is computed with the same effort as the volume computation in terms of time and space complexity; both these algorithms are compact. The time complexity of vertex representation is the same as of facet representation, but it is not compact.

Remark. Let A be an algorithm and let $|X|$ denote the size of an instance X for A. The algorithm A is said to be *compact*, if there is a polynomial $p(|X|)$ such that for each instance X the space consumed in the execution of $A(X)$ is $\leq p(|X|)$, where the size of the output is disregarded. (That is, the Turing machine for A consumes at most $p(|X|)$ cells on the input tape and working tapes, the space on the write-only output tape not being taken into account.)

Our approach is useful for problems where the all three characteristics of a zonotope are required, see [2] for an example.

*Both authors: Department of Econometrics, University of Economics, Prague, nám. W. Churchilla, Praha 3, 130 67 Czech Republic, miroslav.rada@vse.cz, cernym@vse.cz

This is an extended abstract of a presentation given at EuroCG 2012. 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 in a conference with formal proceedings and/or in a journal.

1.3 Remarks on known algorithms for the problems

For the vertex enumeration problem, there is a compact algorithm polynomial in output size, based on linear programming and reverse search in [1]. There is also the worst-case optimal algorithm for this problem as shown in [4]. The optimality is based on the fact that the computation time is asymptotically the same as the size of output.

For the facet enumeration problem, an algorithm polynomial in output size is known, see [6].

There is also an algorithm (see [5]) for enumeration of all faces of a zonotope given by the list of vertices, which is compact and polynomial in output size; hence one can use it together with [1] for the enumeration of all faces of a zonotope given by generators.

By [3], computation of volume of zonotope is a $\#\boldsymbol{P}$-hard problem. Given a zonotope $\mathcal{Z} := \mathcal{Z}(s; g_1, \ldots, g_m)$ in $\mathbb{R}^d$, a simple approach to compute its volume is to decompose it into $\binom{m}{d}$ "small" zonotopes, each of which is given by d generators. The volume of $\mathcal{Z}$ is then given by the formula (see [7])

$$\mathrm{vol}(\mathcal{Z}) = \sum_{1 \le i_1 < \cdots < i_d \le n} \left|\det\left(g_{i_1}, \ldots, g_{i_d}\right)\right|. \qquad (2)$$

2 Reduction-and-Reconstruction Recursive algorithm (*RRR*)

RRR is a common algorithmic frame which can be used for solving (at least) three basic problems: volume computation, facet enumeration and vertex enumeration. The algorithmic frame uses unspecified procedures *BasicCase* and *Combine*. (These procedures depend on the particular problem to be solved and will be specified later.)

Input of the algorithm. As the input we get a d-dimensional zonotope Z given by the generator description $Z := \mathcal{Z}(s; g_1, \ldots, g_m)$. Note that for every permutation π of the set $\{1, \ldots, n\}$ it holds $\mathcal{Z}(s; g_1, \ldots, g_m) = \mathcal{Z}(s; g_{\pi(1)}, \ldots, g_{\pi(m)})$. Hence we can assume that generators $g_1, \ldots, g_d$ are linearly independent.

The algorithmic frame. The symbol $\mathcal{P}(\gamma_1, \gamma_2, H)$ denotes the *projection function*. It computes the image of generator γ_1 projected parallelly with the generator γ_2 into the hyperplane H.

{1} **Function** $RRR(\tilde{s}; \tilde{g}_1, \ldots, \tilde{g}_{\tilde{m}})$
{2} $(s; g_1, \ldots, g_m) := red(\tilde{s}; \tilde{g}_1, \ldots, \tilde{g}_{\tilde{m}})$
{3} **if** $m = dim(g_1, \ldots, g_m)$ **then**
{4} $Output := BasicCase(s; g_1, \ldots, g_m)$
{5} **else**
{6} **choose** $j(g_m)$ **such that** $g_m \notin H_{j(g_m)}$
{7} **define** $\mathcal{P}_m(x) := \mathcal{P}(x, g_m, H_{j(g_m)})$
{8} $D_1 := RRR(s; g_1, \ldots, g_{m-1})$
{9} $D_2 := RRR(\mathcal{P}_m(s); \mathcal{P}_m(g_1), \ldots, \mathcal{P}_m(g_{m-1}))$
{10} $Output := Combine(s; g_1, \ldots, g_m; D_1, D_2)$
{11} **end.**

To explain the symbols in {6} and {9}: $j(g)$ is an index from the set $\{1, \ldots, m\}$ (its choice, depending on the g, was specified in {6}) and $H_{j(g_m)}$ is the hyperplane $H_{j(g_m)} := \{\xi \in \mathbb{R}^d : \xi_{j(g_m)} = 0\}$.

The step {7} introduces a shorthand simplifying notation only.

Idea of the algorithm. We process the sequence of zonotopes $Z_d, \ldots, Z_m$, where $Z_i = \mathcal{Z}(s; g_1, \ldots, g_i)$ for $i \in \{d, \ldots, m\}$. Clearly, $Z_m = Z$.

Z_d is determined by exactly d generators (and shift). By assumption, these generators are linearly independent. Hence Z_d is a combinatorially simple parallelotope, for which the computation of volume, facets and vertices is simple. Z_d is processed by the procedure *BasicCase*.

Construction of Z_i for $d < i \le m$ is based on Z_{i-1}, computed previously, and its image in the hyperplane $H_{j(g_i)}$ under the projection $\mathcal{P}$. If Z_{i-1} is known, we examine how it changes when we add the generator g_i. This is formalized by the procedure *Combine*, particular forms of which will be described later.

Definition 1 *Given Z_i, the symbol $\mathcal{P}(Z_{i-1})$ denotes the input of recursively called RRR on row {9}, ie. $\mathcal{P}(Z_{i-1}) := \mathcal{Z}(\mathcal{P}_i(s); \mathcal{P}_i(g_1), \ldots, \mathcal{P}_i(g_{i-1}))$.*

The generator g_i is the ***used generator*** *for $\mathcal{P}(Z_{i-1})$.*

Definition 2 *Given zonotope $\mathcal{Z}$ and its generator g, the* ***upper boundary*** *of $\mathcal{Z}$ for the generator g is the set $UB(\mathcal{Z}, g) := \{x \in \mathcal{Z} : \max_{\alpha \in \mathbb{R}}\{\alpha : x + \alpha g \in \mathcal{Z}\} = 0\}$. The* ***lower boundary*** *of $\mathcal{Z}$ for the generator g is the set $LB(\mathcal{Z}, g) := UB(\mathcal{Z}, -g)$.*

2.1 Application 1: Volume computation

For computation of volume we define the procedures *BasicCase* and *Combine* as follows:

$$BasicCase(s; g_1, \ldots, g_m) = \left|\det(g_1, \ldots, g_m)\right|,$$
$$Combine(s; g_1, \ldots, g_m, \alpha_1, \alpha_2) = \alpha_1 + \alpha_2 \left|g_{mj(g_m)}\right|. \qquad (3)$$

The symbol $g_{mj(g_m)}$ in (3) stands for the $j(g_m)$-th component of the vector g_m, where $j(g_m)$ is the index chosen in {6}.

Algebraic interpretation and complexity. When we compute the volume of Z_m using (2), we usually multiply the entries of the main diagonal of each of the determinants after its transformation to the lower triangular form. The total number of determinants is $\binom{m}{d}$. Some of them may be null.

When we compute the volume of Z_m using *RRR*, the volume of Z_{i-1} is computed in {8}. In terms of (2) it means that we compute determinants for all

$\binom{m-1}{d}$ d-tuples of generators (α_1 in (3)). It remains to compute the $\binom{m-1}{d-1}$ determinants of all such d-tuples, where one of the components is g_m. Let α_3 be the sum of those determinants. The computation is done in {9} and partly in {10}. The projection function $\mathcal{P}_m(\cdot)$ can be seen as one iteration of a transformation of *all* determinants containing g_m to the lower triangular form. It nullifies the $j(g_m)$-th entry in its first argument. The projected generators form a $(d-1)$-dimensional zonotope with volume equal to the sum of $\binom{m-1}{d-1}$ determinants (α_2 in (3)). When we multiply α_2 by $j(g_m)$-th entry of g_m, we get α_3.

RRR has two advantages for the volume computation. First, assume that procedure *red* is not used. Then some work in evaluation of the determinants is saved:

Theorem 3 *The time complexity of volume computation of $\mathcal{Z}(s; g_1, \ldots, g_m)$ using RRR is $\mathcal{O}(md\binom{m}{d})$.*

This is an improvement by factor d^2/m compared to (2), assuming complexity of determinant computation is $\mathcal{O}(d^3)$.

Second, null determinants are implicitly excluded. Furthermore, some determinants are evaluated together after joining generators in {2}.

Exact quantification of how much computation is saved depends on the initial ordering of generators. At the moment we can not state any particular result on how the initial ordering of generators influences the computational effort saved. This is an interesting question for further research.

2.2 Application 2: Facet enumeration

Let $Facet(s; g_1, \ldots, g_m; k_1, \ldots, k_{d-1})$ be a procedure which, for a given zonotope $\mathcal{Z}(s; g_1, \ldots, g_m)$ in $\mathbb{R}^d$, computes two centrally symmetric hyperplanes containing two facets of $\mathcal{Z}$ generated by $g_{k_1}, \ldots, g_{k_{d-1}}$. Then it is sufficient that *RRR* enumerates $(d-1)$-element index sets generating facets.

Formally, an *index set* is a set $\{k_1, \ldots, k_{d-1}\}$ such that the generators $g_{k_1}, \ldots, g_{k_{d-1}}$ generate a facet. (Observe that they must be linearly independent.)

Furthermore, no pair of index sets spanning the same hyperplane (and hence producing the same facet) should be output. We call two index sets *convertible*, if they span the same hyperplane.

The idea. Let Z_m be the input of *RRR* at the first level of recursion. When we are evaluating *RRR* for, say, Z_i and recursively call *RRR* for $\mathcal{P}(Z_{i-1})$ in {9}, we need to keep information which generator of $\mathcal{P}(Z_{i-1})$ corresponds to which one in Z_i (and hence also in Z_m). [We say that a generator g' of the zonotope $\mathcal{P}(Z_{i-1})$ *corresponds* to the generator g of the zonotope Z_i if g' is the image of g under the projection $\mathcal{P}_i$.] The correspondence must be maintained because we are using the procedure *red* in {2}.

All the procedures *RRR*, *Combine*, *BasicCase* and also *red* will then have a list of indices $\ell_1, \ldots, \ell_i$ as parameter. Each index in this list determines the corresponding generator of original zonotope Z_m. If the procedure *red* reduces a generator, i.e. joins two generators into one generator, we will preserve the lower index of the two generators joined.

The procedure *BasicCase* is evaluated in the following way. Parallelotopes have no convertible index sets, so we simply return all possible ones: $BasicCase(s; g_1, \ldots, g_m; \ell_1, \ldots, \ell_m) := \{\{\ell_1, \ldots, \ell_m\} \setminus \{\ell_k\} : 1 \le k \le m\}$.

The procedure *Combine* uses the fact that every index set determines a facet. For the zonotope Z_i in $\mathbb{R}^d$, the facet representation of Z_{i-1} is the subset of a facet representation of Z_i, since all index sets of generators of Z_{i-1} are index sets of Z_i, too.

The addition of the generator g_i to Z_{i-1} creates some new index sets. The recursive call in {9} in *RRR* computes the facet representation of the $\mathcal{P}(Z_{i-1})$, which is the set of $(d-2)$-element index sets of Z_{i-1}. When we add ℓ_i (as an index of g_i) to all of these index sets, we get the set of all index sets of Z_i containing ℓ_i. Hence *Combine* is simply the union of all computed index sets: $Combine(s; g_1, \ldots, g_m, IS_1, IS_2; \ell_1, \ldots, \ell_m) := IS_1 \cup \{K \cup \{\ell_i\} : K \in IS_2\}$. At the end, we remove convertible index sets before the facets are output.

2.3 Improvement of 2.2: Ensuring compactness

Definition 3 *Let $\mathcal{Z}$ be a zonotope in $\mathbb{R}^d$ and K_1, K_2 be convertible index sets of $\mathcal{Z}$. We say that index set K_1 **is less than** K_2, if $\max(K_1 \setminus K_2) < \max(K_2 \setminus K_1)$.*

*Let K be an index set of $\mathcal{Z}$. Then K is **minimal**, if it is less than all index sets convertible with K.*

We show how to modify the algorithm of Sect. 2.2 to make it compact. During the recursive calls of *RRR*, indices of all used generators (in sense of Definition 1) will be remembered and hence they can be utilized in the procedure *BasicCase* when it is evaluated. The basic observation is that the procedure *Combine* is not longer necessary for joining used generators and $(d-2)$-element index sets.

Furthermore, if we ensure that no two convertible $(d-1)$-element sets will be created by any *BasicCase* call, it will be possible to output the index sets (and hence the facets) in the moment the *BasicCase* is evaluated. In that case *we do not need Combine anymore* and the algorithm becomes compact.

Recall that the *red* procedure preserves the lower indices when reducing generators. Hence, all the minimal convertible index sets will be processed by some *BasicCase* call. So, when evaluating *BasicCase* procedure, we test for each possible index set whether is it minimum. If so, we output it, otherwise we discard it. This observation concludes the description.

2.4 Application 3: Vertex enumeration

The *BasicCase* procedure is simple: vertices of a parallelotope can be enumerated as the set of all 2^d combinations of generators of parallelotope.

Suppose we know the vertex representation $\mathcal{V}(Z_{i-1})$ of the zonotope Z_{i-1}. The set $\mathcal{V}(\mathcal{Z}_{i-1})$ can be decomposed into three subsets according to which boundary they belong to:

$$E := \{v \in \mathcal{V}(\mathcal{Z}_{i-1}) \colon v \in UB(Z_{i-1}, g_i) \cap LB(Z_{i-1}, g_i)\},$$
$$U := \{v \in \mathcal{V}(\mathcal{Z}_{i-1}) \colon v \in UB(Z_{i-1}, g_i) \setminus LB(Z_{i-1}, g_i)\},$$
$$L := \{v \in \mathcal{V}(\mathcal{Z}_{i-1}) \colon v \in LB(Z_{i-1}, g_i) \setminus UB(Z_{i-1}, g_i)\}.$$

Observe that if a vertex v is on the lower boundary of the zonotope Z_{i-1}, then v is the vertex of the zonotope Z_i, too. If a vertex v is on the upper boundary, then $v + g_i$ is the vertex of the zonotope Z_i.

It remains how to test boundary membership efficiently. Denote the vertex representation of $\mathcal{P}(Z_{i-1})$ produced in {9} as $\mathcal{V}(\mathcal{P}(Z_{i-1}))$. Observe that each vertex in E is an inverse image of some vertex in $\mathcal{V}(\mathcal{P}(Z_{i-1}))$. Then, after computation of $\mathcal{V}(\mathcal{P}(Z_{i-1}))$ and $\mathcal{V}(Z_{i-1})$, we know which vertices of $V(Z_{i-1})$ are in E. The remaining vertices are to be decomposed into U and L.

Suppose we know a supporting halfspace for each vertex in Z_{i-1}. For each vertex v in $Z_{i-1} \setminus E$ and its supporting halfspace H, we claim that if for $v + g_i$ the inequality H is valid, then v is in L, otherwise it is in U. Hence, *Combine* can be defined as follows:

$$Combine(s; g_1, \ldots, g_m, VS_1, VS_2) := L' \cup U' \cup E',$$
$$L' := \{v : v \in L\}, \quad U' := \{v + g_m : v \in U\},$$
$$E' := \{v, v + g_m : v \in E\}.$$

The last question is how to get the supporting halfspaces for the vertices of Z_i. The supporting halfspace of each $v \in L' \cup U'$ has the same normal vector as the original vertex. The supporting halfspace of each $v \in E'$ is determined using the supporting halfspace of $\mathcal{P}_i(v)$, which may be assumed to be known by recursion, and the generator g_i.

3 Summary of algorithmic properties and conclusions

It is interesting that the number of calls of the main *RRR* procedure is independent on whether we compute volume, facets or vertices. Hence the problems can be solved simultaneously.

The number of calls is at most $\mathcal{O}(m^{d-1})$ in the worst case (each projection decreases the dimension by one and we can terminate when $d = 1$). It follows that when the dimension is fixed, the algorithm becomes polynomial as long as the *BasicCase* and *Combine* are polynomial procedures.

The number of iterations cannot be improved in general, because the number of facets can attain the same order value by Theorem 2. Computation of volume also cannot be expected to be improved to polynomial time as the problem is $\#\boldsymbol{P}$-hard. (In this sense, *RRR* is a universal method for $\#\boldsymbol{P}$.)

The algorithm is compact for facet enumeration and volume computation. For vertex enumeration, there is a drawback that the procedure *Combine* needs to remember each vertex produced so far. Finding a compact version of the algorithm for vertex enumeration is an interesting problem for further research.

For volume computation, *RRR* is better than (2) by the factor d^2/m (see Theorem 3), which is interesting for example when $d \approx m/2$.

It is also worth mentioning that degeneration of zonotopes is treated by the algorithm implicitly.

It would be also interesting to find further applications of the algorithmic frame in geometry of polytopes.

Acknowledgments

The work of the first author was supported by I.G.A. Project No. F4/18/2011 of University of Economics, Prague. The work of the second author was supported by Grant No. P403/12/1947 of the Czech Science Foundation.

References

[1] D. Avis and K. Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65:21–46, 1993.

[2] M. Černý, J. Antoch, and M. Hladík. On the possibilistic approach to linear regression models involving uncertain, indeterminate or interval data. Technical report 020811, University of Economics, Prague, Prague, 2011. `http://nb.vse.cz/~cernym/plr.pdf`

[3] M. Dyer, P. Gritzmann, and A. Hufnagel. On the complexity of computing mixed volumes. *SIAM J. Comput.*, 27:356–400, April 1998.

[4] H. Edelsbrunner, J. O'Rourke, and R. Seidel. Constructing arrangements of lines and hyperplanes with applications. *SIAM J. Comput.*, 15:341–363, 1986.

[5] K. Fukuda, S. Saito, and A. Tamura. Combinatorial face enumeration in arrangements and oriented matroids. *Discrete Applied Mathematics*, 31(2):141–149, 1991.

[6] P. D. Seymour. A note on hyperplane generation. *Journal of Combinatorial Theory, Series A*, 61(1):88–91, 1994.

[7] G. Ziegler. *Lectures on polytopes*. Springer, 2004.

Domain Mapping using Harmonic Functions in non-convex domains of genus non-zero

Richard Klein* Hari K. Voruganti† Michael Sears†

Abstract

This paper outlines a method to perform domain mapping using harmonic functions, following which an extension is presented that allows it to handle 2D domains containing voids. Many geometric processing tasks can be performed efficiently on regular domains. As complex, non-convex, real–world geometries come into play, however, this efficiency is often lost and sometimes there may be no method to find a solution. Domain Mapping aims to construct some map from a complicated space to a simple, usually convex, space. Once this map is constructed, solutions can be found in the new target domain and transferred back to the original domain. Some problems that benefit from domain mapping include mesh construction, robot path planning and computer animation.

1 Introduction

Many of the problems in mathematics have efficient and well understood solutions in regular domains. As complex, real–world geometries come into play, however, this elegance is often lost. This is particularly the case with numerical meshes of physical problems. Domain mapping helps to move problems from a geometrically complex *source domain* to a regular, easy to use *target domain*.

Numerical methods usually work over some mesh on the relevant domain. The structure and detail of these meshes can affect the overall computational effort and accuracy significantly. Unfortunately, building a good mesh is not always a straight forward task. Finite element analysis in 3D, for example, typically requires 4 to 10 times the number of tetrahedral elements to achieve the same accuracy as the corresponding hexahedral mesh [3]. Constructing such a hexahedral mesh, however, is a difficult task; so in practice many people use tetrahedral meshes instead.

Once the geometrically complex domain has been mapped to a regular domain, one can construct elegant meshes that bear useful properties. After the mesh has been constructed, calculations can be performed in the new domain and transferred back to the original domain later. Alternatively, the mesh can be transferred back to the original domain where calculations can be performed.

Another application of domain mapping is found in robot path planning, where the path between two configurations of a robot is found by connecting two points in the robot's configuration space. In many cases this configuration space is non-convex and in spaces with high dimensionality, finding a path between two arbitrary points becomes a non-trivial task [5].

A final example of domain mapping's use comes from the field of computer animation [2]. In order to morph one object smoothly to another, one needs to calculate some map between them. By using domain mapping to map two topologically equivalent objects to the same target domain, one implicitly creates a diffeomorphism from one object to the other.

Voruganti *et al.* [4, 5] outline a method that allows one to parametrise non-convex source domains of genus-0 by spherical co-ordinates through the use of Harmonic Functions. This paper outlines various ways to extend their method to handle the 2D case where the source domain is not of genus-0. The constraint that the source domain is simply connected is relaxed so that one only requires it to be connected. This allows the space to contain voids. This paper implements a method which is applicable to remeshing, robot path planning and video animation.

The original method for genus-0 domains is described in Section 2 along with some examples. Section 3 then explains the extensions to the method that allow it to handle voids in 2D. Following that, Section 4 shows examples of the method with discussions of the various cases. Finally, Section 5 illustrates domain mapping's path planning and remeshing abilities.

2 Domain Mapping

2.1 Harmonic Functions for Domain Mapping

Domain mapping aims to find some parametrisation of the source domain to some regular target domain. The method outlined in [4, 5] parametrises the source domain into the relevant spherical co-ordinates. Thus in 2D f is a map to polar co-ordinates and in 3D f is a map to spherical co-ordinates. Source domains of higher dimensionalities are mapped to the appropriate

*School of Computer Science, University of the Witwatersrand, South Africa, richard.klein@wits.ac.za

†School of Computer Science, University of the Witwatersrand, South Africa

dimensional sphere, i.e. $f : (\vec{x}) \rightarrow (r, \vec{\theta})$.

While the method is an adaptation of the artificial potential field approach, it is novel in the way it uses the potential field to generate the required parametrisation. Firstly, some arbitrary centre point within the domain, S, is chosen. This is set with some low potential value, say 0. The boundary of the domain is then set with some high potential value, say 1. Laplace's equation ($\bigtriangledown^2\phi = 0$) is then solved over the domain to generate a potential field. Once the potential field has been calculated, any point, $a \in S$, can then be parametrised by tracking a streamline from that point towards the centre. Once this is done, the point a is parametrised into spherical co-ordinates; r is set to be the potential value at the point a and $\vec{\theta}$ is the angle at which its streamline approaches the centre. Rather than tracking streamlines for every point, which is computationally expensive, one need only track streamlines from an adequate selection of boundary points. Every point along a single streamline now has the same $\vec{\theta}$ value and some unique r value (potential) between the boundary values $[0; 1]$. To map any point that does not lie on the calculated streamlines, one can interpolate values from the closest two streamlines. This decreases the computational effort, albeit at the cost of some numerical accuracy.

The method outlined requires that all streamlines finish at the centre and that the computational cost of generating the potential field be relatively low. Laplace's equation is used so that ϕ will be a harmonic function. A function, ϕ, is Harmonic on some domain iff it solves Laplace's equation over that domain. Harmonic functions have two useful properties of which the method takes advantage.

Firstly, Harmonic functions satisfy the *Mean Value Theorem* and conversely, if a function satisfies the mean value property it is Harmonic. If $B(\vec{x}, r)$ is a closed ball with centre $\vec{x}$, radius r, and $B(\vec{x}, r) \subset S$, then the value of the harmonic function at the centre of the ball, $\phi(\vec{x})$, is the average value of ϕ over the surface of the ball [1, 4, 5].

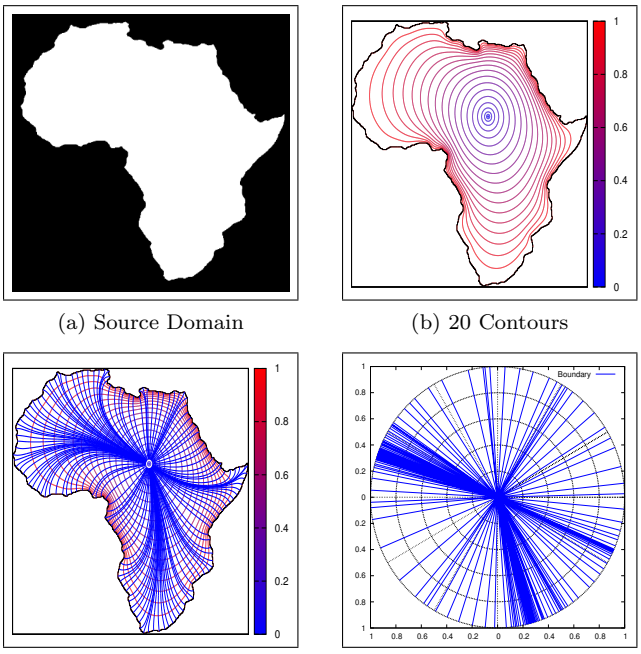
A useful numerical result follows from this property. If S is discretised with some uniform mesh, then the potential value of any point is the average of the neighbouring potential values. These are the finite difference equations derived for Laplace's equation.

This indicates that the simple finite difference method lends itself to the calculation of the potential field. Another benefit is that the finite difference matrices are sparse, which can be leveraged during computation to minimize memory usage.

The method requires that all of the streamlines reach the centre. For this reason, it fails if there are local extrema in the potential field. The maximum principle guarantees that there are none: unless a harmonic function, ϕ, is constant, it only achieves extrema on the domain boundaries [4]. This also ensures that all of the potential values within the domain fall in $[0; 1]$; resulting in a spherical parametrisation with a maximum radius of 1.

2.2 Example

Figure 1 illustrates how the method maps the source domain – namely the outline of Africa – to its corresponding target domain. Figure 1a shows the original domain, Figures 1b and 1c show the contours of the potential field and the boundary streamlines respectively and Figure 1d shows each streamline once it has been mapped to the target domain.



(a) Source Domain (b) 20 Contours

(c) Boundary Streamlines (d) Target Domain

Figure 1: Africa

3 Mapping Domains that Contain Voids

This research focuses on extending the above method to handle domains containing voids in 2D. The source domain must remain connected although it need not be simply connected. The method remains the same with the addition that boundary conditions are applied to the voids when calculating the potential field. Specifically, the aim is to change the flow of the potential field so that the void becomes manageable in the target domain.

To aid understanding of these additional conditions, the potential field can also be interpreted as a solution to the steady state heat equation. Thus, the potential value at a point is comparable to its temperature. In this framework, the centre acts as a heat sink while the domain boundary acts as a heat source. The streamlines now represent the flow of heat in the domain.

By applying a Dirichlet boundary condition to the void one can specify the potential – or "temperature" – along the void boundary. If the void is treated as an infinite conductor, the temperature values on the void boundaries are forced to be equal. This forces the entire void boundary to the same potential value, thus moving all these points to the same radial distance from the centre in the target domain. Streamlines thus flow either from the boundary directly to the centre, or from the boundary to one side of the void. These streamlines have then to be restarted on the other side of the void and allowed to reach the centre. There are then multiple streamlines approaching the centre from the void. This results in multiple angles but a single potential value. Therefore, the void is reduced to a single arc in the the target domain. This is shown in Figure 2 in the following section.

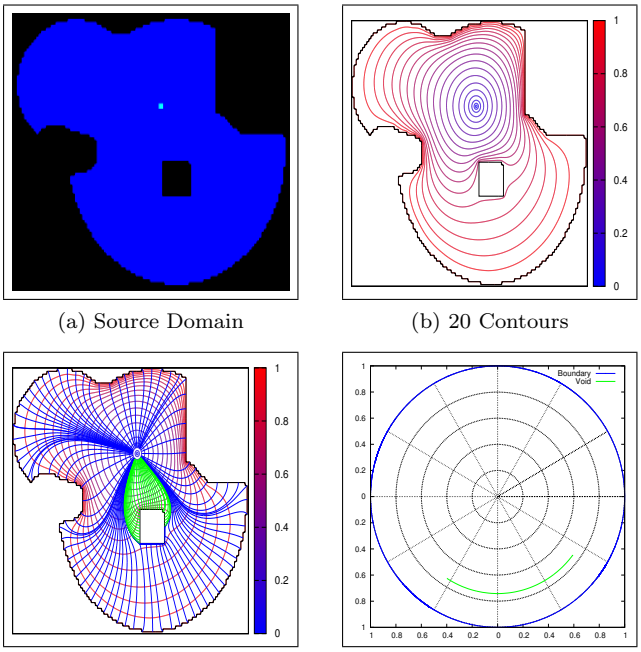
Another approach focuses on applying Neumann boundary conditions to the void. In physical terms, this equates to enforcing some heat flux over the void boundary. By treating the void as an insulator, one prohibits the flow of heat across the void. The flow is now forced around the void. One streamline heads from the boundary towards some fixed point on the void boundary. This streamline is once again restarted on the other side of the void. The boundary of the void now has a number of associated potential values, but there is only a single streamline originating at the void. Hence, the void is associated with a single angle and multiple potential values. Thus, it is reduced to a single radial line in the target domain. This is illustrated in Figure 3 in the following section.

The two methods can be used in conjunction in order to shrink the void to a single point in the target domain. Note, however, that these methods do not preserve the topology of the domain as the boundary of the void is collapsed. This may or may not be acceptable, depending on the application at hand.

If there is only a single void in the domain, the boundary of the void can be used as the centre. This method requires that the void is convex and is only able to handle the case of a single void in the domain.

4 Results

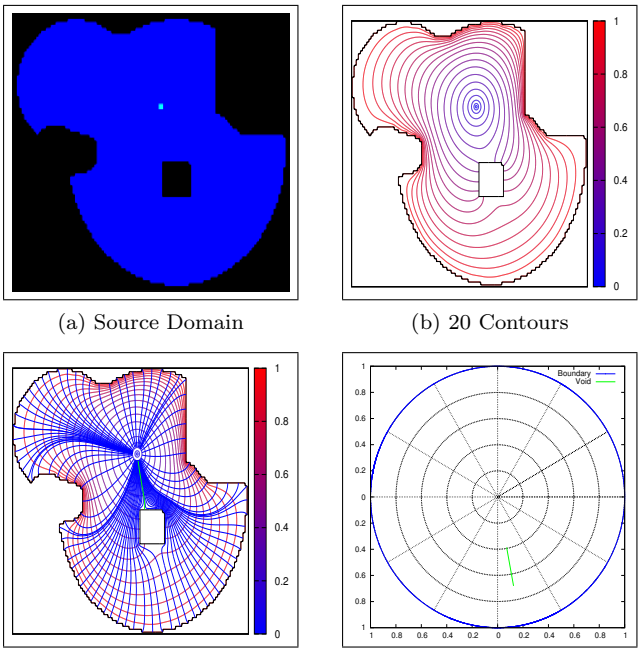
Figures 2d and 3d show how the streamlines are transformed in the target domain. The blue points correspond to the blue streamlines originating at the domain boundaries in Figures 2c and 3c. Only the end points of the streamlines have been marked to help emphasize the transformation of the void. The green points correspond to the endpoints of the green streamlines that were restarted on the other side of the void. Figure 2d shows how the endpoints of the void streamlines form an arc in the target domain. Similarly, Figure 3d shows how they form a radial line.



(a) Source Domain (b) 20 Contours

(c) Boundary Streamlines (d) Target Domain

Figure 2: Void as Infinite Conductor



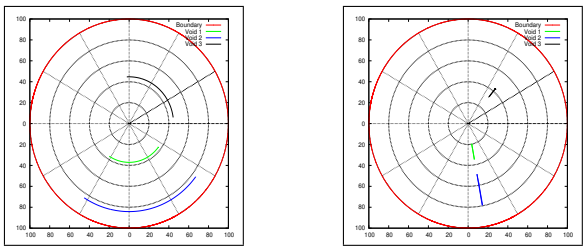
(a) Source Domain (b) 20 Contours

(c) Streamlines (d) Target Domain

Figure 3: Void as Insulator

If there are multiple voids in the source domain, they can be handled in the same way as before. If a streamline passes from one void to another, it should be handled by restarting that streamline on the other side of the second void. This process is continued until the streamline reaches the centre and an angle is assigned. If a streamline flows from one void to another, the void through which it must pass is said to eclipse the original one.

Figure 4 shows how eclipsing affects each of the aforementioned methods. The first method reduces each void into separate arcs. The second method is unlikely to encounter the eclipsing effect as it would require that the single streamline that reaches the first void also then reaches the second. If this does happen, however, there will be two separate lines in the target domain that have the same angular position, but different potential values.



(a) Infinite Conductor (b) Insulator

Figure 4: Eclipsed Voids

If streamlines are traced in the reverse direction – from the centre to the boundary – the method becomes sensitive to minor changes in the angle. This is particularly important when calculating the inverse mapping.

5 Applications

Remeshing and path planning were discussed in Section 1. The application of domain mapping to these problems is discussed below.

As the configuration space of a robot may be any arbitrary connected space, planning a path from one configuration to another is equivalent to finding some path between two points in this space. For example, domain mapping can be used to solve this problem in an arbitrary, connected, 2D space. First the points are mapped forward to the target domain. They are then joined in this domain with a path that avoids the shrunken voids. Finally, this path is mapped back to the original source domain.

In geometrically complicated domains, creating a mesh can be a difficult task. By mapping the domain to the a disk, the mesh can be easily constructed and transferred back to the source domain. It is important that the mesh is constructed around the points representing the voids. Without voids, the map is diffeomorphic. Thus calculations can be performed in the polar space and later mapped back to the initial domain. The benefit of this approach is that the simple Laplace equation is solved on the complicated domain, while a complicated PDE could then be solved on the simple convex domain. Figure 5 shows the result of a rectangular mesh that was constructed in the target domain and then transferred back to the Africa domain from Figure 1.

6 Conclusions and Further Work

Two new developments of the domain mapping procedure developed in [4, 5] have been presented and shown to be effective on test data. The relevance of these new methods to several applications has been indicated. Future work will involve the extension of these ideas to higher dimensional non simply connected regions, and the development of parallel code to deal with real–world size problems.

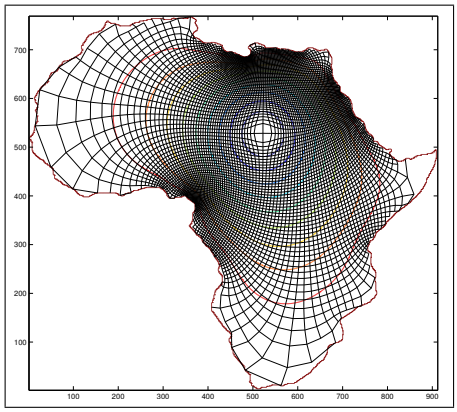


Figure 5: Remeshing

Acknowledgments

This research was funded by the Centre for High Performance Computing (CHPC) and the National Research Foundation (NRF) of South Africa. It forms part of the CHPC's Flagship Project on Computational Imaging and Remote Sensing.

References

[1] S. Axler, P. Bourdon, and W. Ramey. *Graduate Texts in Mathematics, Harmonic Function Theory*. Springer, New York, NY, Second edition, 2001.

[2] Y. Chang, B. Chen, W. Luo, and J. Huang. Skeleton-driven animation transfer based on consistent volume parameterization. In *Proceedings of Computer Graphics International 2006*, pages 78–89, 2006.

[3] S. Han, J. Xia, and Y. He. Hexahedral Shell Mesh Construction via Volumetric Polycube Map. In *Proceedings of the 14th ACM Symposium on Solid and Physical Modeling*, SPM '10, pages 127–136, New York, NY, USA, 2010. ACM.

[4] H. K. Voruganti, B. Dasgupta, and G. Hommel. Harmonic Function based Domain Mapping Method for General Domains. *WSEAS Transactions on Computers*, 5(10):2495–2502, Oct. 2006.

[5] H. K. Voruganti, V. Gupta, and B. Dasgupta. Domain Mapping for Spherical Parameterization of 3-D Domains using Harmonic Functions. In *IISc Centenary - International Conference on Advances in Mechanical Engineering (IC-ICAME), Bangalore, India*. 2-4 July 2008.

Author Index

SPONSORS

European Association for
Theoretical Computer Science
Italian Chapter