XIV Spanish Meeting on Computational Geometry

In Honor of Ferran Hurtado's 60th Birthday

Alcalá de Henares, June 27 – 30, 2011

Edited by

Pedro Ramos (Universidad de Alcalá)

Vera Sacristán (Universitat Politècnica de Catalunya)

Centre de Recerca Matemàtica
Campus de Bellaterra, Edifici C
08193 Bellaterra (Barcelona)

First edition: June 2011

ISSN 2014-2323 (printed edition)
ISSN 2014-2331 (electronic edition)

Legal deposit:

Presentation

This volume contains the abstracts or extended abstracts of the 11 invited talks and 52 communications presented at the *XIV Spanish Meeting on Computational Geometry*, held in Alcalá de Henares from June 27 to June 30, 2011.

The conference is part of the series *Encuentros de Geometría Computacional*. Since their start in Santander in 1990, the *Encuentros* have served not only as a meeting point for computational geometers working in Spain, but also as one of the crucial contributions towards the development of a vigorous Spanish computational geometry community. The *Encuentros* made it possible for all Spanish researchers in the area to get in touch with the most relevant international figures: with the Spanish speaking ones in the first years, and gradually with those of the entire international community.

This year, for the first time, the meeting has a fully international character, and the official language is English. The main reason for this is that the XIV Spanish Meeting has been dedicated to Prof. Ferran Hurtado on his 60th Birthday. Professor Hurtado has played a central role in the Spanish Computational Geometry community since its very beginning. The quantity and quality of international participants in this conference is an indisputable proof of his relevance at international level.

The organizers thank all the authors, invited speakers, and attendants for their participation in the meeting. We also wish to thank the members of the Scientific Committee for their careful revision of the papers, and the following institutions for their financial support: Ministerio de Ciencia e Innovación of the Spanish Government, Consolider Ingenio Mathematica (i-MATH), Universidad de Alcalá, Centre de Recerca Matemàtica, Societat Catalana de Matemàtiques, Departament de Matemàtica Aplicada II (UPC), Departamento de Matemáticas (UAH), Universitat Politècnica de Catalunya, and Real Sociedad Matemática Española.

The Editors

Committees

Scientific Committee

Manuel Abellanas
Franz Aurenhammer
David Avis
José Miguel Díaz-Bañez
Alfredo García
Stefan Langerman
Alberto Márquez (co-chair)
Joseph O'Rourke
Belén Palop
Pedro Ramos (co-chair)
David Rappaport
Eduardo Rivera-Campo
Günter Rote
Vera Sacristán
Francisco Santos
Toni Sellarès
William Steiger
Jorge Urrutia (co-chair)
David Wood

Organizing Committee

Juan Gerardo Alcázar
Marcos Marvá
David Orden (co-chair)
Pedro Ramos (co-chair)
Vera Sacristán
Fernando San Segundo
Raquel Viaña

Contents

Part I: Invited Talks

Part II: Communications

Session 1.1

Session 1.2

Session 1.3

Session 1.4

Session 2.1

Session 2.2

Session 2.3

Session 2.4

Session 3.4

Session 4.1

Session 4.2

Part I
Invited Talks

Generic distributed actuation of lattice-based modular robotic systems

Vera Sacristán[1]

[1] Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, Barcelona, Spain
vera.sacristan@upc.edu

Abstract. We solve a set of tasks for a general 3D-cube-lattice-based homogeneous robotic system, including self organizing —counting the number of modules, finding a master, computing the bounding box, forming a tree— as well as locomotion and reconfiguration. The algorithms proposed are generic and can be applied to a wide variety of particular systems. The modules of the robot are only assumed to move relative to each other, over the surface of the robot configuration. In addition, each module is capable of some small constant size computation, memory, and message passing from and to its neighbors. The algorithms proposed are inspired by cellular automata. Each module actuates in an autonomous and asynchronous way, based on local information. Besides formal correctness proofs and cost analysis of the proposed solutions —in terms of time, number of moves, and overall communication—, a simulator is presented.

1 Modular robotic systems

Since their introduction in the late 80's, modular robotic systems have been envisaged as a very promising field of research, addressing the issues of designing, building and controlling sets of multiple building units or modules that behave autonomously but in a collaborative way in order to perform collective tasks. When compared to fixed-morphology unique-purpose robots, modular robots have the advantages of being more versatile, as they can reconfigure to adapt to new environments and new tasks, more robust, as they can interchange parts and self-repair, and potentially less expensive, as their units can be reused and, in principle, massively produced. As a consequence, they are expected to be useful in building emergency structures, repairing unaccessible machinery, outer space missions, and even in current daily life [**16**]. The counterpart of this flexibility are the difficulties of actuation planning, which is the focus of this work.

Modular robots are frequently classified into homogeneous or heterogeneous, depending on whether their units are all equal or not as, although all structurally equal, some units may incorporate or carry special features such as grippers, cameras, antennas, etc. According to the locomotion autonomy of their units, two kinds of modular robotic systems can be considered, depending on whether each unit of the robot has full locomotion capability or locomotion is achieved by cooperation of the units, based on the movement of docking joints and links between them. Depending on the distribution of the modules in space when connected, modular robotic systems may be organized into lattice, chain or even hybrid architectures. Lattice-based modular robots include hexagonal, triangular, and squared or cubic.

For all these robotic systems, actuation algorithms have been developed with different goals: locomotion, reconfiguration, self-repairing, etc. Although many of them are centralized, the need has emerged for decentralized and local control of the actuation, as the

[1]Partially supported by projects MTM2009-07242 and Gen. Cat. DGR 2009SGR1040.

number of modules of the robot increases. Distributed algorithms have been designed for reconfiguring several systems, and more specifically for lattice-based modular robots such as Proteo, Fracta, Crystalline and Telecube, and large scale modular robots as Catoms.

2 Related work

Within this context, our approach finds its roots in the work of Beni [**2**], who proposed the conceptual model and discussed the parallelism and the differences between cellular automata and cellular robotic systems.

A more direct antecedent is the work of Hosokawa et al. [**9**], who developed a distributed control algorithm (for a specific square lattice-based modular robot design) inspired on cellular automata. Using the so-called sliding cube model, the authors proposed two simple sets of rules, to be locally executed by each robot unit, which allow to reconfigure a strip into a staircase and vice-versa.

Some years later, Butler, Kotay et al. [**4, 10**] proposed a fully decentralized actuation paradigm inspired on cellular automata. In their work, they address locomotion of a rectangular set of modules, with and without obstacles, reconfiguring a strip into a rectangle, and filling holes in 3D configurations.

Then Dumitrescu et al. [**6, 8**] studied fast locomotion cellular automata-like rules for horizontal (vertical) chains and diagonal snake-like formations, and proved universal reconfiguration between 2-dimensional horizontally convex and vertically convex configurations, in a linear number of synchronized time steps.

Later on, considering hand-coded local rules for reconfiguration a difficult task, Støy [**13, 14**] proposed a gradient technique for reconfiguring dense objects.

More recently, Kurokawa et al. [**11**] proposed specific sets of rules to M-TRAN for a particular set of robot reconfigurations. To the best of our knowledge, their work presents the first experimental execution of these strategies on real robot units, hence proving its realizability beyond experimental simulation.

A similar orientation inspired the work of Bojinov et al. [**3**], who proposed specific local rules to produce particular shapes on a 3-dimensional rhombic dodecahedron (Proteo), and Nguyen et al. [**12**] and Walter et al. [**15**] for the reconfiguration of hexagonal lattice-based robotic systems. Deway et al. [**5**] also use local rules for a distributed planner in the framework of their general metamodules' theory.

Our work is also related to that of Dumitrescu, Abel et al. [**1, 7**], who proved universal reconfiguration using the same basic moves, although by means of sequential and centralized algorithms.

3 Some challenges

The results obtained so far on cellular automata like strategies for lattice-based modular robots suggest several interesting challenges.

As has been said, previous work addresses locomotion and reconfiguration for specific shapes or has some restrictions on the configuration characteristics. One of the most appealing challenges in this regard is hence to prove the existence of shape independent completely distributed algorithms, i.e., sets of rules for locomotion and reconfiguration of any modular robot shape.

Another relevant challenge is to produce sets of rules that can be executed in a completely parallel set, as many of the current implementations are, in fact, sequential.

In particular, this implies that the rules should take care of collision detection. Going even further, complete asynchrony of module actuation would be desirable, for example by making communication between neighbors to take care of coordination issues.

One of the advantages of modular robots is the indistinguishability of the robot units. From that viewpoint, another interesting issue is to develop strategies that do not rely on id's, i.e., in a framework in which nothing allows to distinguish one module of the robot from another.

Some of the current procedures require the configuration of modules to have a master (one distinguished module), to know the number of modules of the cluster and the relative position of the bounding box, or to organize the robotic system in a tree structure. This arises the question of how these problems can be autonomously solved by the modules from scratch, without using id's or having an initial master.

4 Results

The work we present is an attempt to address these issues by means of a specific system of rules in a general framework that does not exploit specific characteristics of any particular robotic system and can be instantiated by many of the currently and potentially existent prototypes.

In this framework, a robot is a connected configuration of modules which are located in a 2- or 3-dimensional squared lattice. Modules are assumed to have a simple processor and some small memory, to be able to send and receive short messages to and from neighboring modules and to perform basic computations with a few counters and text strings. Both computation and memory are assumed to be of (small) constant size. Modules have the ability of attaching and detaching from lattice neighbors, and to perform three moves which allow them to walk along the boundary of the robot, namely sliding along straight portions of the boundary and turning convex and concave vertices or edges. Within this framework, our algorithms are completely distributed, local, and —with only one exception— asynchronous. They consist in sets of rules, each one having a priority, a precondition, and an actuation or postcondition. Rules are identical for all modules, and are executed by all them in parallel. In fact, modules are assumed to be homogeneous and indistinguishable, and the rules are run without the help of id's and without the need of any central controller or unique clock.

The problems we solve are essentially of two sorts: self-organizing and self-reconfiguring. In particular, we show a set of rules for each of the following problems: counting the number of modules and making all the units know it, choosing a master, computing the minimum bounding box and forming a tree. We also generalize previous locomotion results to the asynchronous context, and show and prove universal reconfiguration, i.e., transformation between any pair of configurations having the same number of modules, but in this case, at least one of the modules unavoidably needs to use linear memory to store the information of the goal shape.

Besides the correctness and complexity proofs, we also provide a simulator and experiments showing the behavior of the proposed sets of rules in practice, both in 2 and in 3 dimensions.

Acknowledgments

Different portions of the presented results are joint work with Oswin Aichholzer, Thomas Hackl, Birgit Vogtenhuber, Reinhard Wallner, and with Ferran Hurtado and Suneeta Ramaswami. One of the strategies was inspired by previous work with Greg Aloupis, Sébastien Collette, Mirela Damian, Erik D. Demaine, Robin Flatland, Stefan Langerman, Joseph O'Rourke, Suneeta Ramaswami, and Stefanie Wuhrer.

I wish to thank Joe O'Rourke for showing me into the field of reconfiguration problems related to modular robotic systems, which I really enjoy.

I owe much to Ferran Hurtado, with his contagious enthusiasm for research and his mastery of the area he dragged me into Computational Geometry (and he still does).

References

[1] Z. Abel and S. D. Kominers. Pushing hypercubes around, 2008.

[2] G. Beni. The concept of cellular robotic system. In *Proceedings of the IEEE International Symposium on Intelligent Control*, pages 57–62, 1988.

[3] H. Bojinov, A. Casal, and T. Hogg. Emergent structures in modular self-reconfigurable robots. In *Proceedings of the IEEE International Conference on Robotics & Automation*, pages 1734–1741, 2000.

[4] Z. Butler, K. Kotay, D. Rus, and K. Tomita. Generic decentralized control for lattice-based self-reconfigurable robots. *International Journal of Robotics Research*, 23:919–937, 2004.

[5] D. J. Dewey, M. P. Ashley-Rollman, M. De Rosa, S. C. Goldstein, T. C. Mowry, S. S. Srinivasa, P. Pillai, and J. Campbell. Generalizing metamodules to simplify planning in modular robotic systems. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1338–1345, 2008.

[6] A. Dumitrescu, I. Suzuki, and M. Yamashita. Formations for fast locomotion of metamorphic robotic systems. *International Journal of Robotics Research*, 23(6):583–593, 2004.

[7] A. Dumitrescu and J. Pach. Pushing squares around. In *Proceedings of the 20th Annual ACM Symposium on Computational Geometry*, pages 116–123, 2004.

[8] A. Dumitrescu, I. Suzuki, and M. Yamashita. Motion planning for metamorphic systems: Feasibiity, decidability, and distributed reconfiguration. *IEEE Transactions on Robotics and Automation*, 20(3), 2004.

[9] K. Hosokawa, T. Tsujimori, T. Fujii, H. Kaetsu, H. Asama, Y. Kuroda, and I. Endo. Self-organizing collective robots with morphogenesys in a vertical plane. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2858–2863, 1998.

[10] K. Kotay and D. Rus. Generic distributed assembly and repair algorithms for self-reconfiguring robots. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2362–2369, 2004.

[11] H. Kurokawa, K. Tomita, A. Kamimura, S. Kokaji, T. Hasuo, and S. Murata. Distributed self-reconfiguration of M-TRAN III modular robotic system. *The International Journal of Robotics Research*, 27:373–386, 2008.

[12] A. Nguyen, L. J. Guibas, and M. Yim. Controlled module density helps reconfiguration planning. In *Proceedings of the International Workshop on Algorithmic Foundations of Robotics (WAFR)*, pages 23–36, 2000.

[13] K. Støy. How to construct dense objets with self-reconfigurable robots. In *Proceedings of the European Robotics Symposium*, pages 27–37, 2006.

[14] K. Støy. Using cellular automata and gradients to control self-reconfiguration. *Robotics and Autonomous Systems*, 54:135–141, 2006.

[15] J. E. Walter, J. L. Welch, and N. M. Amato. Distributed reconfiguration of metamorphic robot chains. *Distributed Computing*, 17:171–189, 2004.

[16] M. Yim, W.-M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. S. Chirikjian. Modular self-reconfigurable robots systems: Challenges and opportunities for the future. *IEEE Robotics & Automation Magazine*, 14(1):43–52, 2007.

On 5-gons and 5-holes

Oswin Aichholzer[1], **Thomas Hackl**[1], **Birgit Vogtenhuber**[1]

[1] Institute for Software Technology, University of Technology, Graz, Austria
{oaich,thackl,bvogt}@ist.tugraz.at

Abstract. We consider an extension of a question of Erdős on the number of k-gons in a set of n points in the plane. Relaxing the convexity restriction we obtain results on 5-gons and 5-holes (empty 5-gons).

Introduction

Let S be a set of n points in general position in the plane. A k-gon is a simple polygon spanned by k points of S. A k-hole is an empty k-gon, that is, a k-gon which does not contain any points of S in its interior.

Erdős [**9**] raised the following questions for convex k-holes and k-gons. "What is the smallest integer $h(k)$ ($g(k)$) such that any set of $h(k)$ ($g(k)$) points in the plane contains at least one convex k-hole (k-gon)?"; and, more generally, "What is the least number $h_k(n)$ ($g_k(n)$) of convex k-holes (k-gons) determined by any set of n points in the plane?".

As already observed by Esther Klein, every set of 5 points determines a convex 4-hole (and thus 4-gon). Moreover, 9 points always contain a convex 5-gon and 10 points always contain a convex 5-hole, a fact proved by Harborth [**12**]. Only in 2007/08 Nicolás [**14**] and independently Gerken [**11**] proved that every sufficiently large point set contains a convex 6-hole, and it is well known that there exist arbitrarily large sets of points not containing any convex 7-hole [**13**]; see [**2**] for a brief survey.

In this paper we concentrate on 5-gons and 5-holes and generalize the above questions by allowing a 5-gon/5-hole to be non-convex. Thus, when referring to a 5-gon/5-hole, it might be convex or non-convex and we will explicitly state it when we restrict considerations to one of these two classes. Similar results for 4-holes can be found in [**3**]. For 4-gons there is a one-to-one relation to the rectilinear crossing number of the complete graph, and thus results can be found in the respective literature.

A set of five points in convex position obviously spans precisely one convex 5-gon. In contrast, already a set of only five points (with three extremal points) can span eight different 5-gons; see Figure 1 (left). This makes the considered questions more challenging (and interesting) than they might appear on a first glance.

Due to space limitations, all proofs are omitted in this extended abstract.

1 Small sets

For small point sets, Table 1 shows the numbers of 5-gons and 5-holes, respectively. Given are the minimum number of convex 5-gons/5-holes, the maximum number of non-convex 5-gons/5-holes, the minimum and maximum number of (general) 5-gons/5-holes, and, for easy comparison, the number of 5-tuples.

For counting convex 5-gons/5-holes, it is easy to see that their number is maximized by sets in convex position and gives $\binom{n}{5}$. Of course these sets do not contain any non-convex 5-gons/5-holes. From Table 1 we also see that the minimum number of general

n	number of 5-gons: convex min	non-convex max	general min	general max	number of 5-holes: convex min	non-convex max	general min	general max	$\binom{n}{5}$
5	0	8	1	8	0	8	1	8	1
6	0	48	6	48	0	31	6	31	6
7	0	156	21	157	0	76	21	77	21
8	0	408	56	410	0	157	56	160	56
9	1	900	126	909	0	288	126	292	126
10	2	1776	252	1790	1	492	252	501	252
11	7	3192	462	3228	2	779	462	802	462

TABLE 1. Number of 5-gons and 5-holes for $n = 5...11$ points.

5-gons and 5-holes is $\binom{n}{5}$ for $5 \leq n \leq 11$. While for 5-gons this is obviously true in general (a convex 5-tuple has exactly one polygonization, while a non-convex 5-tuple has at least four), this is not the case for 5-holes. In fact, we will show that, for sufficiently large n, the convex set maximizes the number of 5-holes; see Theorem 3.6.

2 Five-gons

The rectilinear crossing number $\bar{cr}(S)$ of a set S of n points in the plane is the number of proper intersections in the drawing of the complete straight line graph on S. It is easy to see that the number of convex 4-gons is equal to $\bar{cr}(S)$ and is thus minimized by sets minimizing the rectilinear crossing number, a well-known, difficult problem in discrete geometry; see **[7]** and **[10]** for details. Tight values for the minimum number of convex 4-gons are known for $n \leq 27$ points; see e.g. **[1]**. Asymptotically we have at least $c_4\binom{n}{4} = \Theta(n^4)$ convex 4-gons, where c_4 is a constant in the range $0.379972 \leq c_4 \leq 0.380488$. As any 4 points in non-convex position span three non-convex 4-gons, we get $3\binom{n}{4} - 3\bar{cr}(S)$ non-convex and $3\binom{n}{4} - 2\bar{cr}(S)$ general 4-gons for a set S. Thus, sets which minimize the rectilinear crossing number also minimize the number of convex 4-gons, and maximize both the number of non-convex 4-gons and the number of general 4-gons.

Surprisingly, a similar relation can be obtained for the number of non-convex 5-gons. To see this, consider the three combinatorial different possibilities (order types) of arranging 5 points in the plane, as depicted in Figure 1 (right). The proof of the following theorem is based on relations between the number of 5-gons and the numbers of crossings of these configurations.

Theorem 2.1. *Let S be a set of $n \geq 5$ points in the plane in general position. Then S contains $10\binom{n}{5} - 2(n-4)\bar{cr}(S)$ non-convex 5-gons.*

Taking the constant c_4 for the rectilinear crossing number into account, we see that asymptotically we can have up to $10\binom{n}{5} - 2(n-4)c_4\binom{n}{4} = 10(1-c_4)\binom{n}{5}$ non-convex 5-gons. This number is obtained for point sets minimizing the rectilinear crossing number and by a factor ≈ 6.2 larger than the maximum number of convex 5-gons.

For the number of convex 5-gons, no simple relation to the rectilinear crossing number is possible: There exist two different sets (order types) S_1 and S_2, both of cardinality 6 with 4 extremal points, with $\bar{cr}(S_1) = \bar{cr}(S_2) = 8$, where S_1 contains one convex 5-gon, while S_2 does not contain any convex 5-gon.

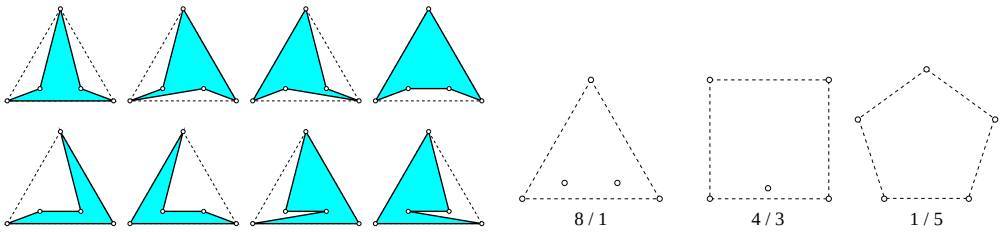


FIGURE 1. Left: The eight different (non-convex) 5-gons spanned by a set of five points with three extremal points. Right: The three order types for $n = 5$. For each set its number of different 5-gons and the number of crossings for the complete graph is shown.

3 Five-holes

3.1 A new lower bound for the number of convex 5-holes

Let $h_5(S)$ denote the number of convex 5-holes of a point set S, and let $h_5(n) = \min_{|S|=n} h_5(S)$ be the number of convex 5-holes any point set of cardinality n has to have. The best upper bound $h_5(n) \leq 1.0207n^2 + o(n^2)$ can be found in [**6**]. The previous best lower bound $h_5(n) \geq \lfloor \frac{n-4}{6} \rfloor$ has been obtained by Bárány and Károlyi [**5**].

Here we give a slight improvement on this bound, which still remains linear in n. It is based on an observation by Dehnhardt [**8**] that every set of 12 points contains at least three convex 5-holes.

Theorem 3.1. *Let S be a set of $n \geq 12$ points in the plane in general position. Then $h_5(n) \geq 3\lfloor \frac{n-4}{8} \rfloor$.*

3.2 A lower bound for the number of (general) 5-holes

We obtained the following observation for general 5-holes by checking all 14 309 547 according point sets from the order type data base [**4**].

Observation 3.2. Let S be a set of $n = 10$ points in the plane in general position, and $p_1, p_2 \in S$ two arbitrary points of S. Then S contains at least 34 five-holes having p_1 and p_2 among their vertices.

This observation implies the following result, using a similar approach as in [**3**].

Theorem 3.3. *Let S be a set of $n \geq 10$ points in the plane in general position. Then S contains at least $17n^2 - O(n)$ five-holes.*

3.3 Maximizing the number of (general) 5-holes

The results for small sets shown in Table 1 suggest that the number of (general) 5-holes is minimized by sets in convex position. We not only show that this is in fact not the case, but rather prove the contrary: For sufficiently large n, sets in convex position maximize the number of 5-holes.

Lemma 3.4. *A point set S with triangular convex hull and i interior points contains at most $4i + 5$ five-holes which have the three extreme points among their vertices.*

Lemma 3.5. *Let Γ be a non-empty convex quadrilateral in S. There are at most four 5-holes spanned by the four vertices of Γ plus a point of S in the interior of Γ.*

Considering the size of the convex hull of each 5-tuple, these two lemmas lead to the following theorem.

Theorem 3.6. *For $n \geq 86$, the number of 5-holes is maximized by a set of n points in convex position.*

4 Conclusion

In this abstract we presented several results for a variant of a classic Erdős–Szekeres-type problem: counting general 5-gons and 5-holes. The following questions remain open: What is the maximum number of general 5-gons? Is there a super-linear lower bound for the number of convex 5-holes (cf. Theorem 3.1) or a super-quadratic lower bound for the number of general 5-holes (cf. Theorem 3.3)?

Acknowledgments

Research supported by the FWF (Austrian Fonds zur Förderung der Wissenschaftlichen Forschung) under grant S9205-N12, NFN Industrial Geometry.

References

[1] O. Aichholzer. *On the rectilinear crossing number.* http://www.ist.tugraz.at/aichholzer/research/rp/triangulations/crossing/.

[2] O. Aichholzer. *[Empty] [colored] k-gons – Recent results on some Erdős–Szekeres type problems.* In Proc. XIII Encuentros de Geometría Computacional, pages 43–52, Zaragoza, Spain, 2009.

[3] O. Aichholzer, R. Fabila-Monroy, H. González-Aguilar, T. Hackl, M. A. Heredia, C. Huemer, J. Urrutia and B. Vogtenhuber. 4-*Holes in point sets.* In Proc. 27th European Workshop on Computational Geometry EuroCG'11, pages 115–118, Morschach, Switzerland, 2011.

[4] O. Aichholzer and H. Krasser. *The point set order type data base: A collection of applications and results.* In Proc. 13th Canadian Conference on Computational Geometry CCCG'01, pages 17–20, Waterloo, Ontario, Canada, 2001.

[5] I. Bárány and Gy. Károlyi. *Problems and results around the Erdős–Szekeres convex polygon theorem.* Discrete and Combinatorial Geometry, LNCS 2098:91–105, 2001.

[6] I. Bárány and P. Valtr. *Planar point sets with a small number of empty convex polygons.* Studia Scientiarum Mathematicarum Hungarica, 41(2):243-269, 2004.

[7] P. Brass, W. Moser and J. Pach. *Research Problems in Discrete Geometry*, Springer, 2005.

[8] K. Dehnhardt. *Leere konvexe Vielecke in ebenen Punktmengen.* TU Braunschweig, Germany, 1987.

[9] P. Erdős. *Some more problems on elementary geometry.* Austral. Math. Soc. Gaz., 5:52–54, 1978.

[10] P. Erdős and R. K. Guy. *Crossing number problems.* Amer. Math. Monthly, 88:52–58, 1973.

[11] T. Gerken. *Empty convex hexagons in planar point sets.* Discrete and Computational Geometry, 39(1-3):239–272, 2008.

[12] H. Harborth. *Konvexe Fünfecke in ebenen Punktmengen.* Elemente Math., 33:116–118, 1978.

[13] J. D. Horton. *Sets with no empty convex 7-gons.* Canad. Math. Bull., 26(4):482–484, 1983.

[14] C. M. Nicolás. *The empty hexagon theorem.* Discrete and Computational Geometry, 38(2):389–397, 2007.

The art of polyhedra

Jin Akiyama[1]

[1] Research Institute of Educational Development, Tokai University, 2-28-4 Tomigaya, Shibuya, Tokyo, 151-8677, Japan
ja@jin-akiyama.com

Abstract. One of the oldest, liveliest branches of mathematics, the study of polyhedra, can trace its roots back to the work of the Greeks. Nevertheless, it abounds in unsolved problems that even a high school student can understand and appreciate. In this talk, we aim to give recent results on polyhedra, together with several artworks created by applying these results.

1 Reversible polygons and polyhedra

Our work on reversible solids (or polygons) was inspired by the famous Dudeney Puzzle. A convex polygon (or polyhedron) P is said to be *reversible to* Q if P can be dissected and turned inside out to form another convex polygon (or polyhedron) Q. We call these P and Q a *reversible pair*.

Theorem 1.1 ([1]). *If P is reversible, then P tiles the plane by translation and rotation.*

Theorem 1.2 ([1]). *Let P and Q be a reversible pair, and let α and β be tilings by P and Q, respectively. Then an appropriate superimposition of two tilings α and β gives the way how to dissect P to make Q.*

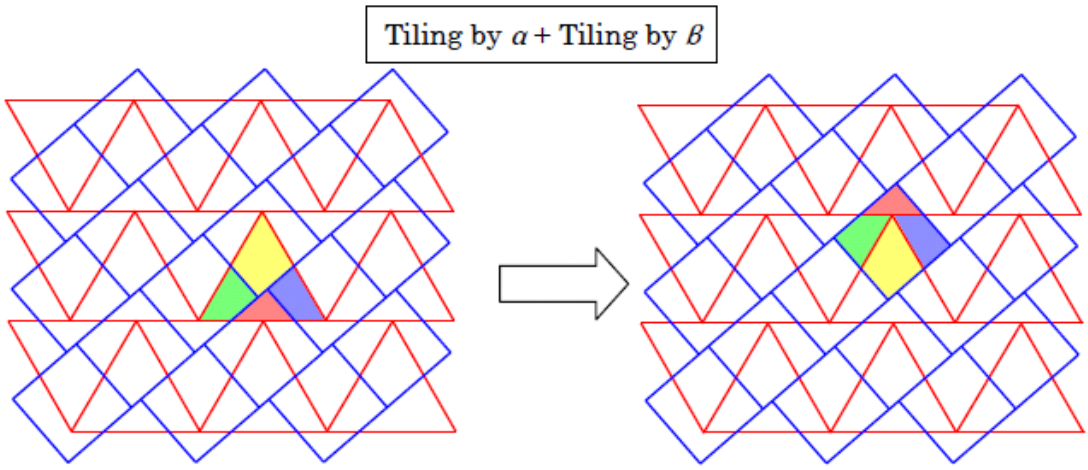


FIGURE 1

A *parallelohedron* is a convex polyhedron which tiles 3-dimensional space (i.e., a *space-filler*) by translations only.

Fedorov established in **[2]** that there are exactly five families $F_1 \sim F_5$ of parallelohedra, namely, *parallelepiped* F_1, *rhombic dodecahedron* F_2, *hexagonal prism* F_3, *elongated rhombic dodecahedron* F_4, and *truncated octahedron* F_5. Note that each family contains infinitely many different shapes of polyhedra, since applying affine transformations to any of them will not affect the space-filling property.

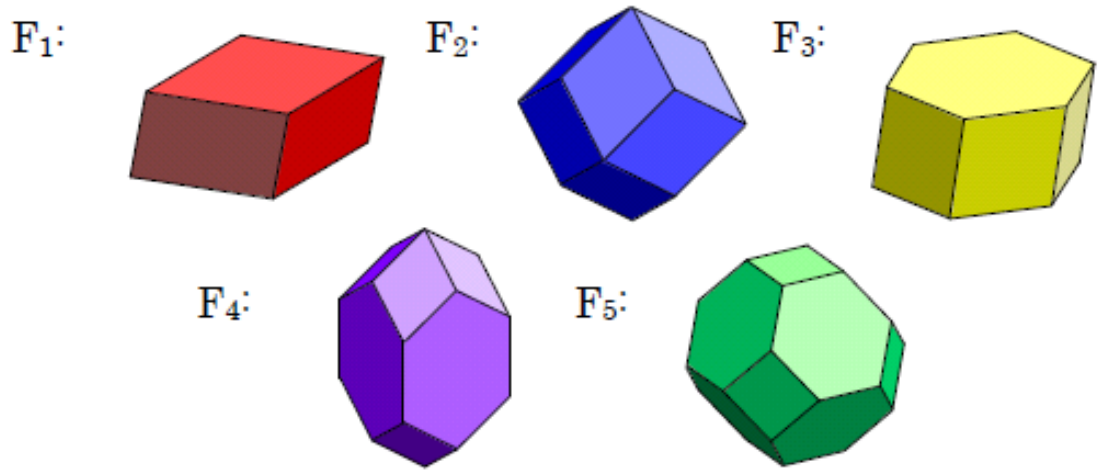


FIGURE 2

Theorem 1.3. *For every pair F_i, F_j $(1 \le i, j \le 5)$ of families of parallelohedra, there exist $P \in F_i$ and $Q \in F_j$ such that P is reversible to Q.*

2 Elements of parallelohedra

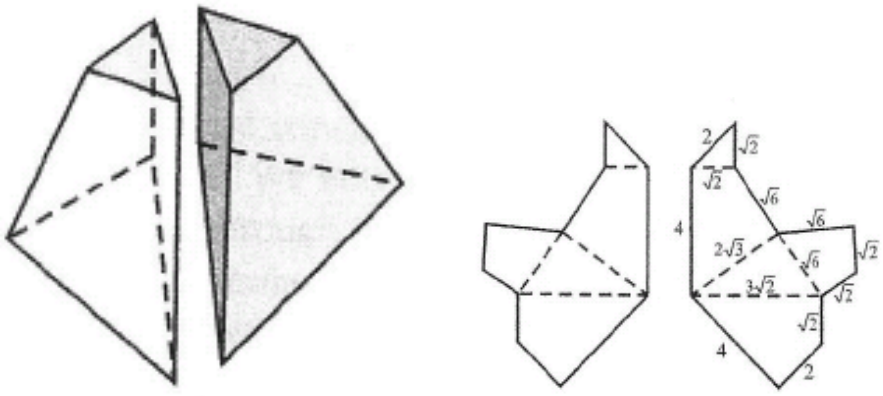
Let Σ be a set of polyhedra. A set Ω of polyhedra is said to be an *element set* for Σ, denoted by $\varepsilon(\Sigma)$, if each polyhedron in Σ is the union of a finite number of polyhedra in Ω, i.e.,

$$\text{for all } P \in \Sigma,\ P = \cup n_i \sigma_i, \text{ where } n_i \in \mathbb{Z}_{\ge 0} \text{ and } \sigma_i \in \Omega.$$

The *element number* of the set Σ of polyhedra, denoted by $e(\Sigma)$, is the minimum cardinality of the element sets for Σ, i.e., $e(\Sigma) = \min |\Omega|$, where the minimum is taken over all possible element sets $\Omega \in \varepsilon(\Sigma)$.

A *pentadron* is either one of the pentahedra shown in Figure 3(a), with nets shown in Figure 3(b).

Theorem 2.1 ([4]). *The element number of the set of all parallelohedra is* 1. *Each parallelohedron can be constructed with a finite number of copies of the pentadron by face to face gluing. See Table* 1.



(a) Two congruent pentadra (b) Nets of pentadra

FIGURE 3

Parallelohedron	Number of pentadra
Cube	96
Rhombic dodecahedron	192
Skewed hexagonal prism	144
Elongated rhombic dodecahedron	384
Truncated octahedron	48

TABLE 1. Number of pentadra in the parallelohedra.

3 Element numbers for regular n-polytopes

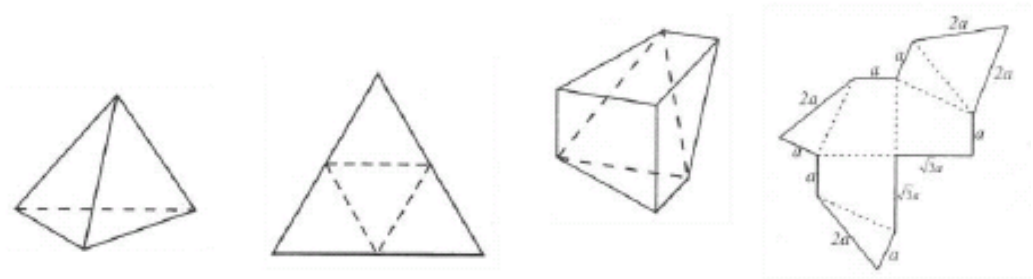
Theorem 3.1 ([4]). *Let* Π *be the set of the five Platonic solids, and denote by* σ_1, σ_2, σ_3, σ_4 *a regular tetrahedron, an equihepta, a golden tetra, and a roof, respectively. Then* $\Phi = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4\}$ *is an element set for* Π*, and the decomposition of each Platonic solid into these elements is summarized in Table* 2.

Platonic solid	Decomposition into elements
Regular tetrahedron	σ_1
Cube	$\sigma_1 \cup 4(\sigma_2 \cup 3\sigma_3) = \sigma_1 \cup 4\sigma_2 \cup 12\sigma_3$
Regular octahedron	$8(\sigma_2 \cup 3\sigma_3) = 8\sigma_2 \cup 24\sigma_3$
Regular dodecahedron	$\sigma_1 \cup 4\sigma_2 \cup 12\sigma_3 \cup 6\sigma_4$
Regular icosahedron	$8\sigma_2$

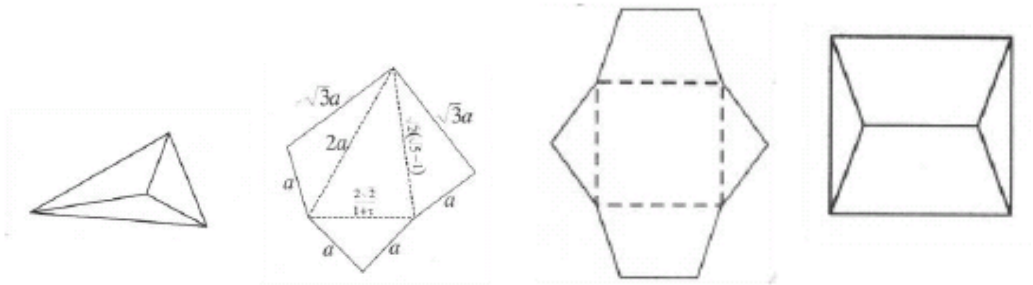
TABLE 2. Decomposition of the Platonic solids.

Let Π_n be a set of all regular n-polytopes. Then we have the following result:

Theorem 3.2. *The element number for regular* 4*-polytopes is* 4*, and the element number for regular* n*-polytopes is* 3 *for all* $n \geq 5$*. See Table* 3.



(σ_1) Regular tetrahedron (σ_2) Equihepta



(σ_3) Golden tetra (σ_4) Roof

FIGURE 4. The four elements for the Platonic solids.

Dimension n	Number of regular polytopes	Element number $e(\Pi_n)$
2	∞	∞
3	5	4
4	6	4
≥ 5	3	3

TABLE 3

References

[1] J. Akiyama and G. Nakamura, Congruent Dudeney dissections of triangles and convex quadrilaterals – All hinge points interior to the sides of the polygons, *Discrete and Computational Geometry, The Goodman–Pollack Festchrift (B. Aronov, S. Basu, J. Pach and M. Sharir, eds.), Algorithms and Combinatorics*, Springer-Verlag, **25** (2003), 43–63.

[2] E. S. Fedorov, An introduction to the theory of figures, in: *Notices of the Imperial Mineralogical Society (St. Petersburg)* Ser. 2, Vol. 21, 1885, 1–279. Republished with comments by Akad. Nanak. SSSR, Moscow, 1953, in Russian.

[3] J. Akiyama, M. Kobayashi, H. Nakagawa, G. Nakamura, and I. Sato, Atoms for parallelohedra, to appear in *Geometry – Intuitive, Discrete and Convex*, Bolyai Soc. Math. Studies (J. Pach et al., eds.), Springer-Verlag.

[4] J. Akiyama, I. Sato, The element number of the convex regular polytopes, *Geometriae Dedicata*, **151:1** (2011), 269–278.

[5] J. Akiyama, H. Maehara, G. Nakamura, and I. Sato, Element number of the Platonic solids, *Geometriae Dedicata*, **145:1** (2010), 181–193.

On edge-disjoint empty triangles of point sets

Javier Cano[1], **L. F. Barba**[1], **Toshinori Sakai**[2], **Jorge Urrutia**[3]

[1] Posgrado en Ciencia e Ingeniería de la Computación, Universidad Nacional Autónoma de México
{j_cano,l.barba}@uxmcc2.iimas.unam.mx

[2] Research Institute of Educational Development, Tokai University, Japan
sakai@tokai-u.jp

[3] Instituto de Matemáticas, Universidad Nacional Autónoma de México
urrutia@matem.unam.mx

Abstract. Let P be a set of points in the plane in general position. Any three points $x, y, x \in P$ determine a triangle $\Delta(x, y, z)$ of the plane. We say that $\Delta(x, y, z)$ is empty if its interior contains no element of P. In this paper we study the following problems: What is the size of the largest family of edge-disjoint triangles of a point set? How many triangulations of P are needed to cover all the empty triangles of P? What is the largest number of edge-disjoint triangles of P containing a point q of the plane in their interior?

Introduction

Let P be a set of n points on the plane in general position. A *geometric graph* on P is a graph G whose vertices are the elements of P, two of which are adjacent if they are joined by a straight line segment. We say that G is *plane* if it has no edges that cross each other. A *triangle* of G consists of three points $x, y, z \in P$ such that xy, yz, and zx are edges of G; we will denote it as $\Delta(x, y, z)$. If in addition $\Delta(x, y, z)$ contains no elements of P in its interior, we say that it is *empty*.

In a similar way, we say that, if $x, y, z \in P$, then $\Delta(x, y, z)$ is a *triangle* of P, and that xy, yz, and zx are the *edges* of $\Delta(x, y, z)$. If $\Delta(x, y, z)$ is empty, it is called a 3-*hole* of P. A 3-hole of P can be thought of as an empty triangle of the complete geometric graph $\mathcal{K}_P$ on P. We remark that $\Delta(x, y, z)$ will denote a triangle of a geometric graph, and also a triangle of a point set.

A well-known result in graph theory says that, for $n = 6k + 1$ or $n = 6k + 3$, the edges of the complete graph K_n on n vertices can be decomposed into a set of $\binom{n}{2}/3$ edge-disjoint triangles. These decompositions are known as *Steiner triple systems* [**18**]; see also Kirkman's schoolgirl problem [**12, 17**]. In this paper, we address some variants of that problem, but for geometric graphs.

Given a point set P, let $\delta(P)$ be the size of the largest set of edge-disjoint empty triangles of P. It is clear that, if P is in convex position and it has $n = 6k + 1$ or $n = 6k + 3$ elements, then $\delta(P) = \binom{n}{2}/3$. On the other hand, we prove that, for some point sets, namely Horton point sets, $\delta(P)$ is $O(n \log n)$.

We then study the problem of covering the empty triangles of point sets with as few triangulations of P as possible. For point sets in convex position, we prove that we need essentially $\binom{n}{3}/4$ triangulations; our bound is tight. We also show that there are point

[1]Partially supported by project SEP-CONACYT of Mexico, Proyecto 80268.
[3]Partially supported by projects MTM2006-03909 (Spain) and SEP-CONACYT 80268 (Mexico).

sets P for which $O(n \log n)$ triangulations are sufficient to cover all the empty triangles of P for a given point set P.

Finally, we consider the problem of finding a point contained in the interior of many edge-disjoint triangles of P. We prove that for any point set there is a point contained in at least $n^2/12$ edge-disjoint triangles. Furthermore, any point in the plane is contained in at most $n^2/9$ edge-disjoint triangles of P, and this bound is sharp. In particular, we show that this bound is attained when P is the set of vertices of a regular polygon.

Preliminary work

The study of counting and finding k-holes in point sets has been an active area of research since Erdős and Szekeres [**6, 7**] asked about the existence of k-holes in planar point sets. It is known that any point set with at least ten points contains 5-holes; e.g. see [**9**]. Horton [**10**] proved that for $k \geq 7$ there are point sets containing no k-holes. The question of the existence of 6-holes remained open for many years, but recently Nicolás [**14**] proved that any point set with sufficiently many points contains a 6-hole. A second proof of this result was subsequently given by Gerken [**8**].

The study of properties of the set of triangles generated by point sets on the plane has been of interest for many years. Let $f_k(n)$ be the minimum number of k-holes that a point set has. Clearly a point set has a minimum of $f_3(n)$ empty triangles. Katchalski and Meir [**11**] proved that $\binom{n}{2} \leq f_3(n) \leq kn^2$ for some $k < 200$; see also Purdy [**16**]. Their lower bounds were improved by Dehnhardt [**4**] to $n^2 - 5n + 10 \leq f_3(n)$. He also proved that $\binom{n-3}{2} + 6 \leq f_4(n)$. Point sets with few k-holes for $3 \leq k \leq 6$ were obtained by Bárány and Valtr [**2**]. The interested reader can read [**13**] for a more accurate picture of the developments in this area of research.

Chromatic variants of the Erdős-Szekeres problem have recently been studied by Devillers, Hurtado, Károly, and Seara [**5**]. They proved among other results that any bi-chromatic point set contains at least $\frac{n}{4} - 2$ compatible monochromatic empty triangles. Aichholzer *et al.* [**1**] proved that every bi-chromatic point set contains $\Omega(n^{5/4})$ empty monochromatic triangles; this bound was improved by Pach and Tóth [**15**] to $\Omega(n^{4/3})$. Due to lack of space, we will omit the proofs of all of our results.

1 Sets of edge-disjoint empty triangles in point sets

Let P be a set of n points on the plane, and $\delta(P)$ the size of the largest set of edge-disjoint empty triangles of the complete graph $\mathcal{K}(P)$ on P. For any integer $k \geq 1$, let H_k denote the Horton set with 2^k points; see [**10**]. We will prove:

Theorem 1.1. *Let $n = 2^k$, and let H_k be the Horton set with $n = 2^k$ elements. Then $\delta(H_k)$ is $O(n \log n)$.*

Conjecture 1.2. *Every point set P in general position with n elements contains a set with at least $O(n \log n)$ edge-disjoint empty triangles.*

2 Covering the triangles of point sets with triangulations

An empty triangle t of a point set P is *covered* by a triangulation T of P if one of the faces of T is t. In this section we consider the following problem:

Problem 2.1. *How many triangulations of a point set are needed so that each empty triangle of P is covered by at least one triangulation?*

We start by studying Problem 2.1 for point sets in convex position, and then for point sets in general position. We will prove first:

Theorem 2.2. *The set of triangles of any convex polygon can be covered with*

(1) $\frac{1}{4}\left[\binom{n}{3} + \frac{n(n-2)}{2}\right]$ *triangulations for n even, and*

(2) $\frac{1}{4}\left[\binom{n}{3} + \frac{n(n-1)}{2}\right]$ *triangulations for n odd.*

This bound is tight.

Thus the number of triangulations needed to cover all the triangles of P is asymptotically $\binom{n}{3}/4$. The next result follows trivially:

Corollary 2.3. *Let P be a set of n points in convex position, and p any point in the interior of $CH(P)$. Then p belongs to the interior of at most $\frac{1}{4}\binom{n}{3} + O(n^2)$ triangles of P.*

Next we prove:

Theorem 2.4. *$\Theta(n \log n)$ triangulations of H_k are necessary and sufficient to cover the set of empty triangles of H_k.*

Conjecture 2.5. *At least $\Omega(n \log n)$ triangulations are needed to cover all the empty triangles of any point set with n points.*

3 A point in many edge-disjoint triangles

The problem of finding a point contained in many triangles of a point set was solved by Boros and Füredi [**3**]. They proved:

Theorem 3.1. *For any set P of n points in general position, there is a point in the interior of the convex hull of P contained in $\frac{2}{9}\binom{n}{3} + O(n^2)$ triangles of P. The bound is tight.*

We consider the following problem:

Problem 3.2. *Let P be a set of points on the plane in general position, and $q \notin P$ a point of the plane. What is the largest number of edge-disjoint triangles of P such that q belongs to the interior of all of them?*

We will prove:

Theorem 3.3. *In any point set in general position there is a point q for which the inequalities $\frac{1}{12}n^2 \leq \tau(q) \leq \frac{1}{9}n^2$ hold. Moreover, $\tau(q) \leq \frac{1}{9}n^2$ for every q.*

3.1 Regular polygons

By Theorem 3.3, any point in the interior of the convex hull of a point set is contained in at most $n^2/9$ edge-disjoint triangles of P. We now show that the upper bound in Theorem 3.3 is achieved when P is the set of vertices of a regular polygon. Proving this result proved to be a nice challenging problem. In what follows, we will assume that $n = 9m$ with $m \geq 1$. We will prove:

Theorem 3.4. *Let P be the set of vertices of a regular polygon with $n = 9m$ vertices, and let c be its center. If m is odd, then $|\tau(c)| \geq \frac{1}{9}n^2$, and if m is even, then $|\tau(c)| \geq \frac{1}{9}n^2 - n$.*

We conclude our paper by proving:

Theorem 3.5. *There are point sets P such that every $q \notin P$ is contained in at most a linear number of empty edge-disjoint triangles of P. This bound is tight.*

We conclude with the following:

Conjecture 3.6. *Let P be a set of n points in general position on the plane. Then there is a point q on the plane which is contained in at least $\log n$ edge-disjoint triangles of P.*

References

[1] O. Aichholzer, R. Fabila-Monroy, D. Flores-Peñaloza, T. Hackl, C. Huemer, and J. Urrutia. Empty monochromatic triangles. *Computational Geometry, Theory and Applications*, 42:934–938, 2009.

[2] I. Bárány and P. Valtr. Planar point sets with a small number of empty convex polygons. *Studia Scientiarum Mathematicarum Hungarica*, 41(2):243–266, 2004.

[3] E. Boros and Z. Füredi. The number of triangles covering the center of an n-set. *Geom. Dedicata*, 17:69–77, 1984.

[4] K. Dehnhardt. Leere konvexe Vielecke in ebenen Punktmengen. *Dissertation, TU Braunschweig*, 1987.

[5] O. Devillers, F. Hurtado, G. Károlyi, and C. Seara. Chromatic variants of the Erdős–Szekeres Theorem. *Computational Geometry, Theory and Applications*, 26(3):193–208, 2003.

[6] P. Erdős. Some more problems on elementary geometry. *Austral. Math. Soc. Gaz.*, 5:52–54, 1978.

[7] P. Erdős and G. Szekeres. A combinatorial problem in geometry. *Compositio Math.*, 2:463–470, 1935.

[8] T. Gerken. Empty convex hexagons in planar point sets. *Discrete & Computational Geometry*, 39(1-3):239–272, 2008.

[9] H. Harborth. Konvexe Fünfecke in ebenen Punktmengen. *Elem. Math.*, 33:116–118, 1978.

[10] J. D. Horton. Sets with no empty convex 7-gons. *Canad. Math. Bull.*, 26:482–484, 1983.

[11] M. Katchalski and A. Meir. On empty triangles determined by points in the plane. *Acta. Math. Hungar.*, 51:323–328, 1988.

[12] T. Kirkman. On a problem in combinatorics. *Cambridge Dublin Math. J.*, 2:191–204, 1847.

[13] W. Morris and V. Soltan. The Erdős–Szekeres problem on points in convex position – a survey. *Bulletin (new series) of the American Mathematical Society*, 37(4):437–458, 2000.

[14] C. M. Nicolás. The empty hexagon theorem. *Discrete & Computational Geometry*, 38:389–397, 2007.

[15] J. Pach and G. Tóth. Monochromatic empty triangles in two-colored point sets. In *Geometry, Games, Graphs and Education: the Joe Malkevitch Festschrift*, pages 195–198, COMAP, Bedford, MA, 2008.

[16] G. Purdy. The minimum number of empty triangles. *AMS Abstracts*, 3:318, 1982.

[17] D. Ray-Chaudhuri and R. Wilson. Solution to Kirkman's schoolgirl problem. *Proc. Sym. Pure Math.*, Amer. Math. Soc., 19:187–204, 1971.

[18] H. J. Ryser. In *Combinatorial Mathematics*, pages 99–102, Buffalo, NY, Math. Assoc. Amer., 1963.

Geometric puzzles: algorithms and complexity

Erik D. Demaine[1]

[1] MIT Computer Science and Artificial Intelligence Laboratory, 32 Vassar St., Cambridge, MA 02139, USA
edemaine@mit.edu

I love computational geometry because the problems and solutions are fun and often tangible. Puzzles are one way to express these two aspects of geometry. Puzzles are also a great source of computational geometry problems: which puzzles can be solved and/or designed efficiently using algorithms? Proving puzzles to be computationally intractable (NP-hard or worse) leads to a more mathematical sort of puzzle, designing gadgets and reductions. I will describe a variety of algorithmic and complexity results on geometric puzzles, focusing on more playful and recent results.

The relative neighborhood graph: an interdisciplinary paradigm

Godfried T. Toussaint[1]

[1] Harvard University, Cambridge, MA, USA
godfried@cs.mcgill.ca

Dedicated to Ferran Hurtado on his 60th birthday

The relative neighborhood graph of a collection of objects assigns an edge to a pair of objects (A, B), provided that no other object is closer to both A and B than A and B are to each other. This graph was originally proposed for the purpose of extracting the visual perceptual structure of a two-dimensional dot pattern [**1**]. During the past thirty-one years, the relative neighborhood graph has been applied to a multiplicity of different disciplines, and sometimes to a variety of different problems within a discipline. Here some of these applications are reviewed, including: wireless network communications, archaeological network analysis, grid typification in cartography, data mining for geographic information systems, shape analysis, image morphology, polygon decomposition, the extraction of primal sketches in computer vision, the reduction of the size of the training sets in instance-based machine learning, the design of non-parametric decision rules, support-vector machines, cluster analysis, manifold learning, the design of nonparametric tests of the independence of dissimilarity matrices, the design of data-depth measures, testing class separability, estimating two-dimensional voids in the cold dark matter universe, multidimensional data-base indexing, image retrieval, adaptive grid generation for solving partial differential equations, clinical case retrieval in health-care systems, modeling road networks in transportation science, modeling leaf venation patterns in biology, plasmodium machines, swarm intelligence, distributed motion coordination, visualizing metabolic reactions in chemistry, tracking defects in crystal structures, and developing visualization tools such as topological zooming as well as Tukey and Tukey *scagnostics*.

References

[1] G. T. Toussaint, The relative neighborhood graph of a finite planar set, *Pattern Recognition* **12** (1980), 261–268.

Local transformations in geometrical structures

Alberto Márquez[1]

[1] University of Seville, Spain
almar@us.es

Abstract. We try to summarize some results concerning local transformations in some geometrical structures.

Probably the first time that a local transformation appeared in the context that we mean here was in the paper by Lawson [**1**]. The abstract of that work (more than forty years ago) says "This paper establishes the possibility of performing certain transformations of triangulations of finite planar point sets".

Since that paper, local transformations have been used extensively as a tool in order to obtain results on enumeration and optimality mainly, but also as a way to describe and so to know better some geometrical structures as matching, triangulations, trees, etc. The main reason to use this tool is (as Ferran Hurtado says in one of his papers [**2**]): "When the quality of a structure with respect to some criterion is considered, and no direct method for obtaining the optimal triangulation is known, it is natural to perform operations that allow local improvements".

One of the main contributors to this field has been (and will be in the future almost for sure) Ferran Hurtado (see, for example, [**3**]). He has several papers on this subject, and, in fact, he has studied local transformations (of flips) in all the structures we mentioned above.

Here we will try to summarize some of the results related with flips, and how this tool has been used to prove some other results.

References

[1] C. L. Lawson, Transforming triangulations, *Discrete Mathematics* vol. 3(4), 365–372 (1972).

[2] M. C. Hernando, F. Hurtado, A. Márquez, M. Mora, and M. Noy, Geometric tree graphs of points in convex position, *Discrete Appl. Math.*, vol. 93(1), 51–66 (1999).

[3] F. Hurtado, M. Noy and J. Urrutia, Flipping edges in triangulations, in: *Proceedings of the Twelfth Annual Symposium on Computational Geometry – SCG'96*, 214–223, 1996.

Recent results on plane geometric spanners

Prosenjit Bose[1]

[1] Carleton University
jit@scs.carleton.ca

A *geometric graph* G is a graph whose vertices are points in the plane and whose edges are line segments weighted by the Euclidean distance between their endpoints. In this setting, a *t-spanner* of G is a spanning subgraph G' with the property that, for every pair of vertices x, y, the shortest path from x to y in G' has weight at most $t \geq 1$ times the shortest path from x to y in G. The parameter t is commonly referred to as the *spanning ratio*, the *dilation* or the *stretch factor*. In addition to having bounded spanning ratio, it is desirable to build t-spanners that possess other properties, such as bounded degree, low weight, or fault-tolerance, to name a few. In this talk, we are particularly interested in planarity. There has been a flurry of activity in this area. We review various results on how to build plane geometric spanners.

[1]Research supported by NSERC.

Erdős-Szekeres-type theorems for convex bodies

János Pach[1]

[1] Rényi Institute, Budapest, and EPFL, Lausanne
pach@cims.nyu.edu

Erdős and Szekeres **[3, 4]** proved that any set of more than $\binom{2n-4}{n-2}$ points in general position in the plane contains n points which are in convex position, i.e., they form the vertex set of a convex n-gon. Bisztriczky and Fejes Tóth **[1, 2]** extended this result to families of convex sets, as follows.

Let $\mathcal{F} = \{B_1, \ldots, B_t\}$ be a family of compact convex sets in the plane in *general position,* i.e., no three of them have a common supporting line, and no two are tangent to each other. We say that $B_i \in \mathcal{F}$ is a *vertex* of $\mathcal{F}$ if B_i is not contained in the convex hull of the union of the others, i.e., if $\operatorname{bd}\operatorname{conv}(\cup\mathcal{F})$, the boundary of the convex hull of the union of all members of $\mathcal{F}$, contains a piece of the boundary of B_i. We say that $\mathcal{F}$ is in *convex position* if every member B_i $(i = 1, \ldots, t)$ of $\mathcal{F}$ is a vertex of $\mathcal{F}$. Evidently, any two members of $\mathcal{F}$ are in convex position.

Bisztriczky and Fejes Tóth proved that there exists a function $N(n)$ such that if $\mathcal{F}$ is a family of *pairwise disjoint* convex sets, $|\mathcal{F}| > N(n)$, and any *three* members of $\mathcal{F}$ are in convex position, then $\mathcal{F}$ has n members in convex position. We survey various generalizations and strengthenings of this result, based on joint work with Géza Tóth **[6, 7]** and with Jacob Fox, Benny Sudakov, and Andrew H. Suk **[5]**.

References

[1] T. Bisztriczky and G. Fejes Tóth, A generalization of the Erdős–Szekeres convex n-gon theorem, *Journal für die reine und angewandte Mathematik* **395** (1989), 167–170.

[2] T. Bisztriczky and G. Fejes Tóth, Convexly independent sets, *Combinatorica* **10** (1990), 195–202.

[3] P. Erdős and G. Szekeres, A combinatorial problem in geometry, *Compositio Mathematica* **2** (1935), 463–470.

[4] P. Erdős and G. Szekeres, On some extremum problems in elementary geometry, *Ann. Universitatis Scientiarum Budapestinensis, Eötvös, Sectio Mathematica* **III–IV** (1960–61), 53–62.

[5] J. Fox, J. Pach, B. Sudakov, and A. Suk, Erdős–Szekeres-type theorems for monotone paths and convex bodies, to appear.

[6] J. Pach and G. Tóth, A generalization of the Erdős–Szekeres theorem to disjoint convex sets, *Discrete and Computational Geometry* **19** (1998), 437–445.

[7] J. Pach and G. Tóth, Erdős–Szekeres-type theorems for segments and noncrossing convex sets, *Geometriae Dedicata* **81** (2000), 1–12.

Approximation algorithms for optimal covering tours in geometric settings

Joseph S. B. Mitchell[1]

[1] Stony Brook University, Stony Brook, NY 11794-3600, USA
Joseph.Mitchell@stonybrook.edu

The geometric optimal covering tour problem asks one to compute a shortest cycle that "covers" a geometric set S, such as a discrete set of points, a set of polygons, or a polygonal domain, where "coverage" generally means that the tour is required to visit each member of S or to come within a specified distance of each member of S. We survey recent approximation algorithm results on computing optimal cover tours in geometric environments.

A fundamental problem in geometric network optimization is the traveling salesman problem (TSP), which is a geometric covering tour problem on a discrete point set S. An extension of the TSP, the TSP with Neighborhoods (TSPN), requires that we find a tour of minimum length that visits a set S of *regions* (e.g., polygons). The TSPN shows up in many related geometric optimization problems, including the *watchman route problem* of computing a tour for a mobile guard to be able to see all of a given geometric domain. The TSPN also arises in the watchman route problem with limited visibility range, the *lawnmower* and *milling* problems, range scanning for model and map acquisition, and in various problems in sensor networks, including relay placement and mobile data mules for sensor network data gathering.

We survey the state of the art in approximation algorithms for geometric optimal covering tours, including recent results on watchman routes in polygonal domains, data gathering in sensor networks, and related geometric network optimization problems.

Linking geometric objects

Manuel Abellanas[1]

[1] Facultad de Informática, Universidad Politécnica de Madrid
manuel.abellanas@upm.es

Abstract. Let S be a set of geometric objects. In many computational geometric problems we are asked to link the elements of S, or part of them, by means of some geometric objects, usually satisfying a set of constraints. In this talk we overview several such problems. I have been lucky by working together with Ferran Hurtado in these and many other problems. I learn a lot from him and have a lot of fun when working together. This talk is a tribute to Ferran on his 60th Birthday.

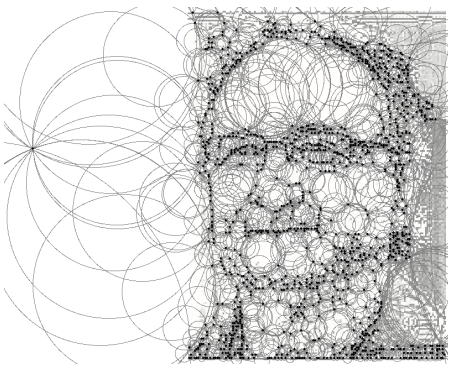


FIGURE 1. Ferran busy with arrangements of circles.

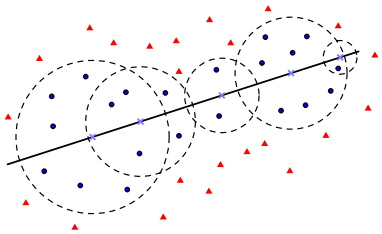
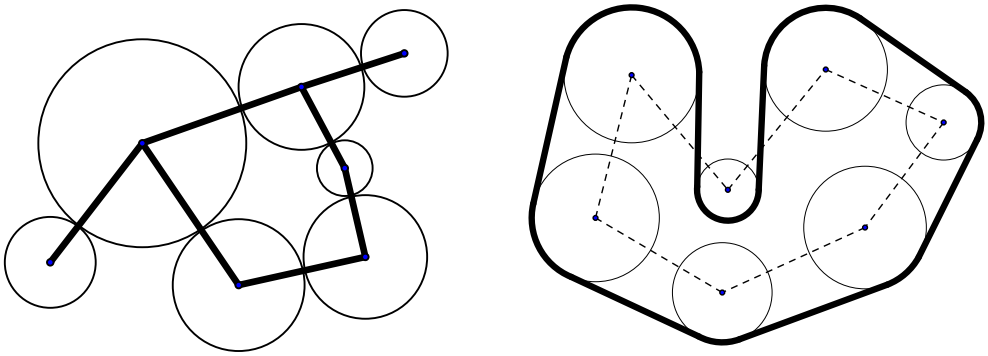


FIGURE 2. Friends and foes.

Problem 1 (Friends and foes [**2**])**.** We are given a set of red points and a set of blue points on the plane. Does there exist a set of circles, centered all of them on a common line, whose union contains the blue points and excludes the red ones? If the answer is no, which is the minimum number of red points that have to be removed in order to have a positive answer?

[1]Partially supported by Project MTM2008-05043.



(a) Problem 2: Contact graph realization. (b) Problem 3: Elastic belt puzzle.

FIGURE 3

Problem 2 (Gear systems [**3**])**.** Let S be a set of points on the plane and G a graph with vertex set S. Is it possible to connect the points of S by means of non-overlapping disks centered on them that touch each other following the pattern given by G? In other words, is it possible to realize G as a contact graph of disks centered at its vertices? How to obtain a realization in the positive case?

Problem 3 (Conveyer belt: a ten-year-old open puzzle [**1, 4**])**.** Let D be a set of disjoint disks on the plane. Suppose they are wheels and we want to link all of them by wrapping an elastic band around that tends to shrink as much as possible, thus obtaining a conveyer belt mechanical system (see Figure 3). Does there exist a configuration such that the band describes a simple Jordan curve? This is a generalization of the polygonization problem, in which the radii are all equal to zero.

References

[1] M. Abellanas, Conectando puntos: poligonizaciones y otros problemas relacionados, *Gaceta de la Real Sociedad Matemática Española* **11**(3), 2008, 543–558.

[2] M. Abellanas, J. M. Díaz-Báñez, P. Pérez-Lantero, I. Ventura, Locating a communication path in a competitive scenario, submitted.

[3] M. Abellanas, G. Hernández, C. Moreno, Conectando puntos con discos centrados, *V Jornadas de Matemática Discreta y Algorítmica*, Soria, 2006.

[4] E. D. Demaine, M. L. Demaine, B. Palop, Conveyer-belt alphabet, In *Pars Foundation, ed.*, Findings on Elasticity, Lars Müller Publishers, 2010.

Part II
Communications

Convexifying monotone polygons while maintaining internal visibility

O. Aichholzer[1], M. Cetina[2], R. Fabila-Monroy[3], J. Leaños[4], G. Salazar[2], J. Urrutia[5]

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

[2] Instituto de Física, Universidad Autónoma de San Luis Potosí, México
mcetina@ifisica.uaslp.mx, gsalazar@ifisica.uaslp.mx

[3] Departamento de Matemáticas, Cinvestav, México
ruyfabila@math.cinvestav.edu.mx

[4] Unidad Académica de Matemáticas, Universidad Autónoma de Zacatecas, México
jleanos@mate.reduaz.mx

[5] Instituto de Matemáticas, Universidad Nacional Autónoma de México
urrutia@matem.unam.mx

Abstract. Let P be a simple polygon on the plane. Two vertices of P are visible if the open line segment joining them is contained in the interior of P. In this paper we study the following questions posed in **[6, 7]**: (1) Is it true that every non-convex simple polygon has a vertex that can be continuously moved such that during the process no vertex-vertex visibility is lost and some vertex-vertex visibility is gained? (2) Can every simple polygon be convexified by continuously moving only one vertex at a time without losing any internal vertex-vertex visibility during the process?

We provide a counterexample to (1). We note that our counterexample uses a monotone polygon. We also show that question (2) has a positive answer for monotone polygons.

Introduction

Let P be a simple polygon with vertices $\{p_1, \ldots, p_n\}$. We say that two vertices of P are *P-visible* if the relative interior of the line segment joining them is contained in the interior of P. The *visibility graph* $VG(P)$ of P is the graph with vertex set $\{p_1, \ldots, p_n\}$ in which two vertices of P are adjacent if they are P-visible. A classical problem in computational geometry is that of convexifying simple polygons; that is, using a given fixed set of transformations that can be applied to the vertices and edges of P, try to transform P into a convex polygon in such a way that some properties of P are preserved. The first formulation of a problem of this kind was proposed by Erdős **[4]**, who proposed a strategy to convexify a non-convex polygon by using *flipturns*. Perhaps the most celebrated result in this area concerns the solution of the Carpenter's Rule conjecture **[3, 10]**; see also **[1, 2, 8, 9]**.

[1]Partially supported by the FWF under grant S9205-N12.
[2]Supported by CONACYT Grant 106432.
[5]Supported by CONACYT Grant 80268 and project MTM2006-03909 (Spain).

Our starting point is the following question posed by Satyan L. Devadoss in the Open Problem Session at CCCG 2008 [**6, 7**]:

Question. Given a simple polygon P and its visibility graph $VG(P)$, can the vertices of P (one at a time or simultaneously) be moved continuously along paths so that:

- the simplicity of the polygon P is maintained all the time, and
- the visibility graph of P never loses edges, only gains them.

In discussions after the workshop, the following specific questions were raised [**5**]:

(1) Has every non-convex simple polygon a vertex p that can be continuously moved so that $VG(P)$ gains at least one extra edge, and never loses any?
(2) Can every simple polygon be convexified by continuously moving several vertices in sequence, but only one at a time, such that $VG(P)$ never loses any edge?

We will prove that Question (2) has a positive answer for monotone polygons. On the other hand, we give an example that shows that the answer to Question (1) is negative, even for monotone polygons. For recent results on this topic, see also [**7**].

1 Polygons and visibility

Let P be a simple polygon as defined above. The interior of P is the area bounded by P and we consider this area as an open set, i.e., vertices and edges of P do not belong to the interior of P. Let u and v be the leftmost and rightmost vertices of P. There are two edge-disjoint paths contained in P joining u to v, which are called the *upper chain* of P, and the *lower chain* of P, respectively. If any vertical line intersects the interior of P in at most one connected component then P is x-monotone, where, for simplicity, we will simply use the term monotone. Finally, we suppose without loss of generality that no vertical line passes through two vertices of P.

A basic operation that we use in this paper is to move the vertices of P around the plane. Strictly speaking, the polygon P defined by its vertices changes. Nevertheless, abusing our terminology a bit, we will always refer to it as P. Moreover, we will restrict our point moves to those that do not destroy the simplicity of P.

We say that the two vertices u and v of P are *P-visible* if the relative interior of the line segment $\overline{uv}$ joining them is contained in the interior of P. We call $\{u, v\}$ a *visibility pair*. Note that, according to our definition, consecutive vertices of P are not visible. Let $\mathcal{N}(P)$ be the set of pairs of vertices of P that are not P-visible. As consecutive vertices of P are not P-visible, $|\mathcal{N}(P)| \geq n$. Note that if the vertices of P move, the set of visible pairs of P may change, and in turn the visibility graph $VG(P)$ may also change.

We say that a vertex move is *visibility-preserving* if the following holds: If p_j and p_k were P-visible, they remain P-visible while p_i moves. If in addition the number of edges of $VG(P)$ increases, then we call it a *visibility-increasing* vertex move.

Our main results are the following:

Theorem 1.1. *There are polygons that have no visibility-increasing vertex moves.*

Theorem 1.2. *Every monotone polygon can be convexified with a sequence of visibility-preserving moves.*

2 A counterexample to Question (1)

Consider the monotone polygon P shown in Figure 1. The coordinates of the vertices of P are $a = (-100, 0)$, $b = (-63, 40)$, $c = (-61, 40)$, $d = (-33, 2)$, and $e = (0, 45)$. The points $\{f, g, h, i\}$ are obtained from the points $\{a, b, c, d\}$ by reflecting them along the y-axis. Points b' to h' are obtained from the points b to h by a reflection along the x-axis.

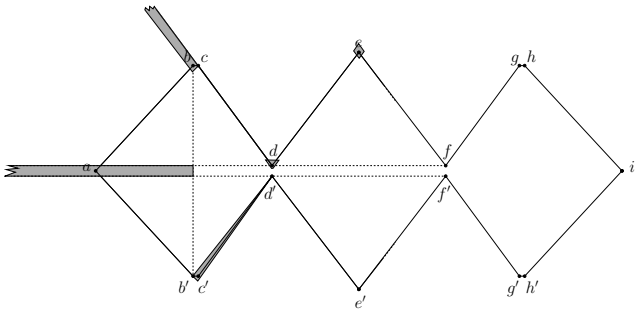


FIGURE 1. A monotone polygon without visibility-increasing vertex moves. Shaded areas indicate visibility-preserving regions. For point a, dashed lines indicate the boundary of its visibility-preserving region.

To show that P does not admit any visibility-increasing vertex move, it is sufficient to consider the vertices of P in the set $\{a, b, c', d, e\}$. The remaining cases follow by symmetry. For each of these vertices, we show in Figure 1 the open shaded region into which any of these points can be translated without losing any visibility pairs in P. It is now easy to see that there is no single vertex move that is visibility-increasing.

3 Visibility-preserving vertex moves

For a point $q \in \mathbb{R}^2$ and some $\delta > 0$, we denote by $B_\delta(q)$ the closed disk with radius δ with center at point q. Let $P = \{p_0, \ldots, p_{n-1}\}$ be a set of points in the plane in general position. We say that $\delta > 0$ is a *safe threshold of* P if there are no three elements p_i, p_j, and p_k of P such that $B_\delta(p_i)$, $B_\delta(p_j)$, and $B_\delta(p_k)$ are all intersected by a line ℓ. Equivalently, we can say that δ is a safe threshold of P if there are no three points $p_i, p_j, p_k \in P$ such that when we translate each of them to a point within δ distance of them, they become aligned.

It is not hard to see that every point set in the plane in general position has a safe threshold δ and that if a vertex move is not visibility-preserving, then at some point while moving the vertex it becomes collinear with two other vertices of P. However, the following lemma shows that collinearity is no problem for our approach. With $V^\circ(P)$ we denote the set of vertices interior to the convex hull of P.

Lemma 3.1. *Let P be a monotone polygon. Then there is a sequence of visibility-preserving vertex moves of some vertices of P such that at the end of the sequence, the vertices of P are in general position, P remains monotone, and $|V^\circ(P)| + |\mathcal{N}(P)|$ does not increase during the vertex movements.*

We are now ready to give a brief sketch of the proof of Theorem 1.2. By Lemma 3.1, we can assume that $V(P)$ is in general position. We proceed by induction on the sum

of the number of interior vertices plus the number of non-visible pairs. If the vertices of P are in convex position, there is nothing to prove. Observe that P is convex if $|V^\circ(P)| + |\mathcal{N}(P)| = n$. Suppose then that $|V^\circ(P)| + |\mathcal{N}(P)| > n$ and assume that the theorem holds for all polygons Q with $|V^\circ(Q)| + |\mathcal{N}(Q)| < |V^\circ(P)| + |\mathcal{N}(P)|$.

Since P is not convex, suppose without loss of generality that there are $k \geq 1$ interior vertices of P on its upper chain. Relabel them as $v_1, v_2, \ldots, v_k$, in increasing order with respect to their x-coordinate. Let $\delta > 0$ be a safe threshold for the *initial position* of $V(P)$. Our algorithm starts by executing the following basic procedure **BP**:

BP: *One at a time from left to right, move $v_1, v_2, \ldots, v_k$ upwards, by a distance δ.*

Once $v_1, v_2, \ldots, v_k$ have all been moved, we execute **BP** repeatedly (using always the same δ!) until one of the following occurs: (1) a vertex in $\{v_1, v_2, \ldots, v_k\}$ reaches the convex hull of P, (2) a new visible pair occurs, or (3) the visibility-preserving property is lost. If we stop because (1) or (2) occurs, then we are done, by our induction hypothesis. Using monotonicity of P we can show that (3) does not happen before a visibility-increasing event, which proves the theorem. Details are omitted in this extended abstract.

4 Conclusion

Several open questions arise from our work: How many vertex moves do we need to convexify a monotone polygon? Can this number be bounded by a polynomial? If we allow only vertical moves we can construct a polygon where the number of vertex moves is unbounded, but how about general moves? What happens if we allow more than one vertex to move at a time? We conclude with the following conjecture.

Conjecture 4.1. *Every simple polygon can be convexified by a sequence of visibility-preserving* 1*-vertex moves.*

Acknowledgments

We would like to thank Erik Demaine and Stefan Langerman for valuable discussions on the topic, as well as the anonymous referees for their comments.

References

[1] O. Aichholzer, C. Cortés, E. Demaine, V. Dujmović, J. Erickson, H. Meijer, M. Overmars, B. Palop, S. Ramaswami, and G. Toussaint. *Flipturning polygons.* Discrete and Computational Geometry, Vol. 28, 2002, pp. 231–253.

[2] T. C. Biedl, E. Demaine, S. Lazard, S. M. Robbins, and M. A. Soss. *Convexifying monotone polygons.* Lecture Notes in Computer Science 1741, pp. 415–424, 1999.

[3] R. Connelly, E. D. Demaine and G. Rote, *Straightening polygonal arcs and convexifying polygonal cycles.* Discrete Comput. Geom. **30** (2003), 205–239.

[4] P. Erdős, *Problem* 3763. Amer. Math. Monthly **42** (1935), p. 627.

[5] E. D. Demaine, S. Langerman. Personal communication, Montreal, 2008.

[6] E. D. Demaine, J. O'Rourke. *Open Problems from CCCG 2008.* Proceedings of the 21st Canadian Conference on Computational Geometry (CCCG 2009), pp. 75–78, 2009.

[7] S. L. Devadoss, R. Shah, X. Shao, and E. Winston. *Visibility graphs and deformations of associahedra.* arXiv:0903.2848, March 2009.

[8] B. Grünbaum. *How to convexify a polygon.* Geocombinatorics **5** (1995), pp. 24–30.

[9] B. de Sz.-Nagy. *Solution of problem* 3763. Amer. Math. Monthly **49** (1939), pp. 176–177.

[10] I. Streinu, *Pseudo-triangulations, rigidity and motion planning.* Discrete Comput. Geom. **34** (2005), 587–635.

Face guards for art galleries

Diane L. Souvaine[1], Raoul Veroy, Andrew Winslow[1]

Tufts University, Medford, MA, USA
dls@cs.tufts.edu, rveroy@cs.tufts.edu, awinslow@cs.tufts.edu

Abstract. A classic problem in computational geometry is the art gallery problem: given an enclosure, how should guards be placed to ensure that every location in the enclosure is seen by some guard. In this paper we consider guarding the interior of a simple polyhedron using *face guards*: guards who roam over an entire interior face of the polyhedron. Bounds for the number of face guards g that are necessary and sufficient to guard any polyhedron with f faces are given. We show that, for orthogonal polyhedra, $\lfloor f/7 \rfloor \leq g \leq \lfloor f/6 \rfloor$, while for general polyhedra $\lfloor f/5 - 2/5 \rfloor \leq g \leq \lfloor f/2 \rfloor$.

Introduction

In computational geometry, few problems are as recognizable as art gallery problems: given a region and a choice of how to place guards in the region, determine the locations of guards in order to see all locations in the region. In 2D, such problems have been considered for many years (see the surveys in [**2**], [**3**] and [**5**]); however, work in 3D is less extensive. Grünbaum and O'Rourke [**2**] and Szabó and Talata [**4**] consider guarding the exterior of a polyhedron with *vertex guards* stationed at vertices of the polyhedron, while Bose *et al.* [**1**] and Urrutia [**5**] consider guarding the interior of a polyhedron with *edge guards* free to walk along an entire edge.

In this work we consider an entirely new type of guard, called a *face guard*, who is free to move about a face (including its edges), and guards any location visible from *some* point on the face. We bound the number of face guards g sufficient to guard orthogonal and general simple polyhedra with f faces. In the orthogonal case we show that $\lfloor f/7 \rfloor \leq g \leq \lfloor f/6 \rfloor$, while in the general case $\lfloor f/5 - 2/5 \rfloor \leq g \leq \lfloor f/2 \rfloor$.

1 Definitions

In this paper we consider polyhedra with genus 0 (i.e., homeomorphic to a sphere) with faces that are *not* necessarily simply-connected (homeomorphic to discs). In contrast to some work on 3D art gallery problems, we consider guarding the interior of a polyhedron, not its exterior. A polyhedron is guarded by selecting interior faces to be face guards.

We say a point p in the interior of the polyhedron is *seen* by a face guard if the open line segment connecting p to some point on the closed face does not intersect the boundary of the polyhedron. Thus the region *guarded* by a face guard is the set of all points that are seen by a point on the face.

2 Orthogonal polyhedra

We start by considering the class of *orthogonal polyhedra*: polyhedra with every edge parallel to one of the three axes. By definition, each face of an orthogonal polyhedron has

[1]Research supported in part by NSF grants CCF-0830734 and CBET-0941538.

a normal vector parallel to one of the three axes. Thus, the interior faces of orthogonal polyhedra can be partitioned into six sets according to the directions of their normal vectors.

Lemma 2.1. *Let F be the set of all interior faces of an orthgonal polyhedron with normal vectors in the same direction. Then F is sufficient to guard the polyhedron.*

Proof. Without loss of generality, let F be the set of faces with normal vectors pointing in the positive x-direction. Let p be a point in the interior of the polyhedron. Consider extending a ray from p in the negative x-direction until it intersects a face f of the polyhedron. This face f must have a normal vector in the positive x-direction and sees p. So the set F guards the entire polyhedron. □

Lemma 2.2. *Let P be a polyhedron with f faces. Then $\lfloor f/6 \rfloor$ face guards are sufficient to guard any orthogonal polyhedron.*

Proof. The normal vectors of the interior faces of P partition these faces into six sets. By Lemma 2.1, each of these six sets are sufficient to guard P. By the pigeonhole principle, at least one of these sets has size at most $\lfloor f/6 \rfloor$. □

Lemma 2.3. *For all $f = 21k$ where k is a positive integer, there exist orthogonal polyhedra with f faces that require $3k$ face guards.*

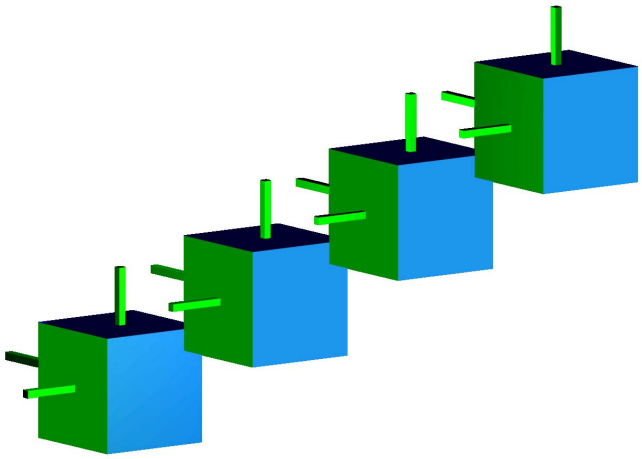


FIGURE 1. An orthogonal polyhedron with $21k$ faces requiring $3k$ guards for $k = 4$.

Proof. The proof is by explicit construction as seen in Figure 1. We use k large cubes, each with 3 narrow 'chimneys' attached to the cube's front, left, and top faces. The large cubes are attached to each other at their corners to form a long chain. Each cube and its 3 chimneys have a total of 21 faces, so the entire construction has $21k$ faces.

Consider using face guards to guard the interior of the polyhedron. We claim that a distinct face guard is needed for each chimney. This can be seen by considering the set of faces that see a point p deep in a chimney. Certainly each face of the chimney and the face of the cube containing can see p. Additionally, the face of the cube opposite of the face containing the chimney can also see p. Because the narrowness and length of the

chimney, no other faces can see p, including those from adjacent cubes. Intuitively, one can think of placing a light at p, and the light leaving the chimney as a focused beam which strikes only the center of the opposite face of the cube. Applying this analysis to all three chimneys in a cube, we see that a distinct face guard is needed for each chimney, thus guarding the entire construction requires $3k$ face guards. □

Theorem 2.4. *Let g be the minimum number of face guards sufficient to guard any orthogonal polyhedron with f faces. Then $\lfloor f/7 \rfloor \leq g \leq \lfloor f/6 \rfloor$.*

Proof. Combining the results from Lemmas 3.1 and 3.2 gives the inequalities. □

3 General polyhedra

In this section we consider guarding general simple polyhedra. In comparison to orthogonal polyhedra, we find that the necessary and sufficient numbers of guards are both increased.

Lemma 3.1. *Let P be a polyhedron with f faces. Then $\lfloor f/2 \rfloor$ face guards are sufficient to guard P.*

Proof. This proof is similar to the proof of Lemma 2.2. Consider the dot product of the normal vectors of the interior faces of P with a vector in the positive x-direction. Each dot product is either negative or non-negative. Partition the faces into two sets according to the values of their dot products. One of these sets must have size at most $\lfloor f/2 \rfloor$. Without loss of generality, suppose that the set of faces with non-negative dot products has size at most $\lfloor f/2 \rfloor$, and call this set F.

Let p be a point in the interior of P. Consider extending a ray from p in negative x-direction. The first face intersected must be in F, and so F guards p. So F is sufficient to guard P. □

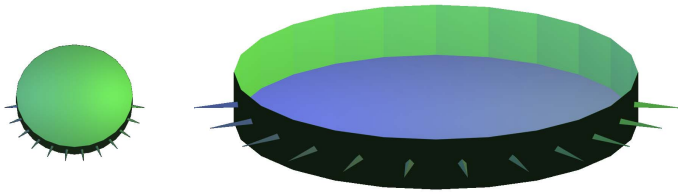


FIGURE 2. A polyhedron with $5k + 2$ faces requiring k face guards for $k = 12$. At left is the complete polyhedron, while at right is a larger version of the polyhedron with the top face removed.

Lemma 3.2. *For all $f = 5k + 2$ where k is a positive integer, there exist polyhedra with f faces that require $\frac{1}{5}f - \frac{2}{5}$ face guards.*

Proof. The proof is by explicit construction (see Figure 2). The construction is a large disc-shaped polyhedron consisting of two large regular faces with $2k$ edges each, and $2k$ small faces connecting them. Half (k) of the small faces have a narrow *spike* consisting of three faces extending out of the polyhedron meeting at a single point. The tip of the spike can only see a single small face on the opposite side of the polyhedron. The face it

sees does not have a spike leaving it. The construction has a total of $2k+2+3k = 5k+2$ faces.

We claim that the vertex at the pointy end of each spike can only been seen by one of 5 faces: the three faces creating the spike, the face the spike leaves, and the face on the far side of the polyhedron that the pointy end sees. Looking at the sets of faces that see the pointy end of each spike, we notice they are disjoint. So a distinct face guard is needed for the pointy end of each spike. There are k such spikes, so k face guards are needed. □

Theorem 3.3. *Let g be the number of face guards sufficient to guard any polyhedron with f faces. Then $\lfloor \frac{1}{5}f - \frac{2}{5} \rfloor \leq g \leq \lfloor f/2 \rfloor$.*

Proof. Combining Lemmas 3.1 and 3.2 gives the inequalities. □

4 Conclusion and open problems

In this work we have introduced the notion of face guards for 3D art gallery problems. This new type of guard is permitted to walk about freely on a closed interior face, and guards any point in the polyhedron seen from some location on this face. We give bounds on the number of face guards needed to guard the interior of both orthogonal and general simple polyhedra.

This new type of guard for polyhedra suggests an interesting set of open problems. Such problems include the complexity of minimizing the number of face guards, necessary and sufficient numbers of guards for tetrahedralizable, surface triangulated, and non-zero genus polyhedra. We also support the investigation of *open face guards*, in which the boundary of the face is omitted.

Acknowledgements

We wish to thank Megan Strait for her encouragement, and anonymous reviewers for their insightful suggestions and comments.

References

[1] P. Bose, T. Shermer, G. Toussaint, B. Zhu, Guarding polyhedral terrains, *Computational Geometry* **7** (1997), 173–185.

[2] J. O'Rourke, *Art Gallery Theorems and Algorithms*, The Intl. Series of Monographs on Comp. Sci., Oxford University Press, New York, 1987.

[3] T. C. Shermer, Recent results in art galleries, *Proc. of the IEEE* **80** (1992), 1384–1399.

[4] L. Szabo, Recent results on illumination problems, *Bolyai Society Mathematical Studies* **6** (1997), 207–221.

[5] J. Urrutia, Art gallery and illumination problems, in: *J.-R. Sack, J. Urrutia (eds.), Handbook on Computational Geometry*, Elsevier, Amsterdam, 2000, 973–1027.

Dynamic circle separability between convex polygons

Luis Barba[1], **Jorge Urrutia**[2]

[1] Posgrado en Ciencia e Ingeniería de la Computación, Universidad Nacional Autónoma de México
l.barba@uxmcc2.iimas.unam.mx

[2] Instituto de Matemáticas, Universidad Nacional Autónoma de México
urrutia@matem.unam.mx

Abstract. Let P, Q be two polygons of n and m vertices, respectively. A circle containing P and whose interior does not intersect Q is called a *separating circle*. We propose an algorithm for finding the minimum separating circle between a fixed convex polygon P and query convex polygon Q. The polygons P and Q are given as ordered lists of vertices (sorted according to their order of appearance along the convex hulls of P and Q respectively). We perform a linear time preprocessing on the number of vertices of P; the query time complexity is $O(\log n \log m)$.

Introduction

Kim and Anderson [**1**] presented a quadratic algorithm for solving the circular separability problem between any two finite planar sets. Bhattacharya [**2**] improved the running time to $O(n \log n)$. Finally O'Rourke, Kosaraju and Megiddo [**3**] found an optimal linear time algorithm to solve this problem. In this paper we study a new version of the problem. Let P be a fixed convex polygon with n vertices. We propose an algorithm for solving the circular separability problem between P and any query convex polygon Q with m vertices, both given as an ordered list of their elements. Our algorithm uses a linear time preprocessing on P, and has $O(\log n \log m)$ query time complexity.

1 Circular separability

Suppose for ease of description that the vertices of P and Q are in general position, and that P has no four co-circular vertices. Let C_P be the minimum enclosing circle of P and let c_P be its center. It is known that c_P can be found in $O(n)$ time [**4**]. Note that c_P is a point on an edge of the farthest-point Voronoi diagram of the vertices of P. Clearly if the interiors of Q and P are not disjoint, our problem has no solution, hence we will suppose that $d(P, Q) \geq 0$. It is also clear that if Q and C_P have disjoint interiors, then C_P is trivially the minimum separating circle.

1.1 Preprocessing

We first calculate the farthest-point Voronoi diagram of the vertices of P in linear time [**5**]. It can be seen as a tree rooted in c_P and created by adding leaves on every unbounded edge; we will denote this tree as $\mathcal{V}(P)$. For each vertex p of P, let $R(p)$ be the farthest-point Voronoi region associated to p, and assume that p has a pointer to $R(p)$. Let x be a point on an edge of $\mathcal{V}(P)$, and let T_x denote the path contained in $\mathcal{V}(P)$ joining c_P to x.

We will use the data structure on $\mathcal{V}(P)$ proposed by Roy, Karmakar, Das and Nandy in [**6**], which can be constructed in linear time and uses linear space. Given a vertex v in the tree $\mathcal{V}(P)$, this data structure allows us to do a binary search on the vertices of $\mathcal{V}(P)$ lying on T_v.

1.2 The minimum separating circle

We will call every circle containing P and whose interior does not intersect Q a *separating circle*. Let c' be the center of the minimum separating circle. In this section we will find c' starting from the center of an arbitrary separating circle.

Given $x \in \mathbb{R}^2$, let $C(x)$ be the minimum enclosing circle of P with center on x, and let $\rho(x)$ be the radius of $C(x)$. The following is a well-known result for the farthest-point Voronoi diagram.

Proposition 1.1. *Let x be a point on $\mathcal{V}(P)$. Then ρ is a monotonically increasing function along the path T_x starting at c_P.*

We now address some properties of separating circles, some of which are given without proof.

Observation 1.2. *The minimum separating circle has its center on $\mathcal{V}(P)$.*

Observation 1.3. *Let $x, y \in \mathbb{R}^2$. For every $z \in [x, y]$, we have $C(z) \subseteq C(x) \cup C(y)$.*

The previous observation implies that the minimum separating circle is unique.

Proposition 1.4. *Let x, y be two points on $\mathcal{V}(P)$ such that $C(x), C(y)$ are separating circles and x, y belong to the boundary of the Voronoi region $R(p)$. If z is the lowest common ancestor of x and y in $\mathcal{V}(P)$, then $C(z)$ is a separating circle; moreover, we have $\rho(z) \leq \min\{\rho(x), \rho(y)\}$.*

Proof. Suppose that $y \notin T_x$ and $x \notin T_y$; otherwise the result follows trivially. Assume then that the paths connecting x and y to z have disjoint relative interiors. Let $\ell_{z,p}$ be the straight line through z and p; this line leaves x and y in different semiplanes. Let z' be the intersection between $\ell_{z,p}$ and $[x, y]$. By Observation 1.3 we know that $C(z') \subseteq C(x) \cup C(y)$. Since z', z, p are co-linear, we have $C(z) \subseteq C(z')$ and thus $\rho(z) < \rho(z')$; see Figure 1(a). Finally, by transitivity we have that $C(z) \subset C(x) \cup C(y)$, which implies that $C(z)$ is a separating circle. Using Proposition 1.1, we conclude that $\rho(z) \leq \min\{\rho(x), \rho(y)\}$. □

Now we generalize the previous result.

Lemma 1.5. *Let x, y be two points on $\mathcal{V}(P)$ such that $C(x), C(y)$ are separating circles. If z is the lowest common ancestor of x and y in the rooted tree $\mathcal{V}(P)$, then $C(z)$ is a separating circle; moreover, $\rho(z) \leq \min\{\rho(x), \rho(y)\}$.*

Proof. Proceeding by contradiction, suppose that $C(z)$ is not a separating circle. Let w_x be a point on T_x such that $\rho(w_x) = \min\{\rho(w) : w \in T_x \text{ and } C(w) \text{ is a separating circle}\}$; thus $w_x \neq z$. Consider the intersections of the segment $[w_x, y]$ with $\mathcal{V}(P)$ and suppose that the intersection points are $w_x = x_0, x_1, \ldots, x_k = y$ in this order. Let z' be the lowest common ancestor of w_x and x_1 in $\mathcal{V}(P)$. It is clear that w_x and x_1 belong to the same Voronoi region. Thus, by Proposition 1.4, $C(z')$ is a separating circle. Note that z' belongs to T_x, which is a contradiction with the definition of w_x. Our result follows. □

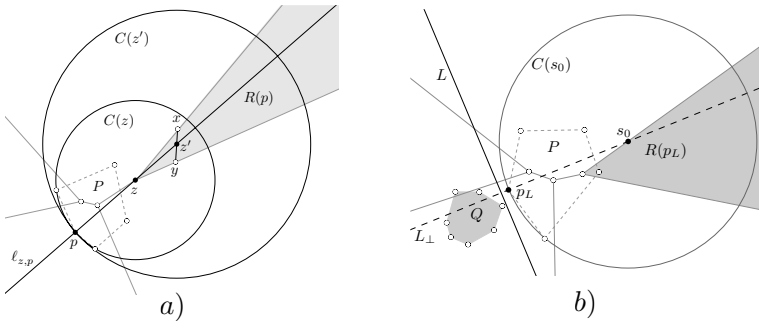


FIGURE 1. (*a*) Proof of Proposition 1.4. (*b*) The construction of s_0.

Theorem 1.6. *Let s be a point on an edge of $\mathcal{V}(P)$ such that $C(s)$ is a separating circle. Then c' belongs to T_s.*

Proof. Let w be a point on an edge of T_s such that

$$\rho(w) = \min\{\rho(z) \mid z \in T_s \text{ and } C(z) \text{ is a separating circle}\}.$$

Suppose that $w \neq c'$; thus $c' \notin T_s$. Therefore, by Lemma 1.5, if z is the lowest common ancestor of c' and w, then $C(z)$ is a separating circle with $\rho(z) \leq \rho(c')$. Also, since $c' \notin T_w \subseteq T_s$, the inequality is strict, which is a contradiction; our result follows. □

2 The algorithm

In this section, we present an algorithm to find c'. Our algorithm first finds a separating circle with center s_0 on an edge of $\mathcal{V}(P)$. Then we search for c' using a binary search on T_{s_0}.

We first construct a straight line L separating P and Q in logarithmic time [**7**]. Let us assume that p_L is the unique point in P closest to L. Otherwise, rotate L slightly, keeping P and Q separated by L. Let $L_\perp$ be the perpendicular to L that contains p_L and let s_0 be the intersection of $L_\perp$ with the boundary of $R(p_L)$. Note that $d(s_0, p_L)$ defines the radius of $C(s_0)$, therefore $C(s_0)$ is a separating circle; see Figure 1(b). Also, by construction s_0 is on an edge of $\mathcal{V}(P)$. It is clear that we can find s_0 in $O(\log n + \log m)$ time. Suppose that s_0 is on the edge xy of $\mathcal{V}(P)$, and let $T_x = (c_P = u_0, u_1, \ldots, u_{r-1} = y, u_r = x)$. It follows from Theorem 1.6 that c' is on an edge of T_x.

Using the data structure proposed by Roy, Karmakar, Das and Nandy [**6**], we perform a binary search for c' on the vertices of T_x as follows. Initially, let $j = 0$, and $k = r$. Let u_i be the mid-vertex on the path of T_x between u_j and u_k. First compute $d(u_i, Q)$ in $O(\log m)$ time [**7**]. Now, in constant time, calculate $\rho(u_i)$. If $d(u_i, Q) = \rho(u_i)$, then $u_i = c'$ and the algorithm ends. If $d(u_i, Q) < \rho(u_i)$, then we search for c' between u_i and u_k; if $d(u_i, Q) > \rho(u_i)$, then we search for c' between u_j and u_i.

Two possibilities arise. If c' is a vertex on $\mathcal{V}(P)$, then we will find it in $O(\log n)$ steps. Otherwise, if c' is an interior point of an edge $S = [u, v]$ of $\mathcal{V}(P)$, our algorithm will return S such that $c' \in S$. Since each step of the binary search requires $O(\log m)$ time, the complexity of the previous search is $O(\log n \log m)$.

Suppose that S is contained in the bisector of two vertices p_0, p_1 of P, and let Q_S be the set of points on the boundary of Q visible from every point in S. It can be computed in $O(\log m)$ time. Let $q_{c'}$ be the point of intersection of $C(c')$ and Q. Clearly $q_{c'}$ belongs

to Q_S; see Figure 2(a). Given three points $p, q, r \in \mathbb{R}^2$, let $C(pqr)$ be the circumcircle of the triangle $\triangle(pqr)$. For $x \in Q_S$, let $F(x)$ be the radius of the circle $C(p_0xp_1)$. It is easy to see that $F(x)$ is unimodal on Q_S and attains its maximal at $q_{c'}$; see Figure 2(b).

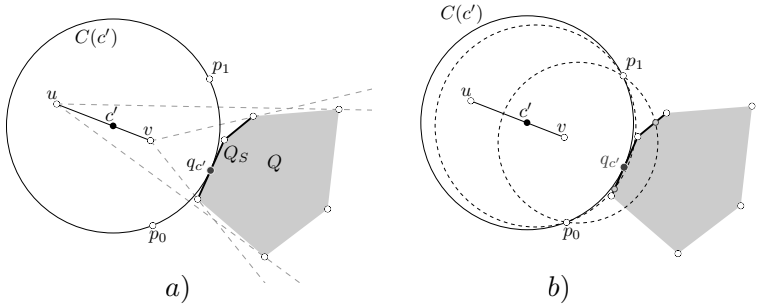


FIGURE 2. (*a*) The construction of Q_S. (*b*) $q_{c'}$ is maximal under F.

Let $Q_S^* = \{q_0, q_1, \ldots, q_r\}$ be the set of vertices of Q lying on Q_S. We can perform a binary search for $q_{c'}$ on the sorted list Q_S^* as follows. At each step we take the midpoint q^* of the current search list (initially Q_S^*), and compute the value of $F(q^*)$ in constant time. Take two points on each side of q^* at epsilon distance on the boundary of Q. If q^* is a local maximum of F, then the algorithm returns $q_{c'} = q^*$. Otherwise, determine if $q_{c'}$ lies to the left or to the right of q^*. Eliminate half of the list according to the position of $q_{c'}$ and repeat recursively. Our algorithm returns either the value of $q_{c'}$ if it is a vertex of Q, or a segment $H = (q_i, q_{i+1})$ of Q_S such that $q_{c'}$ belongs to H. In the first case, we are done, since c' can be determined in constant time given the position of $q_{c'}$. In the second case, the problem is reduced to that of finding a point $c' \in S$ such that $d(c', p_0) = d(c', H)$. This case can be solved with a quadratic equation in constant time.

Since each step of the binary search requires constant time, the algorithm finds the point $q_{c'}$ in $O(\log m)$ time, giving an overall complexity of $O(\log n \log m)$ for the algorithm.

References

[1] C. E. Kim and T. A. Anderson, Digital disks and a digital compactness measure, *STOC'84: Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, 1984, 117–124.

[2] B. K. Bhattacharya, Circular separability of planar point sets, *Computational Morphology* (1988), 25–39.

[3] J. O'Rourke, S. R. Kosaraju and N. Megiddo, Computing circular separability, *Discrete and Computational Geometry* **1** (1986), 105–113.

[4] N. Megiddo, Linear-time algorithms for linear programming in $\mathbb{R}^3$ and related problems, *SIAM Journal on Computing* **12** (1983), 759–776.

[5] 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 and Computational Geometry* **4** (1989), 591–604.

[6] S. Roy, A. Karmakar, S. Das and S. C. Nandy, Constrained minimum enclosing circle with center on a query line segment, *Computational Geometry* **42** (2009), 632–638.

[7] H. Edelsbrunner, Computing the extreme distances between two convex polygons, *Journal of Algorithms* **6** (1985), 213–224.

The chromatic number of the convex segment disjointness graph

Ruy Fabila-Monroy[1], **David R. Wood**[2]

[1] Departamento de Matemáticas, Centro de Investigación y Estudios Avanzados del Instituto Politécnico Nacional, México, D.F.
ruyfabila@math.cinvestav.edu.mx

[2] Department of Mathematics and Statistics, The University of Melbourne, Melbourne, Australia
woodd@unimelb.edu.au

Abstract. Let P be a set of n points in general and convex position in the plane. Let D_n be the graph whose vertex set is the set of all line segments with endpoints in P, where disjoint segments are adjacent. The chromatic number of this graph was first studied by Araujo et al. [*CGTA*, 2005]. The previous best bounds are $\frac{3n}{4} \leq \chi(D_n) < n - \sqrt{\frac{n}{2}}$ (ignoring lower order terms). In this paper we improve the lower bound to $\chi(D_n) \geq n - \sqrt{2n}$, to conclude a near-tight bound on $\chi(D_n)$.

1 Introduction

Throughout this paper, P is a set of $n > 3$ points in general and convex position in the plane. The *convex segment disjointness graph*, denoted by D_n, is the graph whose vertex set is the set of all line segments with endpoints in P, where two vertices are adjacent if the corresponding segments are disjoint. Obviously D_n does not depend on the choice of P. This graph and other related graphs were introduced by Araujo, Dumitrescu, Hurtado, Noy and Urrutia [**1**], who proved the following bounds on the chromatic number of D_n:

$$2\left\lfloor \tfrac{1}{3}(n+1)\right\rfloor - 1 \leq \chi(D_n) < n - \tfrac{1}{2}\left\lfloor \log n\right\rfloor .$$

Both bounds were improved by Dujmović and Wood [**5**] to

$$\tfrac{3}{4}(n-2) \leq \chi(D_n) < n - \sqrt{\tfrac{1}{2}n} - \tfrac{1}{2}(\ln n) + 4.$$

In this paper we improve the lower bound to conclude near-tight bounds on $\chi(D_n)$.

Theorem 1.1.

$$n - \sqrt{2n + \tfrac{1}{4}} + \tfrac{1}{2} \leq \chi(D_n) < n - \sqrt{\tfrac{1}{2}n} - \tfrac{1}{2}(\ln n) + 4.$$

The proof of Theorem 1.1 is based on the observation that each colour class in a colouring of D_n is a convex thrackle. We then prove that two maximal convex thrackles must share an edge in common. From this we prove a tight upper bound on the number of edges in the union of k maximal convex thrackles. Theorem 1.1 quickly follows.

2 Convex thrackles

A *convex thrackle* on P is a geometric graph with vertex set P such that every pair of edges intersect; that is, they have a common endpoint or they cross. Observe that a geometric graph H on P is a convex thrackle if and only if $E(H)$ forms an independent set in D_n. A convex thrackle is *maximal* if it is edge-maximal. As illustrated in Figure 1(a),

it is well known and easily proved that every maximal convex thrackle T consists of an odd cycle $C(T)$ together with some degree 1 vertices adjacent to vertices of $C(T)$; see **[2, 3, 4, 5, 6, 7, 8, 9]**. In particular, T has n edges. For each vertex v in $C(T)$, let $W_T(v)$ be the convex wedge with apex v, such that the boundary rays of $W_T(v)$ contain the neighbours of v in $C(T)$. Every degree-1 vertex u of T lies in a unique wedge and the apex of this wedge is the only neighbour of u in T.

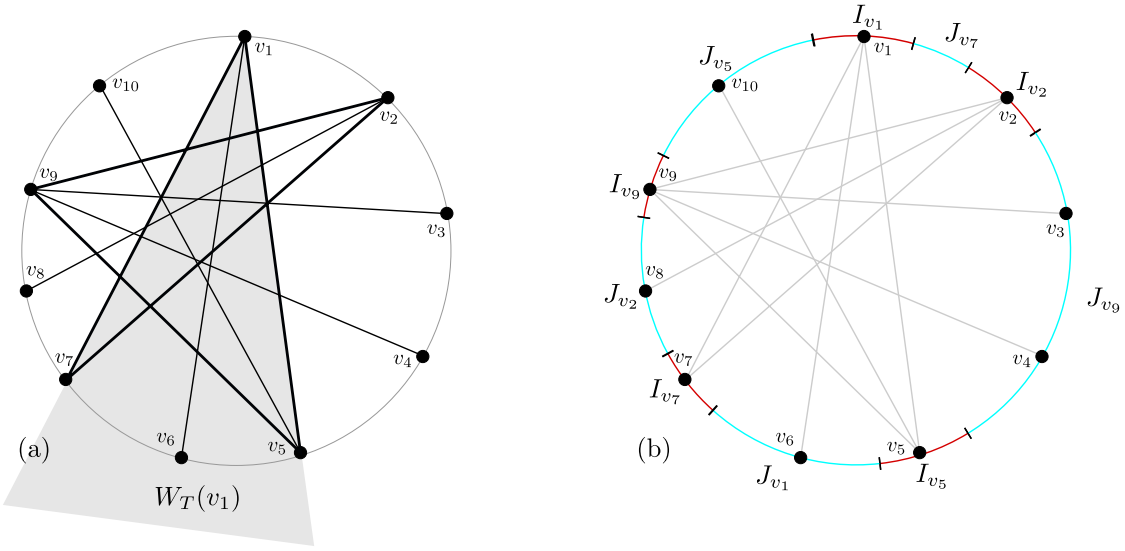


FIGURE 1. (a) Maximal convex thrackle. (b) The interval pairs (I_u, J_u).

3 Convex thrackles and free $\mathbb{Z}_2$-actions of S^1

A $\mathbb{Z}_2$-*action* on the unit circle S^1 is a homeomorphism $f\colon S^1 \to S^1$ such that $f(f(x)) = x$ for all $x \in S^1$. We say that f is *free* if $f(x) \neq x$ for all $x \in S^1$.

Lemma 3.1. *If f and g are free $\mathbb{Z}_2$-actions of S^1, then $f(x) = g(x)$ for some $x \in S^1$.*

Proof. For points $x, y \in S^1$, let $\overrightarrow{xy}$ be the clockwise arc from x to y in S^1. Let $x_0 \in S^1$. If $f(x_0) = g(x_0)$ then we are done. Now assume that $f(x_0) \neq g(x_0)$. Without loss of generality, $x_0, g(x_0), f(x_0)$ appear in this clockwise order around S^1. Parametrise $\overrightarrow{x_0 g(x_0)}$ with a continuous injective function $p\colon [0,1] \to \overrightarrow{x_0 g(x_0)}$ such that $p(0) = x_0$ and $p(1) = g(x_0)$. Assume that $g(p(t)) \neq f(p(t))$ for all $t \in [0,1]$; otherwise we are done. Since g is free, $p(t) \neq g(p(t))$ for all $t \in [0,1]$. Thus $g(p([0,1])) = \overrightarrow{g(p(0))g(p(1))} = \overrightarrow{g(x_0)x_0}$. Also $f(p([0,1])) = \overrightarrow{f(x_0)f(p(1))}$, as otherwise $g(p(t)) = f(p(t))$ for some $t \in [0,1]$. This implies that $p(t), g(p(t)), f(p(t))$ appear in this clockwise order around S^1. In particular, with $t = 1$, we have $f(p(1)) \in \overrightarrow{x_0 g(x_0)}$. Thus $x_0 \in \overrightarrow{f(x_0)f(p(1))}$. Hence $x_0 = f(p(t))$ for some $t \in [0,1]$. Since f is a $\mathbb{Z}_2$-action, $f(x_0) = p(t)$. This is a contradiction, since $p(t) \in \overrightarrow{x_0 g(x_0)}$ but $f(x_0) \notin \overrightarrow{x_0 g(x_0)}$. □

Assume that P lies on S^1. Let T be a maximal convex thrackle on P. As illustrated in Figure 1(b), for each vertex u in $C(T)$, let (I_u, J_u) be a pair of closed intervals of S^1 defined as follows. Interval I_u contains u and bounded by the points of S^1 that are

1/3 of the way towards the first points of P in the clockwise and anticlockwise direction from u. Let v and w be the neighbours of u in $C(T)$, so that v is before w in the clockwise direction from u. Let p be the endpoint of I_v in the clockwise direction from v. Let q be the endpoint of I_w in the anticlockwise direction from w. Then J_u is the interval bounded by p and q and not containing u. Define $f_T\colon S^1 \to S^1$ as follows. For each $v \in C(T)$, map the anticlockwise endpoint of I_v to the anticlockwise endpoint of J_v, map the clockwise endpoint of I_v to the clockwise endpoint of J_v, and extend f_T linearly for the interior points of I_v and J_v, such that $f_T(I_v) = J_v$ and $f_T(J_v) = I_v$. Since the intervals I_v and J_v are disjoint, f_T is a free $\mathbb{Z}_2$-action of S^1.

Lemma 3.2. *Let T_1 and T_2 be maximal convex thrackles on P with $C(T_1) \cap C(T_2) = \emptyset$. Then there is an edge in $T_1 \cap T_2$ with one endpoint in $C(T_1)$ and one endpoint in $C(T_2)$.*

Topological proof. By Lemma 3.1, there exists $x \in S^1$ such that $f_{T_1}(x) = y = f_{T_2}(x)$. Let $u \in C(T_1)$ and $v \in C(T_2)$ be so that $x \in I_u \cup J_u$ and $x \in I_v \cup J_v$, where (I_u, J_u) and (I_v, J_v) are defined with respect to T_1 and T_2, respectively. Since $C(T_1) \cap C(T_2) = \emptyset$, we have $u \neq v$ and $I_u \cap I_v = \emptyset$. Thus $x \notin I_u \cap I_v$. If $x \in J_u \cap J_v$, then $y \in I_u \cap I_v$, implying $u = v$. Thus $x \notin J_u \cap J_v$. Hence $x \in (I_u \cap J_v) \cup (J_u \cap I_v)$. Without loss of generality, $x \in I_u \cap J_v$. Thus $y \in J_u \cap I_v$. If $I_u \cap J_v = \{x\}$, then x is an endpoint of both I_u and J_v, implying $u \in C(T_2)$, which is a contradiction. Thus $I_u \cap J_v$ contains points other than x. It follows that $I_u \subset J_v$ and $I_v \subset J_u$. Therefore the edge uv is in both T_1 and T_2. Moreover, one endpoint of uv is in $C(T_1)$ and one endpoint is in $C(T_2)$. □

Combinatorial proof. Let H be the directed multigraph with vertex set $C(T_1) \cup C(T_2)$, where there is a *blue* arc uv in H if u is in $W_{T_1}(v)$ and there is a *red* arc uv in H if u is in $W_{T_2}(v)$. Since $C(T_1) \cap C(T_2) = \emptyset$, every vertex of H has outdegree 1. Therefore $|E(H)| = |V(H)|$ and there is a cycle Γ in the undirected multigraph underlying H. In fact, since every vertex has outdegree 1, Γ is a directed cycle. By construction, vertices in H are not incident to an incoming and an outgoing edge of the same color. Thus Γ alternates between blue and red arcs. The red edges of Γ form a matching as well as the blue edges, both of which are thrackles. However, there is only one matching thrackle on a set of points in convex position. Therefore Γ is a 2-cycle and the result follows. □

4 Main results

Theorem 4.1. *For every set P of n points in convex and general position, the union of k maximal convex thrackles on P has at most $kn - \binom{k}{2}$ edges.*

Proof. For a set $\mathcal{T}$ of k maximal convex thrackles on P, define

$$r(\mathcal{T}) = \left|\{(v, T_i, T_j) \,:\, v \in C(T_i) \cap C(T_j),\ T_i, T_j \in \mathcal{T} \text{ and } T_i \neq T_j\}\right|.$$

The proof proceeds by induction on $r(\mathcal{T})$.

Suppose that $r(\mathcal{T}) = 0$. Thus $C(T_i) \cap C(T_j) = \emptyset$ for all distinct $T_i, T_j \in \mathcal{T}$. By Lemma 3.2, T_i and T_j have an edge in common, with one endpoint in $C(T_i)$ and one endpoint in $C(T_j)$. Hence distinct pairs of thrackles have distinct edges in common. Since every maximal convex thrackle has n edges and we overcount at least one edge for every pair, the total number of edges is at most $kn - \binom{k}{2}$.

Now assume that $r(\mathcal{T}) > 0$. Thus there is a vertex v and a pair of thrackles T_i and T_j such that $v \in C(T_i) \cap C(T_j)$. As illustrated in Figure 2, replace v by two consecutive vertices v' and v'' on P, where v' replaces v in every thrackle except T_j, and v'' replaces

v in T_j. Add one edge to each thrackle so that it is maximal. Let $\mathcal{T}'$ be the resulting set of thrackles. Observe that $r(\mathcal{T}') = r(\mathcal{T}) - 1$ and the number of edges in $\mathcal{T}'$ equals the number of edges in $\mathcal{T}$ plus k. By induction, $\mathcal{T}'$ has at most $k(n+1) - \binom{k}{2}$ edges, implying that $\mathcal{T}$ has at most $kn - \binom{k}{2}$ edges. □

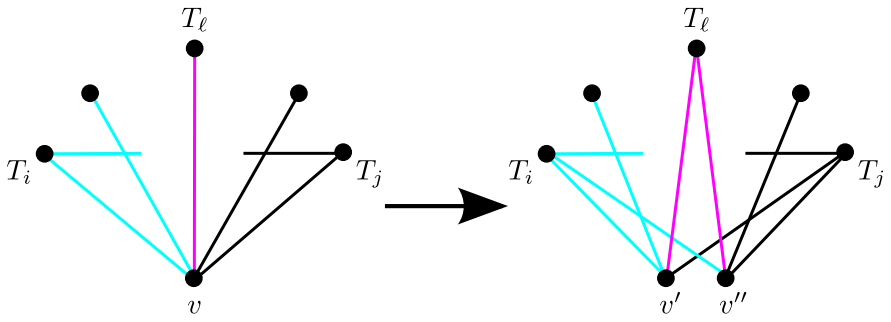


FIGURE 2. Construction in the proof of Theorem 4.1.

We now show that Theorem 4.1 is best possible for all $n \geq 2k$. Let S be a set of k vertices in P with no two consecutive vertices in S. If $v \in S$ and x, v, y are consecutive in this order in P, then $T_v = \{vw : w \in P \setminus \{v\}\} \cup \{xy\}$ is a maximal convex thrackle, and $\{T_v : v \in S\}$ has exactly $kn - \binom{k}{2}$ edges in total.

Proof of Theorem 1.1. If $\chi(D_n) = k$, then there are k convex thrackles whose union is the complete geometric graph on P. Possibly add edges to obtain k maximal convex thrackles with $\binom{n}{2}$ edges in total. By Theorem 4.1, $\binom{n}{2} \leq kn - \binom{k}{2}$. The quadratic formula implies the result. □

Acknowledgements. R. F.-M. is supported by an Endeavour Fellowship from the Department of Education, Employment and Workplace Relations of the Australian Government. D. W. is supported by a QEII Fellowship from the Australian Research Council.

References

[1] Gabriela Araujo, Adrian Dumitrescu, Ferran Hurtado, Marc Noy, and Jorge Urrutia. On the chromatic number of some geometric type Kneser graphs. *Comput. Geom. Theory Appl.*, 32(1):59–69, 2005.

[2] Grant Cairns and Yury Nikolayevsky. Bounds for generalized thrackles. *Discrete Comput. Geom.*, 23(2):191–206, 2000.

[3] Grant Cairns and Yury Nikolayevsky. Generalized thrackle drawings of non-bipartite graphs. *Discrete Comput. Geom.*, 41(1):119–134, 2009.

[4] Grant Cairns and Yury Nikolayevsky. Outerplanar thrackles. *Graphs and Combinatorics*, to appear.

[5] Vida Dujmović and David R. Wood. Thickness and antithickness, 2010, in preparation.

[6] W. Fenchel and J. Sutherland. Lösung der Aufgabe 167. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 45:33–35, 1935.

[7] H. Hopf and E. Pammwitz. Aufgabe no. 167. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 43, 1934.

[8] László Lovász, János Pach, and Mario Szegedy. On Conway's thrackle conjecture. *Discrete Comput. Geom.*, 18(4):369–376, 1997.

[9] Douglas R. Woodall. Thrackles and deadlock. In *Combinatorial Mathematics and its Applications* (Proc. Conf., Oxford, 1969), pages 335–347. Academic Press, London, 1971.

Notes on the twisted graph

Elsa Omaña-Pulido[1], **Eduardo Rivera-Campo**[1]

[1] Departamento de Matemáticas, Universidad Autónoma Metropolitana – Iztapalapa, Av. San Rafael Atlixco 186, México D.F. 09340, México
eop@xanum.uam.mx, erc@xanum.uam.mx

Abstract. The twisted graph T_n is a complete topological graph with n vertices $v_1, v_2, \ldots, v_n$ in which two edges v_iv_j $(i < j)$ and v_sv_t $(s < t)$ cross each other if and only if $i < s < t < j$ or $s < i < j < t$. We study several properties concerning plane topological subgraphs of T_n.

Introduction

Let P be a set of points in the plane. A *topological graph* with vertex set P is a simple graph drawn in the plane with Jordan curves as edges in such a way that any two edges have at most one point in common. A *geometric graph* is a topological graph in which all edges are straight line segments.

Two topological graphs G and G' with vertex sets P and P', respectively, are *weakly isomorphic* if there is a bijection $\alpha\colon P \to P'$ such that $\alpha(u)\,\alpha(v) \in E(G')$ if and only if $uv \in E(G)$ and two edges $\alpha(x)\,\alpha(y)$ and $\alpha(u)\,\alpha(v)$ of G' intersect each other if and only if xy and uv intersect each other in G.

The *twisted graph* T_n is a complete topological graph with n vertices $v_1, v_2, \ldots, v_n$ in which two edges v_iv_j $(i < j)$ and v_sv_t $(s < t)$ cross each other if and only if $i < s < t < j$ or $s < i < j < t$; see Fig. 1.

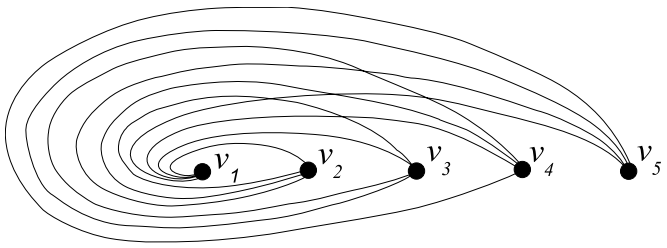


FIGURE 1. A twisted graph T_5.

The graphs T_n, $n \geq 5$ were introduced by Harborth and Mengersen [**4**] as examples of complete topological graphs containing no topological subgraphs weakly isomorphic to the complete convex geometric graph C_5 on 5 vertices. Later, Pach *et al.* [**7**] proved that every complete topological graph with n vertices contains a complete topological subgraph with $m \geq c \log^{1/8} n$ vertices which is weakly isomorphic to either the complete convex graph C_m or to the twisted graph T_m. In this paper we study several problems for the twisted graph T_m for which solutions for the corresponding problems for the complete geometric convex graph C_m are known. Due to the above result by Pach *et al.*, in most cases a result for general topological graphs follows.

[1]Partially supported by Conacyt 83856, México.

A *colouring* of the vertices of a graph G is an assignment of colours to the vertices of G where two adjacent vertices may have the same colour. A colouring of a graph G with two colours is a *balanced colouring* of G if the colour classes have the same size. In Section 1 we show that, for each balanced colouring of T_n, there is a plane spanning path of T_n whose vertices alternate colours. For the corresponding problem for complete geometric graphs C_n with vertices in convex position, the best known lower bound for the largest plane alternating path is $\frac{n}{2} + c\sqrt{n/\log n}$ [**6**]. Akiyama and Urrutia [**1**] gave an algorithm to decide whether there is a plane alternating spanning path for a given balanced colouring of C_n.

The *tree graph* $G(T)$ of a topological graph T is the abstract graph whose vertices are the plane spanning trees of T in which two trees Q and R are adjacent if there are edges q and r of T such that $R = (Q - q) + r$. Avis and Fukuda [**2**] proved that the graph $G(T)$ is connected whenever T is a geometric graph. In Section 2 we prove that $G(T_n)$ is always connected.

A plane topological subgraph G of T_n is a *maximal plane subgraph* of T_n if, for each edge uv of T_n not in G, there is an edge xy of G that intersects uv. The *max graph* $MP(T_n)$ of T_n is the abstract graph whose vertices are the maximal plane topological subgraphs of T_n in which two graphs F and H are adjacent if one can be obtained from the other by a single edge exchange. The *matching graph* $\mathcal{M}(T_{2m})$ of T_{2m} is the abstract graph with vertex set given by the set of plane perfect matchings of T_{2m} where two matchings L and N are adjacent if the symmetric difference $L \triangle N$ is a plane cycle with four edges. We prove that both graphs $MP(T_n)$ and $\mathcal{M}(T_{2m})$ are always connected.

Let G be a geometric graph with $n \geq 3$ vertices in convex position, Rivera-Campo [**8**] proved that if $G - v$ has a plane spanning tree for each vertex v of G, then G also has a plane spanning tree. In Section 4 we study the corresponding problem for topological subgraphs of T_n.

Throughout the paper we denote by $v_1, v_2, \ldots, v_n$ the vertices of T_n from left to right as in Fig. 1.

1 Alternating paths

Let c be a colouring of the vertex set of a topological graph G with two colours. An *alternating path* of G is a path in G whose vertices alternate colours.

Theorem 1.1. *For every balanced colouring of T_n with two colours, there is a plane alternating spanning path of T_n.*

Proof. Let c be a balanced colouring of T_n. Start a path at vertex $v_{i_1} = v_1$ and, for $k = 1, 2, \ldots, n-1$, let the next vertex $v_{i_{k+1}}$ be such that

$$i_{k+1} = \min \{j \,:\, v_j \neq v_{i_1}, v_{i_2}, \ldots, v_{i_k} \text{ and } c(v_j) \neq c(v_{i_k})\}.$$

Since $c(v_{i_1}) \neq c(v_{i_2})$, the path v_{i_1}, v_{i_2} is a plane alternating path R_1 of T_n with length one. Let $k > 1$ and assume that the path $R_{k-1} = v_{i_1}, v_{i_2}, \ldots, v_{i_k}$ is a plane alternating path of T_n. If the edge $v_{i_k}v_{i_{k+1}}$ intersects an edge $v_{i_j}v_{i_{j+1}}$ of S_{k-1}, then either $v_{i_k} < v_{i_j}$, $v_{i_{j+1}} < v_{i_{k+1}}$, $v_{i_{k+1}} < v_{i_j}$, $v_{i_{j+1}} < v_{i_k}$, $v_{i_j} < v_{i_k}$, $v_{i_{k+1}} < v_{i_{j+1}}$, or $v_{i_{j+1}} < v_{i_k}$, $v_{i_{k+1}} < v_{i_j}$. We claim that we reach a contradiction in all cases because of the choice of $v_{i_{j+1}}$ and of $v_{i_{k+1}}$. Therefore, $R_k = v_{i_1}, v_{i_2}, \ldots, v_{i_k}, v_{i_{k+1}}$ is a plane alternating path of T_n with length k. By induction, T_n contains a plane spanning alternating path R_{n-1}. □

2 Tree graph

Let S_n be the plane spanning tree of T_n with edges $v_1v_2, v_1v_3, \ldots, v_1v_n$. That is, S_n is the spanning star of T_n with center vertex v_1.

Theorem 2.1. *For each plane spanning tree R of T_n there is a path $R_1, R_2, \ldots, R_t$ in $G(T_n)$ with $R = S_n$ and $R_t = R$.*

Proof. For each plane spanning tree R of T_n, let $k(R)$ be the largest integer j such that $v_1v_2, v_1v_3, \ldots, v_1v_j$ are edges of R. If $n - k(R) = 0$, then $S_n = R$. Assume that $n - k(R) = m + 1$ and that the result holds for each plane spanning tree R' of T_n for which $n - k(R') = m$.

Since R is a plane subgraph of T_n and $v_1v_{k(R)}$ is an edge of R, for $2 \leq i < j \leq k(R)-1$ the edge v_iv_j of T_n cannot be an edge of R. This implies that $v_1v_{k(R)+1}$ does not intersect any edge of R and therefore $R+v_1v_{k(R)+1}$ is a plane subgraph of T_n. Clearly $R+v_1v_{k(R)+1}$ contains an edge $uv_{k(R)+1}$ with $u \neq v_1$ such that $R' = \left(R + v_1v_{k(R)+1}\right) - uv_{k(R)+1}$ is a tree. Then R' is a plane spanning tree of T_n with $k(R') = k(R) + 1$ and $n - k(R') = m$. By induction there is a path $R_1, R_2, \ldots, R_t$ in $G(T_n)$ with $S_n = R_1$ and $R_t = R'$. Since $R'$and R are adjacent in $G(T_n)$, $R_1, R_2, \ldots, R_t, R$ is also a path in $G(T_n)$. □

3 Max graph and matching graph

A *vertex ordering* of a graph G is a numbering of the vertices $v_1, v_2, \ldots, v_n$ of G. Let G be a graph with a vertex ordering $v_1, v_2, \ldots, v_n$. Two edges v_iv_j $(i < j)$ and v_sv_t $(s < t)$ of G are *nested* if $i < s < t < j$ or $s < i < j < t$. A set X of edges of G is a *queue* if no two edges in X are nested. Clearly the sets of edges of plane subgraphs of T_n are queues of the complete graph K_n with the corresponding vertex ordering. Dujmović and Wood **[3]** proved that every maximal queue of a graph with n vertices has at most $2n-3$ edges. The following lemmas will be used in the proof of Theorem 3.3.

Lemma 3.1. *If $n \geq 3$, then the following properties are satisfied by any maximal plane topological subgraph F of T_n.*

(1) *F has $2n - 3$ edges.*
(2) *If v_1v_k is an edge of F, then $v_1v_2, v_1v_3, \ldots, v_1v_{k-1}$ are also edges of F.*
(3) *If v_tv_n is an edge of F, then $v_{t+1}v_n, v_{t+2}v_n, \ldots, v_{n-1}v_n$ are also edges of T_n.*
(4) *v_1v_2, v_1v_3, $v_{n-2}v_n$ and $v_{n-1}v_n$ are edges of F.*

For any maximal plane topological subgraph F of T_n, let $k(F)$ be the largest integer k such that v_1v_k is an edge of F. By Lemma 3.1 (3), $k(F) \geq 2$. Let $MP_2(T_n)$ be the subgraph of $MP(T_n)$ induced by the set of maximal plane topological subgraphs F of T_n for which $k(F) = 2$.

Lemma 3.2. *For $n \geq 2$, $MP_2(T_{n+1})$ and $MP(T_n)$ are isomorphic graphs.*

Theorem 3.3. *For each positive integer n, the graph $MP(T_n)$ is connected.*

Proof. The graphs $MP(T_1)$, $MP(T_2)$ and $MP(T_3)$ contain exactly one vertex and therefore are connected. We proceed by induction assuming $n \geq 3$ and that $MP(T_n)$ is a connected graph. We claim that, for each maximal plane topological subgraph F of T_{n+1} with $k(F) = k$, there is a path $F_k, F_{k-1}, \ldots, F_2$ in $MP(T_{n+1})$ such that $F = F_k$ and F_2 lies in $MP_2(T_{n+1})$. By the induction hypothesis and by Lemma 3.2, the graph

$MP_2(T_{n+1})$ is connected. Therefore $MP(T_{n+1})$ is also connected and the theorem follows by induction. □

Hernando *et al.* [**5**] proved that, given any two plane perfect matchings N and M of the complete convex geometric graph C_{2m}, there is a sequence $N = N_0, N_1, \ldots, N_t = M$ of plane perfect matchings of C_{2m} such that, for $i = 0, 1, \ldots, t-1$, the symmetric difference $N_i \triangle N_{i+1}$ is a plane cycle with four edges. Here we prove the corresponding result for the twisted graph T_{2m}.

For any plane matching N of T_{2m}, let $k(N)$ be such that $v_1v_k \in N$ and let $\mathcal{M}_2(T_{2m})$ be the subgraph of $\mathcal{M}(T_{2m})$ induced by the matchings N for which $k(N) = 2$.

Lemma 3.4. *For $m \geq 1$, $\mathcal{M}_2(T_{2m+2})$ and $\mathcal{M}(T_{2m})$ are isomorphic graphs.*

Theorem 3.5. *For each positive integer n, the graph $MP(T_n)$ is connected.*

Proof. The graph $\mathcal{M}(T_2)$ is a graph with exactly one vertex. Assume $m \geq 1$ and that $\mathcal{M}(T_{2m})$ is a connected graph. We claim that, for each plane matching N of T_{2n+2} with $k(N) = k$, there is a path $N_t, N_{t-1}, \ldots, N_2$ in $\mathcal{M}(T_{2m})$ such that $N = N_t$ and N_2 lies in $\mathcal{M}_2(T_{2m+2})$. By the induction hypothesis and by Lemma 3.4, the graph $\mathcal{M}_2(T_{2m+2})$ is connected. Therefore $\mathcal{M}(T_{2m})$ is also connected, which completes the induction. □

4 Plane spanning trees

Theorem 4.1. *Let $n \geq 3$ and let G be a topological spanning subgraph of T_n. If $G - v$ contains a plane spanning tree for each $v \in V(G)$, then G also contains a plane spanning tree.*

Proof. Let $t = \min\{j : v_1v_j \in E(G)\}$ and let F_t be a plane spanning tree of $G - v_t$. Since F_t is connected, there is at least one edge v_1v_s of F_t incident with v_1. As no edge of F_t intersects the edge v_1v_s, we have $v_iv_j \notin E(F_t)$ for $1 < i < j < s$. By the choice of t, $s > t$ and therefore v_1v_t does not intersect any edge of F_t. This implies that F_t together with the edge v_1v_t is a plane spanning tree of G. □

References

[1] J. Akiyama, J. Urrutia, Simple alternating path problem, Discrete Math. 84 (1990), no. 1, 101–103.
[2] D. Avis, K. Fukuda, Reverse search enumeration, Discrete Applied Math. 6 (1996), 21–46.
[3] V. Dujmović, D. R. Wood, On linear layouts of graphs, Discrete Maths. & Theoretical Computer Science 6 (2004), no. 2, 339–358.
[4] H. Harborth, I Mengersen, Drawings of the complete graph with maximum number of crossings, in *Proceedings of the Twenty-Third Southeastern International Conference on Combinatorics, Graph Theory and Computing* (Boca Raton, FL 1992), Congressus Numerantium 88, Utilitas Math., Winnipeg, 1992, 225–228.
[5] C. Hernando, F. Hurtado, M. Noy, Graphs of non-crossing perfect matchings, Graphs and Combin. 18 (2002), 517–532.
[6] J. Kynčl, J. Pach, G. Tóth, Long alternating paths in bicolored point sets, Discrete Math. 308 (2008), no. 19, 4315–4321.
[7] J. Pach, J. Solymosi, G. Tóth, Unavoidable configurations in complete topological graphs, Discrete Comput. Geom. 30 (2003), 311–320.
[8] E. Rivera-Campo, A note on the existence of plane spanning trees of geometric graphs, in *Discrete and Computational Geometry* (Tokyo, 1998), 274–277, Lecture Notes in Comput. Sci., 1763, Springer, Berlin, 2000.

Non-crossing configurations revisited

Anna de Mier[1], **Marc Noy**[1]

[1] Universitat Politècnica de Catalunya, Jordi Girona 1–3, 08034 Barcelona, Spain
anna.de.mier@upc.edu, marc.noy@upc.edu

Abstract. We provide a new decomposition scheme for non-crossing graphs on the vertices of a convex polygon. This allows us to analyze the limiting degree distribution, the maximum degree, and the size of the largest connected component.

Introduction

Flajolet and Noy [**6**] studied non-crossing configurations from an enumerative and probabilistic perspective, obtaining precise asymptotic estimates for the number of configurations of several kinds, as well as limit probability laws for several basic parameters, such as the number of edges or the number of components.

Motivated by an open question of Clemens Huemer [**10**], we revisit the problem and study more advanced parameters, such as the limiting degree distribution, the maximum degree, and the size of the largest component. Our approach is based on a new decomposition of non-crossing configurations into 2-connected components inspired in recent work [**4, 9**], together with analytic and probabilistic methods.

1 The decomposition scheme

Consider the vertices $p_1, \ldots, p_n$ of a convex polygon in the plane, labelled counterclockwise. A *non-crossing graph* (or configuration) is any graph on this set of vertices such that when the edges are drawn as straight lines the only intersections occur at vertices. The *root* of a graph is vertex p_1. We will sometimes call the edge $\{p_1, p_2\}$ (if present) the *root edge*. From now on, all graphs are assumed to be non-crossing graphs.

We follow the approach introduced in [**9**], which consists in decomposing an arbitrary graph into connected components and then expressing these in terms of the 2-connected components. We denote by $G(z)$ and $C(z)$ the generating functions for arbitrary and connected graphs, respectively, counted by the number of vertices. Let $B(z)$ be the generating function for 2-connected graphs (blocks) where the variable z marks the number of non-root vertices minus one. The 2-connected graphs in our case are the same as polygon dissections; we will use both names.

The following equation (which is Equation (19) in [**6**]) links $G(z)$ and $C(z)$:

$$G(z) = 1 + C(zG(z)).$$

This expresses the fact that a graph can be constructed from a connected graph C by placing an arbitrary graph between every pair of consecutive vertices of C.

The relation between $C(z)$ and $B(z)$ is

$$C(z) = \frac{z}{1 - B(C(z)^2/z)}.$$

CRM Documents, vol. **8**, Centre de Recerca Matemàtica, Bellaterra (Barcelona), 2011

To see this, consider the ordered sequence $\beta_1, \ldots, \beta_k$ of blocks that contain the root vertex p_1. Now, to each vertex other than the root in each β_i there are two connected graphs attached, one to the left and one to the right of the vertex. Since these two graphs share a vertex, we have the term $C(z)^2/z$. Since $B(z)$ does not count the root vertex in a block, the substitution is correct and the result follows.

Finally, we have the following equation for $B(z)$:

$$B(z) = z + \frac{B(z)^2}{1 - B(z)},$$

which appears also in [**6**] in an equivalent form. To obtain this equation, consider the face of the dissection that contains the root edge $\{p_1, p_2\}$. Following the other edges of this face cyclically, we see that each of them can be viewed as the root edge of another dissection. Hence, a dissection with at least three vertices is a sequence of dissections joined into a cycle by their root edges and an extra edge.

2 Degree distribution and maximum degree

Here we study the parameter 'degree of the root vertex' in graphs. We let $G(z, w)$ be the generating function of graphs, where w marks the degree of the root. The equations in the previous section can be enriched, yielding the following:

$$G(z, w) = 1 + C(zG(z), w), \tag{1}$$

$$C(z, w) = \frac{z}{1 - B(C(z)^2/z, w)}, \tag{2}$$

$$B(z, w) = wz + \frac{wB(z, w)B(z)}{1 - B(z)}. \tag{3}$$

Let d_k^B be the limiting probability that the root in a 2-connected graph has degree k, that is,

$$d_k^B = \lim_{n\to\infty} \frac{[z^n][w^k]B(z, w)}{[z^n]B(z)}.$$

Define analogously d_k^C and d_k^G for connected and arbitrary graphs.

The following was proved in [**4**], and independently in [**1**].

Theorem 2.1. *The limiting distribution of the root degree in dissections is given by*

$$\sum_{k\geq 1} d_k^B w^k = \frac{2(3 - 2\sqrt{2})w^2}{(1 - (\sqrt{2} - 1)w)^2} = 0.34w^2 + 0.28w^3 + 0.18w^4 + 0.01w^5 + 0.05w^6 + \cdots.$$

Using similar tools as in [**4**], applied to Equations (1) and (2), we prove:

Theorem 2.2. (*a*) *The limiting distribution of the root degree in connected graphs is given by*

$$\sum_{k\geq 1} d_k^C w^k = \frac{(1 - \frac{1}{\sqrt{3}})^2}{2} \frac{w(w + 1 + \sqrt{3})}{(1 - (1 - \frac{1}{\sqrt{3}})w)^2} = 0.24w + 0.29w^2 + 0.21w^3 + 0.12w^4 + 0.07w^5 + \cdots.$$

(*b*) *The limiting distribution of root degree in graphs is given by*

$$\sum_{k\geq 1} d_k^G w^k = \frac{(1 - \sqrt{2})^2}{2} \frac{(1 + w)^2}{(1 - (\sqrt{2} - 1)w)^2} = 0.09 + 0.24w + 0.27w^2 + 0.18w^3 + 0.11w^4 + \cdots.$$

The maximum degree for polygon triangulations was first studied in [**2**], showing that it is of order $\log n$. A refined analysis [**8**] showed convergence to an extreme value distribution. For non-crossing graphs, we show convergence in probability of the maximum degree after logarithmic scaling.

Let us define Δ_n^B as the maximum vertex degree in dissections of size n, and similarly for Δ_n^C and Δ_n^G. Using the first and second moment method, together with precise estimates for the number $b(n, k, \ell)$ of dissections of size n in which the root has degree k and a second vertex different from the root has degree ℓ, it is proved in [**5**] that the maximum degree is of order $\log n$. More precisely:

Theorem 2.3. *The maximum degree in dissections satisfies*

$$\frac{\Delta_n^B}{\log n} \longrightarrow \frac{1}{\log(1+\sqrt{2})} \quad \textit{in probability.}$$

The fact that an arbitrary graph is obtained from a dissection by removing a subset of the boundary edges, implies that the same result holds.

Theorem 2.4. *The maximum degree in graphs satisfies*

$$\frac{\Delta_n^G}{\log n} \longrightarrow \frac{1}{\log(1+\sqrt{2})} \quad \textit{in probability.}$$

An analogous result should hold for connected graphs. However, a full proof needs to adapt the arguments from [**5**], involving quite subtle multivariate estimates obtained through double Cauchy integrals, to this situation. We believe this is only a technical matter, and one should expect the following, that we choose to formulate as a conjecture in the absence of a detailed proof.

Conjecture 2.5. *The maximum degree in connected graphs satisfies*

$$\frac{\Delta_n^C}{\log n} \longrightarrow \frac{1}{\log((3+\sqrt{3})/2)} \quad \textit{in probability.}$$

3 Largest component

Let $G(z, u)$ be the generating function of non-crossing graphs, where now u marks the size of the connected component containing the root. Elementary considerations imply

$$G(z, u) = 1 + C(uzG(z)).$$

It is easily checked that this is a *subcritical* composition scheme, in the terminology of [**7**], meaning that the evaluation of $zG(z)$ at its dominant singularity is smaller than the singularity of $C(z)$. Then Proposition IX.1 from [**7**] applies and we have:

Theorem 3.1. *The size X_n of the connected component containing the root has a discrete limit distribution, given by*

$$\mathbf{P}(X_n = k) \to p_k \sim c \cdot k^{-1/2} q^k, \quad \textit{as } k \to \infty,$$

for some q with $0 < q < 1$.

Consider the random variable $Y_{n,k}$, defined on graphs of size n, equal to the number of vertices contained in a component of size k. From the previous theorem it follows that $\mathbf{E}(Y_{n,k}) \sim p_k n$. The exponential tail of the distribution p_k suggests that the largest

component is of order $\log n$ with high probability. In fact, we believe the following is true, and can be proved using tools similar to those for analyzing the maximum degree.

Conjecture 3.2. *Let L_n be the size of the largest component in graphs of size n. Then there exists $c > 0$ such that*

$$\frac{L_n}{\log n} \longrightarrow c \quad \textit{in probability.}$$

An analogous conjecture can be made for the size of the largest 2-connected component.

4 Concluding remarks

The open question from [**10**] mentioned in the introduction was to count bipartite non-crossing graphs. This amounts to count graphs in which all faces have even size. For dissections this is easy (recursively allow even sizes for the face containing the root edge), and it can be extended to connected and arbitrary graphs using our decomposition scheme. It can be shown then that the number a_n of bipartite graphs satisfies

$$a_n \sim cn^{-3/2}\gamma^n,$$

where $\gamma \approx 7.5289$ is the largest positive root of $2z^4 + 52y^3 - 417y^2 - 658y - 27 = 0$. The first values are $\sum a_n z^n = z + 2z + 7z^3 + 34z^4 + 196z^5 + \cdots$. Clearly, the same scheme can be used to count graphs with girth at least g, and several other restrictions on the size of the faces.

On the other hand, it is shown in [**3**] that the number of vertices of degree k in dissections is asymptotically normally distributed for fixed k, with linear expectation and variance. Using the same technique, based on the analysis of systems of functional equations, we should be able to prove the same result for non-crossing graphs.

Finally, we mention as an open problem to investigate the diameter of random non-crossing graphs. We suspect that, as it is the case for trees, it should be of order $\sqrt{n}$ w.h.p. The first step would be to prove such a statement for random dissections.

References

[1] N. Bernasconi, K. Panagiotou, A. Steger, The degree sequence of random graphs from subcritical classes, *Combin. Probab. Comput.* 18 (2009), 647–681.

[2] L. Devroye, P. Flajolet, F. Hurtado, M. Noy, W. Steiger, Properties of random triangulations and trees, *Discrete Comput. Geom.* 22 (1999), 105–117.

[3] M. Drmota, O. Giménez, M. Noy, Vertices of given degree in series-parallel graphs, *Random Structures Algorithms* 36 (2010), 273–314.

[4] M. Drmota, O. Giménez, M. Noy, Degree distribution in random planar graphs, *J. Combin. Theory Ser. A* (to appear). arXiv:0911.4331 (32 pages).

[5] M. Drmota, O. Giménez, M. Noy, The maximum degree of planar graphs I. Series-parallel graphs. arXiv:1008.5361 (38 pages).

[6] P. Flajolet, M. Noy, Analytic combinatorics of non-crossing configurations, *Discrete Math.* 204 (1999), 203–229.

[7] P. Flajolet, R. Sedgewick, *Analytic combinatorics*, Cambridge U. Press, Cambridge, 2009.

[8] Z. Gao, N. C. Wormald, The distribution of the maximum vertex degree in random planar maps, *J. Combin. Theory Ser. A* 89 (2000), 201–230.

[9] O. Giménez, M. Noy, Asymptotic enumeration and limit laws for planar graphs, *J. Amer. Math. Soc.* 22 (2009), 309–329.

[10] C. Huemer, *Structural and Enumerative Problems for Plane Geometric Graphs*, Ph.D. Thesis, Universitat Politècnica de Catalunya, 2007.

Sweeping an oval to a vanishing point

Adrian Dumitrescu[1], **Minghui Jiang**[2]

[1] University of Wisconsin–Milwaukee, WI 53201-0784, USA
dumitres@uwm.edu

[2] Utah State University, Logan, UT 84322-4205, USA
mjiang@cc.usu.edu

Abstract. Given a convex region in the plane, and a sweep-line as a tool, what is the best way to reduce the region to a single point by a sequence of sweeps? The problem of sweeping points by orthogonal sweeps was first studied in [**2**]. Here we consider the following *slanted* variant of sweeping recently introduced in [**1**]: In a single sweep, the sweep-line is placed at a start position somewhere in the plane, then moved continuously according to a sweep vector $\vec{v}$ (not necessarily orthogonal to the sweep-line) to another parallel end position, and then lifted from the plane. The cost of a sequence of sweeps is the sum of the lengths of the sweep vectors. The optimal sweeping cost of a region is the infimum of the costs over all finite sweeping sequences for that region. An optimal sweeping sequence for a region is one with a minimum total cost, if it exists. Another parameter of interest is the number of sweeps.

We show that there exist convex regions for which the optimal sweeping cost cannot be attained by two sweeps. This disproves a conjecture of Bousany, Karker, O'Rourke, and Sparaco stating that two sweeps (with vectors along the two adjacent sides of a minimum-perimeter enclosing parallelogram) always suffice [**1**]. Moreover, we conjecture that, for some convex regions, no finite sweeping sequence is optimal. On the other hand, we show that both the 2-sweep algorithm based on minimum-perimeter enclosing rectangle and the 2-sweep algorithm based on minimum-perimeter enclosing parallelogram achieve a $4/\pi \approx 1.27$ approximation of the optimal sweeping cost in this model.

Introduction

The following question was raised by Paweł Żyliński [**4**]; see also [**2**]: Given a set of points in the plane, and a sweep-line as a tool, what is the best way to move the points to a target point using a sequence of sweeps? The target point may be specified in advance or freely selected by the algorithm. In a single sweep, the sweep-line is placed in the plane at some start position, then moved orthogonally and continuously to another parallel end position, and then lifted from the plane. All points touched by the line are moved with the line in the direction of the sweep. When a point is swept over another point, and both are in the current set, the two points merge into one, and they are subsequently treated as one point. The cost of a sequence of sweeps is the total length of the sweeps, with no cost assessed for positioning or repositioning the line. Dumitrescu and Jiang have obtained several results on this question, among which we mention a ratio $4/\pi \approx 1.27$ approximation that uses at most 2 sweeps (or 4 sweeps if the target point is specified) and can be computed in $O(n \log n)$ time. We refer to this (original) model of sweeping as the *orthogonal sweeping model*.

Bousany et al. [**1**] have recently explored another variant, with points being replaced by a planar connected region, and orthogonal sweeps replaced by (possibly non-orthogonal) *slanted sweeps*. We refer to their model of sweeping as the *slanted sweeping*

[1]Supported in part by NSF CAREER grant CCF-0444188 and NSF grant DMS-1001667.
[2]Supported in part by NSF grant DBI-0743670.

model or the *generalized sweeping model*: in a sweep operation the infinite sweep line may translate by any vector $\vec{v}$, not only by a vector orthogonal to the line; the corresponding cost is the (Euclidean) vector length $|\vec{v}|$. The goal is to sweep the given region by a sequence of sweeps to a single unspecified target point.

Given a planar region, the *optimal sweeping cost* (or just *sweeping cost*) of the region is the infimum of the costs over all finite sweeping sequences of that region to a single point. An *optimal sweeping sequence* is one with a minimum total cost, if it exists. It is conceivable that the optimal sweeping cost of a region could be only approached in the limit, and not be attained through a finite sequence of sweeps. Another parameter of interest is the number of sweeps. We refer to a sequence of k sweeps as a *k-sweep sequence*.

It is easy to exhibit non-convex planar regions whose optimal 2-sweep sequences are not optimal over all sequences [**1**]. Given a convex n-gon, Bousany et al. derived a linear-time algorithm for computing a minimum-perimeter parallelogram enclosing P, and thereby the optimal corresponding 2-sweep sequence [**1**]. They went further and conjectured that, for planar convex regions, an optimal 2-sweep sequence makes in fact an optimal sweep sequence. Here we disprove this conjecture and thereby answer the main problem left open in [**1**].

Theorem 1. *There exist convex regions (such as the Reuleaux triangle, or a disk, or a suitable isosceles trapezoid) for which an optimal* 2*-sweep sequence is not optimal.*

We present two proofs of Theorem 1, based on different counterexamples. While the first proof is shorter, the second gives a better lower bound on the approximation ratio of the 2-sweep algorithm (minimum-perimeter enclosing parallelogram) from [**1**]. Moreover, the second proof also implies the existence of convex polygons with n sides, for any $n \geq 4$, for which an optimal 2-sweep sequence is not optimal.

Corollary 2. *For every $n \geq 4$ there exist convex polygons with n vertices for which an optimal* 2*-sweep sequence is not optimal.*

In light of Theorem 1, one may naturally ask whether the 2-sweep algorithm of Bousany et al. has a good approximation ratio. In our previous paper based on the orthogonal sweeping model [**2**], we showed that a simple algorithm A2, which first computes a minimum-perimeter rectangle enclosing the given point set, then moves the point set to a single point by two orthogonal sweeps (four orthogonal sweeps if the target point is not freely chosen but is specified in the input), achieves an approximate ratio of $4/\pi$ in that model. Here we further show that the same algorithm A2 also achieves an approximation ratio of $4/\pi$ in the slanted sweeping model introduced in [**1**].

Theorem 3. *The* 2*-sweep Algorithm A*2 *based on minimum-perimeter enclosing rectangle (from* [**2**]*) gives a $4/\pi$-approximation of the optimal sweeping cost of a (discrete or continuous) point set in the slanted sweeping model.*

Now for any point set S, the minimum-perimeter of a parallelogram enclosing S is at most the minimum-perimeter of a rectangle enclosing S. Thus Theorem 3 implies that the approximation ratio of the 2-sweep algorithm of Bousany et al. is also at most $4/\pi$.

Corollary 4. *The* 2*-sweep algorithm based on minimum-perimeter enclosing parallelogram (from* [**1**]*) gives a $4/\pi$-approximation of the optimal sweeping cost of a (discrete or continuous) point set in the slanted sweeping model.*

A full version of this paper is available at `http://arxiv.org/abs/1101.4667`.

1 Proof of Theorem 1

We start with an upper bound on the cost of sweeping a convex polygon.

Lemma 5. *Let $P = A_1 \dots A_n$ be a convex polygon of perimeter* $\mathrm{per}(P)$*, and let s be an arbitrary side of P. Then the sweeping cost of P is at most* $\mathrm{per}(P) - |s|$.

Proof. We can assume that $s = A_1A_n$. Consider the triangulation of P from the single point A_n; see Fig. 1. Make the first sweep with the line initially incident to A_1 and parallel

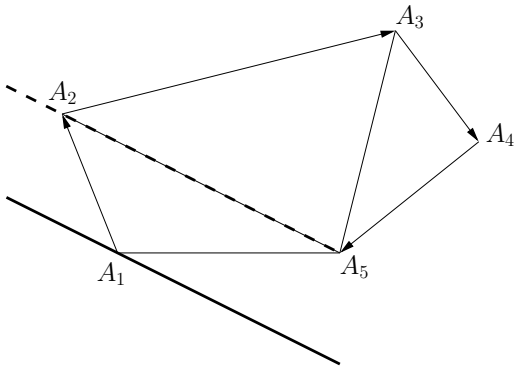


FIGURE 1. Sweeping a convex polygon; here $n = 5$.

to A_2A_n until the line is incident to A_2A_n; the sweep vector is $\overrightarrow{A_1A_2}$. Observe that after the sweep, we reduced the problem of sweeping $P = A_1 \dots A_n$ to that of sweeping $P' = A_2 \dots A_n$. We continue in a similar way: in the ith sweep ($i = 1, \dots, n-2$), start with the line incident to A_i and parallel to $A_{i+1}A_n$ until the line is incident to $A_{i+1}A_n$; the sweep vector is $\overrightarrow{A_iA_{i+1}}$. In the last sweep, $n-1$, we position the line incident to A_{n-1} and (say) orthogonally to $A_{n-1}A_n$ and sweep until the line is incident to A_n. The polygon P has been swept to the point A_n in $n-1$ sweeps of total cost $|A_1A_2| + \dots + |A_{n-1}A_n| = \mathrm{per}(P) - |A_1A_n| = \mathrm{per}(P) - |s|$, as required. □

First proof of Theorem 1. We first observe that the minimum-perimeter enclosing parallelogram of a *convex figure* Θ *of constant width* w is a square whose side-length equals w. Indeed, assume that the parallelogram Γ encloses Θ, a figure of constant unit width. Let ℓ_1, ℓ_1' be a pair of parallel supporting lines of Γ, and let ℓ_2, ℓ_2' be the other pair of parallel supporting lines of Γ. The distance between ℓ_1 and ℓ_1' is 1, so the total length of the two parallel sides of Γ along ℓ_2 and ℓ_2' is at least 2, with equality if and only if $\ell_1 \perp \ell_2$. Similarly, the distance between ℓ_2 and ℓ_2' is 1, so the total length of the two parallel sides of Γ along ℓ_1 and ℓ_1' is at least 2, with equality if and only if $\ell_1 \perp \ell_2$. Hence the unit square is the minimum-perimeter enclosing parallelogram of Θ, and its semi-perimeter 2 is the conjectured sweep cost of Θ [**1**].

In the role of Θ, consider a *Reuleaux triangle* Δ obtained from an equilateral unit triangle ABC: a Reuleaux triangle can be obtained from an equilateral triangle ABC by joining each pair of its vertices by a circular arc whose center is at the third vertex [**3**]. We can assume that BC is horizontal and first sweep from A orthogonally to BC (the sweep-line is parallel to BC) until the sweep-line reaches BC. Observe that since the two upper tangents to Δ at B and C are vertical, the sweep reduces Δ to the circular cap $\Psi \subset \Delta$ based on BC. Let $B'C'$ be a slightly longer segment containing BC: $BC \subset B'C'$.

Consider a slightly larger concentric circular cap based on $B'C'$ enclosing the cap Ψ based on BC. Now make a fine equidistant subdivision of the circular arc connecting B' and C' to obtain a convex polygon Λ enclosing the circular cap Ψ. By Lemma 5, the cost of sweeping Ψ is at most the length of the arc BC plus ε, for any given $\varepsilon > 0$. So the total cost of sweeping the Reuleaux triangle Δ is at most $\frac{\sqrt{3}}{2} + \frac{\pi}{3} + \varepsilon \leq 1.9133$, for a suitably small ε. Since this cost is smaller than 2, the 2-sweep conjecture is disproved. □

2 Proof of Theorem 3

We refer to **[2]** for a description of our Algorithm A2 and its analysis in the orthogonal sweeping model. The analysis bounds the effect of each sweep on the current point set, in terms of the reduction in semi-perimeter of the minimum enclosing rectangle in every orientation β. We then integrate the total effect of all sweeps in an optimal (or nearly optimal solution) over all orientations. That is, we consider an arbitrary k-sweep sequence σ given by the vectors $\vec{v_i}$, $i = 1, \ldots, k$, where $x_i = |\vec{v_i}|$ and α_i are the length and the direction (angle) of $\vec{v_i}$, respectively. Let γ_i, $i = 1, \ldots, k$, be the direction (angle) of the sweep-line in the ith sweep. The cost of σ is $x = \sum_{i=1}^{k} x_i$. In the end we let $x \leq \mathrm{OPT} + \varepsilon$, where OPT is the optimal sweeping cost (in the analysis in **[2]** we overlooked the possibility that no finite sweeping sequence is optimal; this only introduces the correction by ε in the calculation —see below).

Fix an orientation β, and consider the minimum enclosing rectangle $Q_i(\beta)$ of the current point-set in this orientation just before the ith sweep. To adapt the analysis from the orthogonal sweeping model to the slanted sweeping model, we only need to show, as in **[2]**, that the reduction in the semi-perimeter of $Q_i(\beta)$ (to that of $Q_{i+1}(\beta)$ in the next step) due to the ith slanted sweep is still bounded from above by $x_i(|\cos(\alpha_i - \beta)| + |\sin(\alpha_i - \beta)|)$, the expression on the left-hand side of equation (1) in **[2]**.

Observe that, in the ith sweep, any point swept moves in the direction α_i by at most x_i, regardless of the sweep-line orientation γ_i. Thus the projections of this move onto the two orthogonal directions β and $\beta + \pi/2$ are at most $x_i|\cos(\alpha_i - \beta)|$ and $x_i|\sin(\alpha_i - \beta)|$, respectively. Hence their sum is at most $x_i(|\cos(\alpha_i - \beta)| + |\sin(\alpha_i - \beta)|)$, and consequently the reduction in the semi-perimeter of $Q_i(\beta)$ is bounded by the same quantity, as claimed. Analogously as in **[2]**, by integration we find that the approximation ratio of the 2-sweep algorithm is bounded from above by $4/\pi + \varepsilon$, for any $\varepsilon > 0$. By letting ε tend to zero, we conclude that this ratio is at most $4/\pi$, as required. □

References

[1] Y. Bousany, M. L. Karker, J. O'Rourke, and L. Sparaco, Sweeping minimum perimeter enclosing parallelograms: optimal crumb cleanup, in *Proceedings of the 22nd Canadian Conference on Computational Geometry* (CCCG 2010), Winnipeg, Manitoba, Canada, August 2010, pp. 167–170. Full version online at `http://www.cs.umanitoba.ca/~cccg2010/program.html`.

[2] A. Dumitrescu and M. Jiang, Sweeping points, *Algorithmica*, doi:10.1007/s00453-009-9364-6, to appear. A preliminary version in *Proceedings of the 11th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems* (APPROX 2008), MIT, Boston, USA, August 2008, Vol. 5171 of LNCS, Springer, pp. 63–76.

[3] I. M. Yaglom and V. G. Boltyanskiĭ, *Convex Figures*, Holt, Rinehart and Winston, New York, 1961.

[4] P. Żyliński, personal communication, June 2007.

String-wrapped rotating disks

Joseph O'Rourke[1]

[1] Department of Computer Science, Smith College, Northampton, MA 01063, USA
orourke@cs.smith.edu

Abstract. Let the centers of a finite number of disjoint, closed disks be pinned to the plane, but with each free to rotate about its center. Given an arrangement of such disks with each labeled + or −, we investigate the question of whether they can be all wrapped by a single loop of string so that, when the string is taut and circulates, it rotates by friction all the (+)-disks counterclockwise and all the (−)-disks clockwise, without any string-rubbing conflicts. We show that, although this is not always possible, natural disk-separation conditions guarantee a solution. This work suggests many open problems.

Introduction

Let $\mathcal{A}$ be a collection of n disjoint closed disks in the plane, each labeled + or −. We seek to wrap them all in one continuous loop of string so that, were one of the disks rotated by a motor, all the others would spin by friction with the string/belt in a direction consistent with the labeling: counterclockwise (ccw) for + and clockwise (cw) for −. See Figure 1.

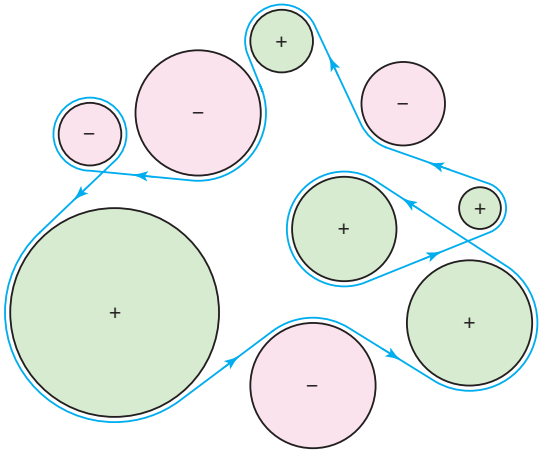


FIGURE 1. A proper wrapping of disks with a loop of string: each + disk rotates counterclockwise, each − clockwise.

We call a wrapping *proper* if it satisfies these conditions:

(1) The string is *taut*: it follows arcs of disk boundaries and disk-disk tangents only.
(2) Each disk boundary circle has a positive-length arc in contact with the string. (It is fine if the string wraps around a disk more than once.)
(3) One of the two possible circulation directions (i.e., orientations) for the string loop rotates each disk in the direction consistent with its labeling.
(4) If the string contacts a point of a disk boundary circle, its circulation there must be in the direction consistent with that disk's label, i.e., there is no *rubbing conflict*.

We permit the string to cross itself at points not on a disk boundary. Indeed, such crossings are necessary: for any pair of disks, one + and one −, the string must form a figure-8 shape. Although the conditions for a proper wrapping are suggested by physical analogy, the pursuit here is not driven by any application.

Proper string wrappings bear some resemblance to sona sand drawings [**DDTT07, LT09**], but are more closely related to the conveyor-belt wrappings studied in [**DDP10**]. Those belts differ from string wrappings in that the rotation directions were not prespecified, and the belt could not self-cross. Although the models are different, the questions raised are analogous.

We show that not all arrangements of disks have a proper wrapping, but that various separation conditions guarantee proper wrappings. For example, every collection of unit disks has a proper wrapping when each pair is separated by a distance of 0.31 or more. Characterizing the disk arrangements that admit proper wrappings is posed as an open problem in Section 3.

1 Unwrappable arrangements

An example of an unwrappable arrangement is shown in Figure 2.

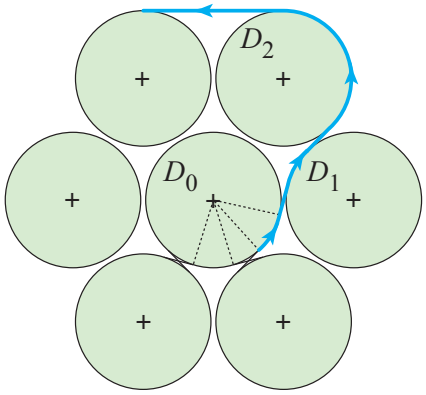


FIGURE 2. An unwrappable arrangement of seven unit disks.

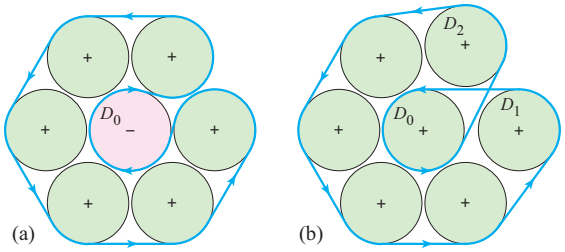


FIGURE 3. (a) Proper wrapping with D_0 labeled −.
(b) Proper wrapping with D_1 and D_2 displaced slightly.

It consists of one unit disk surrounded by six others, arranged in a hexagonal penny-packing pattern, except the disks are just barely disjoint. We now prove that this configuration is unwrappable.

The central disk D_0 must have a positive-length arc of ccw string touching it. Because a taut string can only leave the boundary of a disk along a tangent, the string follows at least the arc between two adjacent tangents. In order for the string to reach another disk, say D_2, and contribute a ccw arc, it must first touch another disk, D_1 in the figure, but now rubbing it in a cw arc. Thus a rubbing conflict is unavoidable.

Without moving the disks, this arrangement can be properly wrapped with a different pattern of ± labels. For example, reversing the central disk D_0 enables a proper wrapping: Figure 3(a). Indeed, all other ± patterns of labels (except all −) in this example are wrappable. Retaining the original + labels but moving two disks slightly also permits the configuration to be wrapped: Figure 3(b).

2 Separation conditions

The primary impediment to a proper wrapping is the 4th no-rubbing-conflicts condition. Figure 3(b) indicates that disk-separation conditions may suffice to ensure the existence of a proper wrapping, as separation of the disks separates their tangents and avoids unwanted rubbings. In this section we offer three conditions that ensure a proper string wrapping exits.

2.1 Connected hull-visibility graph

Define two disks to be *hull-visible* to one another (a symmetric relation) iff the (closed) convex hull of the disks does not intersect any other disk; see Figure 4. If two disks can

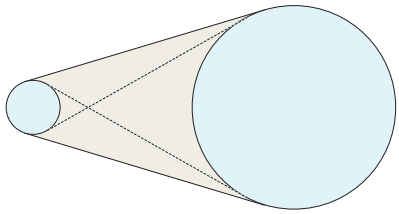


FIGURE 4. Two disks are visible to one another if their hull does not intersect any other disk.

see one another in this sense, then none of their four bi-tangents are blocked (or even touched) by any other disk.

For an arrangement $\mathcal{A}$ of disks, define their *hull-visibility graph* $H(\mathcal{A}) = H$ to have a node for each disk, and an arc connecting two disk nodes iff the disks are visible to one another. Call the hull of a pair of disks connected in H to the *edge corridor* for that edge.

Lemma 2.1 (Vis. Gr.). *If $H(\mathcal{A})$ is connected, then there is a proper wrapping of $\mathcal{A}$.*

The conditions of this lemma are by no means necessary for the existence of a proper wrapping: H for the configuration in Figure 3(a) is completely disconnected —seven isolated nodes— and yet it can be properly wrapped.

2.2 Unit disks halo

The sufficiency condition of Lemma 2.1 is a global property of the arrangement $\mathcal{A}$ of disks, not immediately evident upon inspection. Next we explore local separation conditions that allow us to conclude that H is connected.

Define an *α-halo*, $\alpha > 0$, for a disk D of radius r to be a concentric disk D' of radius $(1 + \alpha)$ such that no other disk of $\mathcal{A}$ intersects D'.

Lemma 2.2 (Unit Halo). *Let D_a, D_b, and D_c be three unit disks with centers at a, b, and c respectively, each with α-halos for $\alpha = 4/\sqrt{3} - 2 \approx 0.31$. Then, if D_c intersects the (D_a, D_b) corridor, c is closer to a and to b than is a to b: $|ac| < |ab|$ and $|bc| < |ab|$.*

Theorem 2.3 (Unit Disks). *An arrangement $\mathcal{A}$ of unit disks with α-halos, $\alpha = 4/\sqrt{3} - 2$, has a connected visibility graph $H(\mathcal{A})$, and so can be properly wrapped.*

2.3 Arbitrary radii halo

For disks of different radii, we define the distance between them to be the distance between their bounding circles (rather than their centers). The assumption in Theorem 2.3 that all disks are congruent can be removed at the cost of a significant increase in the value of α.

Note that, for disks of arbitrary radii, we must allow for a disk radius to be arbitrarily small, effectively a point regardless of α. Thus the strategy to avoid blockage of a (D_a, D_b) corridor is to cover it entirely with the α-halos of D_a and D_b, for no D_c can penetrate these halos by definition.

Lemma 2.4 ($\alpha = 1$ Halo)**.** *The conclusion of Lemma* 2.2 *holds for three disks D_a, D_b, and D_c of arbitrary radius if $\alpha = 1$.*

Now the analog of Theorem 2.3 follows immediately by an identical proof, only invoking Lemma 2.4 rather than Lemma 2.2:

Theorem 2.5 ($\alpha = 1$ Halo)**.** *Any arrangement $\mathcal{A}$ of disks with α-halos, $\alpha = 1$, has a connected visibility graph $H(\mathcal{A})$, and so can be properly wrapped.*

3 Conclusion

There is no question that Theorems 2.3 and 2.5 do not approach a full characterization of the conditions that ensure proper wrapping, as Figure 3 so dramatically indicates. Finding a tighter characterization is one central open question. Surely the $\alpha = 1$ halo is more generous than needed.

An approachable specific version of this question is an arrangement $\mathcal{A}$ of unit disks in a hexagonal-packing pattern (as in Figure 2, but with n disks at hexagon lattice points). Which $\pm$ labelings are wrappable? I can prove that, if the adjacency graph of like-labeled disks is a forest (a union of disjoint trees), then $\mathcal{A}$ is properly wrappable. But this sufficient condition is not necessary.

A second central open question is to find a shortest wrapping when proper wrappings exist. For widely spaced disks, this reduces to a version of TSP,[1] but the situation is less clear for congested arrangements.

References

[DDP10] E. D. Demaine, M. L. Demaine, and B. Palop. Conveyer-belt alphabet. In H. Aardse and A. van Baalen, eds., *Findings in Elasticity*, pages 86–89. Pars Foundation, Lars Müller, 2010.

[DDTT07] E. D. Demaine, M. L. Demaine, P. Taslakian, and G. T. Toussaint. Sand drawings and Gaussian graphs. *J. Math. Arts*, 1(2):125–132, 2007.

[LT09] Y. Liu and G. T. Toussaint. A simple algorithm for constructing perfect monolinear sona tree drawings, and its application to visual art education. In *Proc. 8th Internat. Conf. AI, KE, DB*, pages 288–294, 2009.

[1] I thank Erik Demaine for this observation.

On the number of radial orderings of colored planar point sets

J. M. Díaz-Báñez[1,4], R. Fabila-Monroy[2], P. Pérez-Lantero[3,4]

[1] Departamento de Matemática Aplicada II, Universidad de Sevilla, Spain
dbanez@us.es

[2] Departamento de Matemáticas, Centro de Investigación y Estudios Avanzados del Instituto Politécnico Nacional, México, D.F.
ruyfabila@math.cinvestav.edu.mx

[3] Departamento de Computación, Universidad de Valparaíso, Chile
pablo.perez@uv.cl

Abstract. Let S be a set of n *red* and n *blue* points in general position in the plane. Let $p \notin S$ be a point such that $S \cup \{p\}$ is in general position. A *radial ordering* of S with respect to p is a circular ordering of the elements of S by angle around p. A *colored radial ordering* of S with respect to p is the circular list of colors of the points in S in their radial ordering with respect to p. In this paper we show that: the number of distinct radial orderings of S is at most $O(n^4)$ and at least $\Omega(n^2)$; the number of colored radial orderings of S is at most $O(n^4)$ and at least $\Omega(n)$; there exist sets of points with $\Theta(n^4)$ radial and colored radial orderings; and there exist sets of points with $O(n^2)$ colored radial orderings.

Introduction

Throughout this paper S is a set of n *red* and n *blue* points in general position in the plane. Let p be a point not in S, such that $S \cup \{p\}$ is also in general position; we call p an *observation point*. A *radial ordering* of S with respect to p is a circular clockwise ordering of the elements of S by their angle around p. A *colored radial ordering* with respect to p is the circular list of the colors of the points in S in their radial ordering with respect to p. Thus permutations between points of the same color yield the same colored radial ordering. Unless otherwise noted, all point sets in this paper are in general position.

Let $\rho(S)$ be the number of distinct radial orderings of S with respect to every observation point in the plane. Likewise, let $\tilde{\rho}(S)$ be the number of distinct colored radial orderings of S with respect to every observation point in the plane. We define the following functions:

$$f(n) = \max\{\rho(S) \,:\, |S| = 2n\}; \quad \tilde{f}(n) = \max\{\tilde{\rho}(S) \,:\, |S| = 2n\};$$
$$g(n) = \min\{\rho(S) \,:\, |S| = 2n\}; \quad \tilde{g}(n) = \min\{\tilde{\rho}(S) \,:\, |S| = 2n\}.$$

In this paper we prove the following bounds:

$$f(n) = \Theta(n^4);$$
$$\tilde{f}(n) = \Theta(n^4);$$
$$\Omega(n^2) \leq g(n) \leq O(n^4);$$
$$\Omega(n) \leq \tilde{g}(n) \leq O(n^2).$$

[4]Partially supported by MEC project MTM2009-08652.

For the first equality, the fact that $f(n)$ is $O(n^4)$ has been noted before in the literature [**1, 2**]. As far as we know, all the other bounds are new.

1 Preliminaries

We discretize the problem by partitioning the set of observation points into a finite number of sets so that two points in a same element of the partition induce the same radial ordering. This partition is made by half-lines which, if crossed by an observation point, generate a transposition of two consecutive elements in the radial ordering. For every pair of points $x_1, x_2 \in S$, consider the line passing through them. Contained in this line we have two swap lines; one begins in x_1 and does not contain x_2, while the other begins in x_2 and does not contain x_1. Two observation points are in the same element of the partition if they can be connected by a curve which does not intersect any swap lines. We call this partition the *order partition*, and since it induces a decomposition of the plane, we refer to its elements as *cells*. Note that if a point moves in a curve not crossing any swap line, the radial ordering with respect to this point is the same throughout out the motion. Thus, as mentioned above, points in the same cell induce the same radial ordering.

2 Bounds

Theorem 2.1. *$f(n) \leq O(n^4)$ and $\tilde{f}(n) \leq O(n^4)$.*

Proof. The first inequality follows from the fact that the number of cells in the order partition is $O(n^4)$. The second inequality follows from the first and from the observation that $\tilde{\rho}(S) \leq \rho(S)$. □

Theorem 2.2. *$g(n) \geq \Omega(n^2)$.*

Proof. Let ℓ be a straight line having points of S both above and below. Let p and q be two points in ℓ, such that when walking from p to q in a straight line a swap line is crossed. Let $x_1, x_2 \in S$ be the points defining this swap line. We show that p and q induce different radial orderings. Note that x_1 and x_2 are both above or below ℓ. Let $x_3 \in S$ be on the side opposite to x_1 and x_2. Assume without loss of generality that the radial ordering of $\{x_1, x_2, x_3\}$ with respect to p is $[x_1, x_2, x_3]$. Since the swap line is crossed only once, the radial ordering of $\{x_1, x_2, x_3\}$ with respect to q is $[x_1, x_3, x_2]$. Therefore, the radial ordering of S with respect to p is different from the radial ordering with respect to q. This implies that the number of different radial orderings with respect to points in ℓ is equal to the number of swap lines intersecting ℓ plus one. It remains to show that there is a choice for ℓ crossing a quadratic number of swap lines. Choose ℓ to be a line having only one point of S above and all the others below, and not parallel to any swap line. Note that ℓ intersects every swap line defined by pairs of points of S below ℓ. Since there are $2n - 1$ such points, they define $\Theta(n^2)$ swap lines and the result follows. □

Theorem 2.3. *$f(n) \geq \Omega(n^4)$ and $\tilde{f}(n) \geq \Omega(n^4)$.*

Proof (sketch). Given that $\rho(S) \geq \tilde{\rho}(S)$, it suffices to construct a set P of n red and n blue points such that $\tilde{\rho}(P) \geq \Omega(n^4)$. We sketch such a construction but for lack of space we omit counting the number different colored radial orderings.

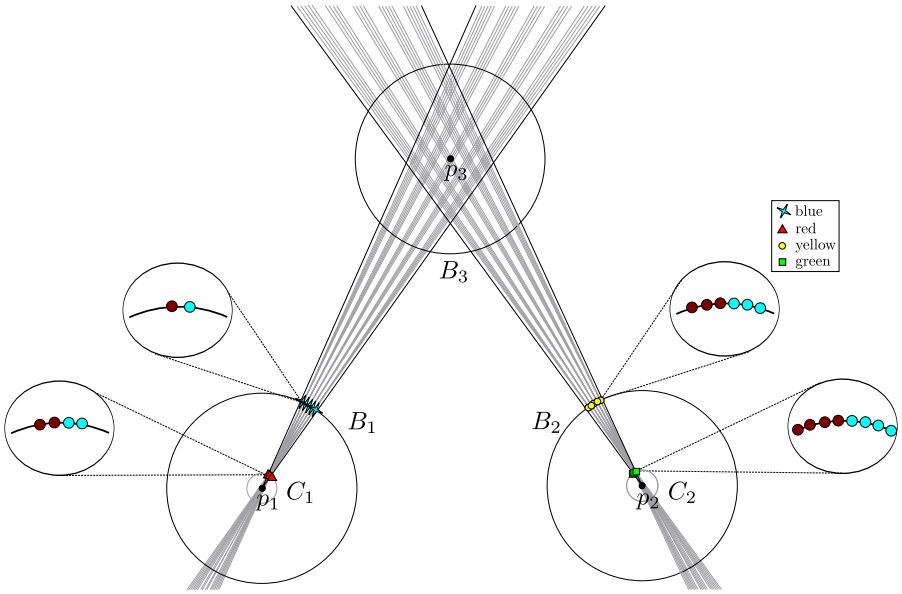


FIGURE 1. A bi-colored set with $\Omega(n^4)$ different colored radial orderings.

We start by constructing a four-colored set of points P' with $\Theta(n^4)$ distinct colored radial orderings; afterwards we replace each point of a given color with a suitable "pattern" of red and blue points. These four patterns are chosen so that, if they appear consecutively in a radial ordering, then any other equivalent radial ordering must match patterns of the same type. Since the patterns behave like the original four colors, the new set also has $\Theta(n^4)$ colored radial orderings.

Let B_1, B_2 and B_3 be three balls of radius $1/4$, whose centers p_1, p_2 and p_3, are the vertices of an equilateral triangle of side length equal to one. Choose $\varepsilon, \alpha > 0$. Let C_1 and C_2 be circles of radius $\varepsilon > 0$ centered at p_1 and p_2, respectively. Let γ_1 and γ_2 be infinite wedges of angle α, with apices p_1 and p_2 respectively. Assume that γ_1 is bisected by the line segment joining p_1 and p_3, while γ_2 is bisected by the line segment joining p_2 and p_3. Refer to Figure 1. Let m and r be the only natural numbers such that $n = 10m + r$ and $0 \leq r \leq 9$. Divide γ_1 with $m - 1$ infinite rays emanating from p_1 and intersecting B_3, such that the angle between two consecutive rays is α/m. Do likewise for γ_2, with $m - 1$ infinite rays emanating from p_2. At every point of intersection of these rays with the boundary of B_1, place a blue point; at every point of intersection with C_1 a red point; at every point of intersection with the boundary of B_2 a yellow point; finally at every point of intersection with C_2 a green point. Note that neither the rays emanating from p_1 intersect B_2 nor the rays emanating from p_2 intersect B_1. Thus $4(m-1)$ colored points are placed in total. We omit the proof that if α and ε are small enough, the number of colored radial orderings of P' as seen from observation points inside B_3 is at least $\Omega(n^4)$.

To construct P, we replace the points in P' by patterns of red and blue points, in such a way that if the colored radial orderings with respect to two points in B_3 are different, they remain different afterwards. The points in the patterns replacing a point $q \in S'$ are placed consecutively in the same circle containing q. If these points are placed close enough to q, then they will appear consecutive in the colored radial orderings with respect to every observation point in B_3. The points of S' are replaced in the following way: every

blue point with a pattern of one red and one blue point; every red point with a pattern of two red and two blue points; every yellow point with a pattern of three red and three blue points; and every green point with a pattern of four red and four blue points. Refer to Figure 1. Note that our choice of patterns implies that two equivalent radial orderings must match patterns of the same type. Also, if necessary, we may assume that α is small enough so that this augmented set is in general position. The remaining points can be placed in such a way that the colored radial orderings does not decrease. □

Theorem 2.4. $\tilde{g}(n) \leq O(n^2)$.

Proof (sketch). For simplicity assume that n is even. We employ a technique similar to the one used in the proof of Theorem 2.3. We start with a set S' of $n/2$ points, placed evenly in the unit circle. All the points of S' have the same color and thus the colored radial orderings of S' are all equivalent. Afterwards, we replace each point of S' with a symmetric pattern of red and blue points. This is done in such a way that the new number of distinct colored radial orderings increases at most to $O(n^2)$. We replace each point of S' with a pattern of "red, blue, blue, red" points, placed clockwise consecutively on the circle. If necessary, the points are perturbed to avoid degeneracies. It can be then shown that, if each pattern is placed close enough to the original point, the new set has at most $O(n^2)$ colored radial orderings. □

Theorem 2.5. $\tilde{g}(n) \geq \Omega(n)$.

Proof (sketch). Due to lack of space, we only give a very broad sketch of the main ideas of the proof. Let $p \in S$ be a red point. Let C be a circle centered at p. If the radius of C is chosen small enough, then the colored radial orderings as seen from every point in C are obtained by considering the colored radial ordering of $S \setminus \{p\}$ with respect to p and then inserting p between every pair of consecutive elements. Careful analysis using the fact that $S \setminus \{p\}$ has one more blue point than red points enable us to show that the number of distinct radial orderings as seen from observation points in C is at least $n/2$. □

3 Conclusions

Based on our experience, we make the following conjectures:

Conjecture 3.1. $g(n) = \Omega(n^4)$.

Conjecture 3.2. $\tilde{g}(n) = \Omega(n^2)$.

References

[1] Linda Deneen and Gary Shute. Polygonizations of point sets in the plane. *Discrete Comput. Geom.*, 3(1):77–87, 1988.

[2] B. Tovar, L. Freda, and S. M. LaValle. Using a robot to learn geometric information from permutations of landmarks. In *Topology and Robotics*, volume 438 of *Contemp. Math.*, pages 33–45. Amer. Math. Soc., Providence, RI, 2007.

Measuring regularity of convex polygons: experimental results

Ramon Chalmeta[1], **Vera Sacristán**[2], **Maria Saumell**[2]

[1] Facultat d'Informàtica de Barcelona, Universitat Politècnica de Catalunya, Barcelona, Spain
rchalmeta@gmail.com

[2] Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, Barcelona, Spain
vera.sacristan@upc.edu, maria.saumell@upc.edu

Abstract. We aim to evaluate to which extent the shape of a given convex polygon is close to be regular, focusing on diverse characteristics of regularity: optimal ratio area-perimeter, equality of angles and edge lengths, regular fitting, angular and areal symmetry. We have designed and implemented algorithms to compute the resulting measures and we provide and discuss experimental results on a large set of polygons to illustrate their behavior as detection and as quality control tools.

Introduction

Regular polygons are interesting for pure mathematical purposes, but also because they appear in nature, as well as in a wide spectrum of human made objects such as traffic signals, furniture, coins, tools and architecture. As a consequence, some methods to detect regular polygons in given images have been developed in recent years [**2, 7**]. Some other works have focused on detecting regular polygons within sets of points [**1**].

Our approach is related to the aforementioned works, although in our case the goal is to evaluate to which extent a given convex polygon resembles a regular polygon. Our work, hence, is particularly related to applications in metrology, automatic shape recognition, and discretization of planar domains into suitable meshes. In the three cases, the measurements we present can be seen as quality control tests to check regularity.

In [**6, 8**] we proved the correctness of our algorithms and analyzed their asymptotic behavior, and in [**3, 6**] we associated to each algorithm a measure guaranteed to be correctly normalized. We now show and evaluate our experimental results [**3, 4, 5**], which is the goal of this paper.

1 Preliminaries

Our measures are based on a range of different aspects of regularity, such as optimal ratio area-perimeter, equality of angles and edge lengths, regular fitting, angular and areal symmetry. For the sake of readability, in this section we summarize the definitions and theoretical results that can be found in detail in [**5**].

Following the rational from [**9**], we require several conditions on the measures, to ensure that they are well defined: *i*) the regularity measure is a number in $(0, 1]$; *ii*) the regularity measure of a given polygon P equals 1 if and only if P is regular; *iii*) there exist polygons whose regularity measure is arbitrarily close to 0; *iv*) the regularity measure of a polygon is invariant under similarity transformations.

[2]Partially supported by projects MTM2009-07242 and Gen. Cat. DGR 2009SGR1040.

Let P be any convex n-gon.

Definition 1.1. Let R be the regular n-gon with the same perimeter as P. We define $\mu_1(P) = \frac{\text{area}(P)}{\text{area}(R)}$.

Definition 1.2. Consider the polygon P' having the same ordered sequence of edge lengths as P, and all its vertices on a circle, and let R be the regular n-gon whose vertices lie on the same circle, and such that each vertex of P' is matched to one of R as to minimize either the maximum angular distance or the sum of angular distances between matched vertices. We define $\mu_2(P) = \mu_2^1(P)\,\mu_2^2(P)$, and $\mu_3(P) = \mu_3^1(P)\,\mu_3^2(P)$, where $\mu_2^1(P) = \mu_3^1(P) = 1 - \frac{\alpha(P,P')}{\pi}$, and $\alpha(P,P')$ is the absolute value of the maximum difference between each interior angle of P and its corresponding interior angle in P'; $\mu_2^2(P) = 1 - \frac{\max_i d_i}{\pi - \pi/n}$; and $\mu_3^2(P) = 1 - \frac{2\sum_{i=1}^n d_i}{n\pi}$, if n is even, and $\mu_3^2(P) = 1 - \frac{2n\sum_{i=1}^n d_i}{(n^2-1)\pi}$ if n is odd, where d_i are the above mentioned optimal angular distances.

Definition 1.3. Let l_i and φ_i respectively denote the lengths of the edges of P and its interior angles. We associate to P the point $X_P = \left(\frac{l_1}{\sum_{i=1}^n l_i}, \ldots, \frac{l_n}{\sum_{i=1}^n l_i}, \varphi_1, \ldots, \varphi_n\right)$ in $\mathbb{R}^{2n}$. Then, $\mu_4(P) = 1 - \frac{d(X_P,\ell)}{1+\left(4-\frac{8}{n}\right)\pi}$, if $n \geq 4$, and $\mu_4(P) = 1 - \frac{d(X_P,\ell)}{\frac{1}{2}+\frac{4\pi}{3}}$, if $n = 3$, where d is the L_1 distance, and ℓ is the half-line of all regular n-gons in $\mathbb{R}^{2n}$.

Definition 1.4. We define $\mu_5(P) = \frac{\text{area}\,(P)}{\text{area}\,(R_E)}$, $\mu_6(P) = \frac{\text{area}\,(R_I)}{\text{area}\,(P)}$, and $\mu_7(P) = \frac{\text{area}\,(R_I)}{\text{area}\,(R_E)}$, where R_E and R_I are respectively the minimum enclosing and the maximum enclosed regular m-gon for P.

Definition 1.5. Let $a_i(\varphi)$ be the area of P intercepted by the i-th wedge of an angularly equally distributed pencil of n half-lines which can rotate around the barycenter of P. We define $\mu_8(P) = \frac{n \min_i a_i}{\text{area}\,(P)}$, $\mu_9(P) = \frac{5\,\text{area}\,(P) - 9\max_i a_i}{(5n-9)\max_i a_i}$, and $\mu_{10}(P) = \frac{\min_i a_i}{\max_i a_i}$, where the angle values respectively minimize $\min_i a_i$, maximize $\max_i a_i$, and maximize the difference $\max_i a_i - \min_i a_i$.

Definition 1.6. Consider the point $q \in P$ minimizing the maximal triangular area qp_ip_{i+1}; maximizing the minimal one, or minimizing the maximal difference. Let $d_{\min} = \min_i d(q,p_i)$, and $\alpha_{\min} = \min_i \angle p_iqp_{i+1}$ (analogously for $d_{\max}$ and $\alpha_{\max}$). Then μ_{11}, μ_{12}, and μ_{13} are defined as $\sqrt{\frac{d_{\min}}{d_{\max}}\frac{\alpha_{\min}}{\alpha_{\max}}}$ for the three cases.

Theorem 1.1. *The measures μ_5, μ_6, and μ_7 satisfy conditions i), iii), and iv). If $n = m$, they also satisfy ii). All remaining measures satisfy conditions i)–iv).*

Theorem 1.2. *The value of $\mu_1(P)$ can be computed in $O(n)$ time. The values of $\mu_2(P)$ and $\mu_3(P)$ can be computed in $O(n\log n)$ time. The value of $\mu_4(P)$ can be computed in $O(n)$ time. The values of $\mu_5(P)$, $\mu_6(P)$, and $\mu_7(P)$ can be respectively computed in $O\big(\min\{nm^2\log m, (n+m)^2\}\big)$, $O(n^2+m)$ and $O\big(\min\{nm^2\log m, (n+m)^2\} + n^2 + m\big)$ time. The values of $\mu_8(P)$, $\mu_9(P)$, and $\mu_{10}(P)$ can be computed in $O(\lambda_6(n)\log n)$ time, where $\lambda_s(k)$ denotes the maximal complexity of the upper or lower envelope of a set of k curves which pairwise intersect at most s times. The values of $\mu_{11}(P)$, $\mu_{12}(P)$, and $\mu_{13}(P)$ can be computed in $O(n)$ time.*

It is worth mentioning that the complexity results hold in an extended Real RAM model of computation, since in some cases the optimization algorithm or the computation of the measure requires numerically solving a transcendent equation or computing an angle or a square root.

2 Experimental results

We have generated 553 different polygons to apply the measures to, of which 73 triangles, 70 quadrilaterals, 67 pentagons, 73 hexagons, 73 heptagons, 73 octagons, and 31 of each 12-, 25-, 50- and 100-gons. As for their shapes, the experiments included 222 pseudo-random polygons; 117 regular polygons with small errors, as well as with aligned vertices and/or with chopped vertices; 136 deformed regular polygons of all sorts (by scaling the y-coordinates of their vertices, by translating one vertex, by slanting all the vertices...), as well as 36 almost degenerate polygons; the remaining being special cases for polygons with a small number of vertices.

The experiments, described in detail in **[3, 5]**, confirm that all proposed measures reflect human intuition on regularity of polygons. The top row in Figure 1 illustrates this behavior on pseudo-randomly generated triangles (left) and on deformed equilateral octagons (right).

Measures based on the same regularity characteristic tend to give very similar results, as could be expected. This is the case for μ_2 and μ_3, for μ_5, μ_6 and μ_7, and so on. Measures based on different regularity characteristics give somehow different absolute values, although their relative behavior is consistent and they only substantially differ for highly degenerate polygons.

Specific characteristics of the different measures can be exploited in choosing the most appropriate measure for each goal and the characteristics of the polygons it will be applied to. In this sense, we find particularly interesting to mention their behavior as quality control tools. In order to offer experimental results on this respect, we have adapted the ISO norms for manufacturing cylinders to the case of regular polygons and we have run our measures on different levels of perturbed regular polygons. For high precision experiments, we have randomly perturbed one or all the vertices of regular polygons with an error within $[0.00083\%, 0.0050\%]$. Precision examples had errors bounded within $[0.00500\%, 0.0188\%]$; fine manufacturing within $[0.01833\%, 0.0438\%]$; general manufacturing within $[0.04500\%, 0.1088\%]$; and, finally, forgery errors were bounded within $[0.11667\%, 2.7500\%]$. The results indicate that all but one of our measures detect even high precision errors (the definition of μ_1 is too loose for such a task), and that measures μ_{11}, μ_{12}, and μ_{13} are particularly good at detecting them, as the second row of Figure 1 illustrates.

Another relevant application of our proposed measures is related to shape recognition. Our experiments show that measures μ_5, μ_6 and μ_7 are very efficient at detecting the cases where a convex n-gon resembles a regular m-gon, no matter the values of n and m. The third row of Figure 1 illustrates the result for a danger and a stop traffic signs. In the examples shown, the vertices of the input polygon are imprecise, in the sense that they do not form an exact regular polygon (triangle), or the polygon has more than the apparent number of vertices (octagon).

From the experiments we conclude that measure μ_1 is too loose for some purposes, although it reflects human perception of regularity as closely related to roundness. Measure μ_4 is probably at its best for n-gons with large n. Measures μ_5, μ_6 and μ_7 have a good general performance and, in addition, they are particularly interesting for automatic regularity recognition when the number of vertices of the polygon is unknown. Quality control, especially when high precision is required, is particularly well served by measures μ_{11}, μ_{12} and μ_{13}. The remaining measures, μ_1 and μ_2, as well as μ_8, μ_9 and μ_{10}, are, from our viewpoint, good multipurpose measures.

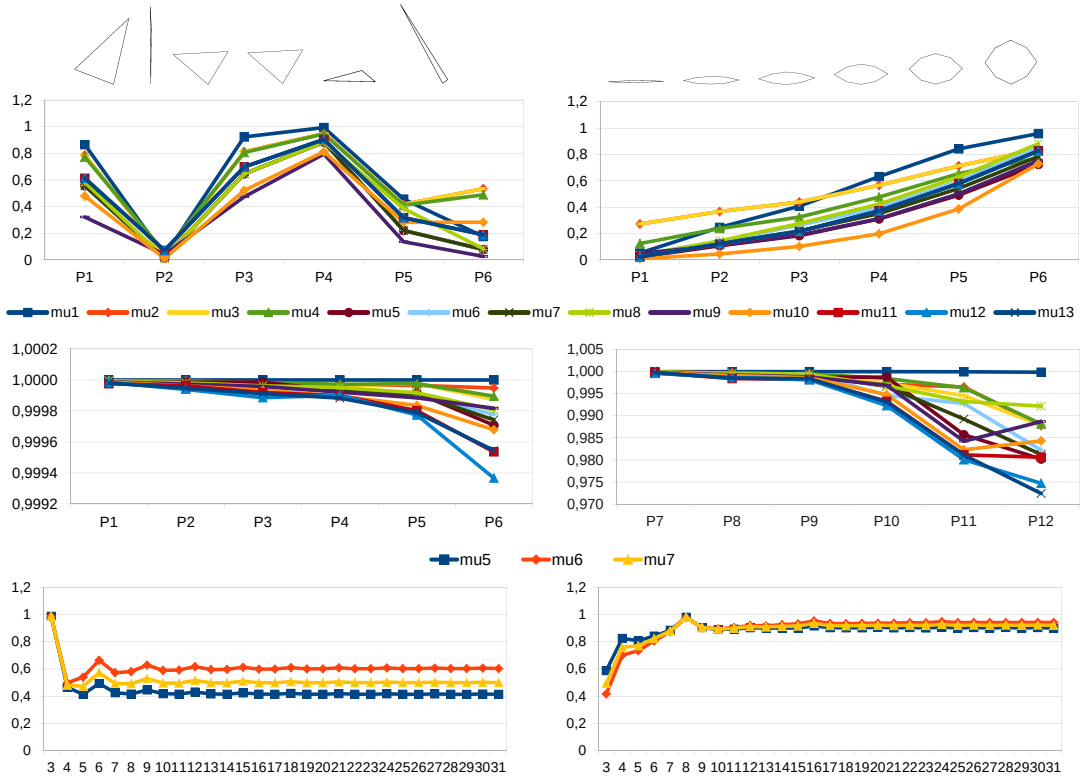


FIGURE 1. Row 1: Measuring regularity of pseudo-random triangles (left) and deformed equilateral octagons (right). Row 2: Quality control, from high precision (leftmost) to forgery (rightmost). Row 3: Detecting the shape of an equilateral triangle (left) and a regular octagon (right) from imprecise coordinates and number of vertices.

References

[1] G. Aloupis, J. Cardinal, S. Collette, J. Iacono and S. Langerman, Where to build a temple, and where to dig to find one, *Proc. European Workshop on Computational Geometry* (2006), 1–4.

[2] N. Barnes, G. Loy, D. Shaw and A. Robles-Kelly, Regular polygon detection, *Proc. International Conference on Computer Vision ICCV'05* **1** (2005), 778–785.

[3] R. Chalmeta, *Mesures de regularitat per a polígons convexos* (in Catalan), Master Thesis in Computer Science, Universitat Politècnica de Catalunya, 2010.

[4] R. Chalmeta, F. Hurtado, V. Sacristán and M. Saumell, Regularity measures for convex polygons, http://www-ma2.upc.es/∼vera/RegMeas/.

[5] R. Chalmeta, F. Hurtado, V. Sacristán and M. Saumell, Measuring regularity of convex polygons, submitted to *IEEE Transactions on Image Processing*, 2011.

[6] F. Hurtado, V. Sacristán and M. Saumell, Some regularity measures for convex polygons, *Proc. 25th European Workshop in Computational Geometry* (2009), 125–129.

[7] G. Piccioli, E. D. Micheli, P. Parodi and M. Campani, Robust method for road sign detection and recognition, *Image and Vision Computing* **14(3)** (1996), 209–223.

[8] M. Saumell, *Mesures de regularitat per a polígons convexos* (in Catalan), Master Thesis in Mathematics, Universitat Politècnica de Catalunya, 2008.

[9] J. Žunić and P. L. Rosin, A new convexity measure for polygons, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **26(7)** (2004), 923–934.

Multiple range searching on the GPU

Marta Fort[1], **J. Antoni Sellarès**[1]

[1] Universitat de Girona
mfort@ima.udg.edu, sellares@ima.udg.edu

Abstract. Multiple disk/ball range searching queries in 2d/3d Euclidean space are solved in parallel using a uniform grid under CUDA architecture. Experimental results of our implementation are presented.

Introduction

Let S be a set of n points and $\mathcal{R}$ be a family of subsets in 2d/3d Euclidean space. The sets of $\mathcal{R}$ are called *ranges*. In a *range searching query problem*, we want to preprocess S into a data structure so that, for a query range $R \in \mathcal{R}$, the points in S can be counted or reported efficiently. Typical examples of ranges include axis-parallel rectangles/rectangular parallelepipeds and disks/balls; see the survey by Agarwal and Erickson [**1**]. In a *multiple range searching query problem*, instead of a unique range $R \in \mathcal{R}$ we have a subfamily $\mathcal{R}'$ of ranges of $\mathcal{R}$ and we want to solve a range searching query for each range $R \in \mathcal{R}'$. Applications of multiple range searching queries could be found in point cloud matching, traveler information systems and biological sequence similarity analysis.

A *uniform grid* subdivides a bounding box of the 2d/3d point set S into a set of regular axis parallel rectangles/rectangular parallelepipeds, called *cells*. Each cell in the grid is referenced by (row, column)/(row, column, floor) of the cell in the grid and stores the list of points of S that contains. The uniform grid is one of the simplest data structures used to solve proximity queries and other geometric problems. It has been experimentally shown that, for many applications, uniform grids are efficient for both evenly and unevenly spaced data and they are appropriate for parallel processing [**2**].

The advances in GPU (Graphics Processing Unit) hardware design, together with CUDA (Compute Unified Device Architecture) and some programming languages such as OpenCL (Open Computing Language), make GPUs attractive to solve problems in a parallel way. The parallelizable parts of an algorithm are executed by a collection of threads running in parallel —these threads are grouped into 1D, 2D or 3D blocks of user defined size which are also processed in parallel. The instructions to be executed by each thread are written in a kernel, where different types of memory can be used. Global memory is the biggest but slowest access time memory; it is accessible by every thread and is visible from the CPU. Shared memory is a fast memory shared between all the threads in a block to co-operate. Registers are the fastest memory and store local variables of each single thread. Some functions which are read-modify-write atomic operations can be used; 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. For further details, see [**4**].

In this paper we present fast and scalable GPU algorithms designed under CUDA architecture for: *a*) constructing uniform grids; *b*) solving multiple range counting and reporting queries with the support of a uniform grid structure. Experimental results of our implementation in OpenCL are provided.

[1]Partially supported by Spanish MCI research grant TIN2010-20590-C02-02.

1 Uniform grid

To represent a uniform grid of size $G = H \times W/H \times W \times Z$ over an axis-parallel rectangle/rectangular parallelepiped containing the points of S, we use a grid vector v_S and two integer matrices, the counting grid C and the positioning grid P. The grid vector v_S stores the points of S and the counting grid C provides the number of points contained in each cell. The points of a cell are stored sequentially in the grid vector v_S starting at the position indicated by the positioning grid P. See Figure 1 for a 2d example. Our approach is similar to one proposed in [**3**]; however, in our case we also store the counting grid C, even though it is not strictly necessary since it can be computed from the positioning grid P. The uniform grid is built in three steps that compute: 1) the counting grid; 2) the positioning grid; 3) the grid vector.

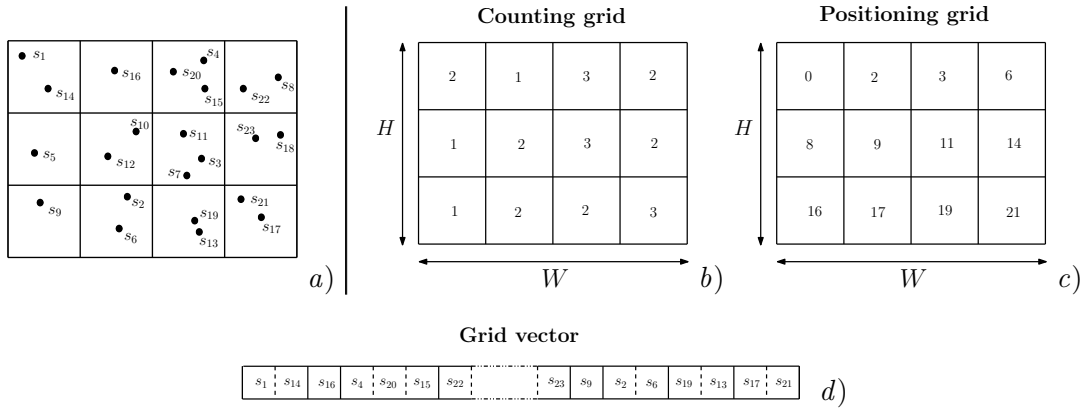


FIGURE 1. *a*) A set S of 2d points on a uniform grid; *b*) the counting grid; *c*) the positioning grid; and *d*) the grid vector.

All the 1D-arrays we use in the presented algorithms are stored in GPU global memory. The points of S and the grid vector v_S are stored in the GPU in 1D-arrays of size n, denoted s and v. According to the CUDA memory model, the counting matrix C and the positioning matrix P are linearized by rows/(rows, columns) and are stored in 1D-arrays of size G, that we denote c and p.

Counting grid. Points of S are transferred to the GPU and stored in s and the 1D-array c used to store the counting grid is initialized to 0s. Then a thread per point in S is run. The thread with global identifier i reads from the 1D-array s the point $s(i)$ of S, determines the grid position j it belongs to and increments the counting value $c(j)$ by one using an atomic incremental function.

Positioning grid. The positioning value $p(i)$ is obtained as $\sum_{j=0}^{i-1} c(j)$ by using the exclusive scan (prefix sum) algorithm of [**5**].

Grid vector. We use an auxiliary 1D-array a of size G initialized to 0s. The value $a(j)$ represents the number of points that have already been located at the grid cell of linearized index j. We run n threads one per point of S. The thread with global identifier i reads from the 1D-array s the point $s(i)$ of S and computes the linearized position j of the cell it corresponds to in the grid. Then, by using an atomic incremental function, stores point s_i in the position $p(j) + a(j)$ of the 1D-array v and increments $a(j)$ by one. An atomic function reads the value that will return, operates on it and stores the result, meanwhile, no other access to that memory position is allowed.

Complexity analysis. The total number of operations done to obtain the counting grid is $O(n)$, the positioning grid $O(G)$, and the grid vector $O(n)$. Thus, building the grid structure has a total work of $O(n + G)$.

2 Range searching queries

Given a set $\mathcal{R} = \{R_1, \ldots, R_m\}$ of m disk/ball ranges where range R_i is defined by the center q_i and a radius r_i, we answer multiple range searching queries in parallel. We distinguish between range counting and reporting queries. The presented solutions are integrally obtained in the GPU by transferring the range centers and radii to the GPU 1D-arrays q and r, respectively, and considering the grid containing S stored and built in the GPU.

Range counting. The range counting solution is stored in a 1D-array r_c of size m; $r_c(i)$ stores the number of points of S contained in range R_i. We run m threads, one per range. Thread of global identifier i reads from global memory $q(i)$ and $r(i)$ and explores a square/cube centered at $q(i)$ and of side size defined by $2r(i) + 1$ cells. Notice that the number of grid cells intersected by the square/cube depends on the grid and bounding box dimensions. Each thread traverses its squared/cubic corresponding region in a row/(row, column) order. By using a local counter, the number of points contained in R_i is obtained. Once all the cells have been traversed, the number of obtained points is stored in global memory in $r_c(i)$. When all the threads have finished, r_c contains the range counting answer.

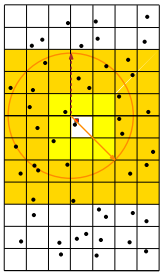


FIGURE 2. Points on a grid, a red query point and its radius.

Range reporting. The range reporting problem cannot be solved in one single step because its answer size is not known in advance and it is not possible to dynamically allocate memory in the GPU. The answer is given by two 1D-arrays r_p and r_r. Points contained in range R_i are stored in consecutive positions of r_r starting at position $r_p(i)$.

We start by solving the range counting problem. Next, an exclusive scan prefix sum is run on the range counting answer to obtain r_p. The exclusive scan is used also to store the total number M of obtained points. This value is read from the CPU and memory in the GPU to store the array r_r containing the M points is allocated. Finally, the reporting problem is solved by using the arrays of centers q, radii r, positions r_p and the grid v, c and p. Again a thread per query point is run. Thread of global identifier i reads $q(i)$ from global memory and computes the linearized position j of the grid cell where $q(i)$ is contained. The squared/cubic region is revisited and the indices on v_s of the points contained in R_j are stored in consecutive positions of r_r, starting at $r_p(j)$.

Observe that, if we are interested in obtaining the range searching solutions in the CPU, the grid vector v and the arrays r_p and r_r have to be read back from GPU to CPU.

Complexity analysis. Let I be the number of points contained in the cells intersected by the ranges in $\mathcal{R}$, and M the maximal size of the range searching output. The global work of the range counting is $O(I)$. To answer the range reporting problem, the total number of operations is $O(I)$ for the range counting, $O(m)$ for the scan algorithm, and $O(I + M)$ to store the points in the range. Thus, the total work complexity of the range reporting is $O(m + I + M)$.

3 Experimental results

The results of Table 1 are obtained under our implementation using a Intel Core 2 CPU 2.13 GHz, 2 GB RAM and a GPU NVidia GeForce GTX 480 which has a cached global memory, reducing the access to global memory problems and time. Points are randomly generated; grids are chosen with $H = W(= Z)$, and their discretization size is chosen after testing different values. We choose the same radius value for all the ranges. The considered radii are related to d_m, which is the median of the smallest distance between the randomly generated points of S. This value is commonly used in different experimental problems related to point clouds.

				Range searching times (ms)					
			Grid	m	100		10,000		1,000,000	
	n	H	*comp. (ms)*		d_m	$4d_m$	d_m	$4d_m$	d_m	$4d_m$
2d	10^2	16	0.39		0.24	0.32	1.57	1.92	122.3	147.62
	10^4	96	0.77		0.29	0.31	1.91	2.01	147.52	148.61
	10^6	160	13.10		1.24	1.26	8.18	8.23	701.46	104.57
3d	10^2	8	0.34		0.36	0.64	2.42	7.85	171.83	579.25
	10^4	16	0.74		0.46	0.96	2.50	8.13	205.58	692.61
	10^6	64	20.91		0.52	1.61	3.52	12.39	293.76	1077.31

TABLE 1. Computational times in *miliseconds* obtained as the median of 10 executions.

The time needed to transfer the points of S from the CPU to the GPU for our 2d examples are 0.004 ms, 0.05 ms and 10.1 ms, and for the 3d examples become 0.01 ms, 0.14 ms and 23.2 ms. Concerning the transferring times of the solution from the GPU to the CPU, for the 2d case, considering $n = m = 10^2$ and $r = 4d_m$, the about $1.2 \cdot 10^3$ obtained points are transferred in 0.22 ms. For $m = n = 10^6$ and $r = 4d_m$, the about $13 \cdot 10^6$ obtained points are transferred in 103.06 ms. Considering 3d points, in the former case more than $4.2 \cdot 10^2$ points are transferred in 0.28 ms and in the latter more than $45 \cdot 10^6$ points are transferred in 328.42 ms.

For bigger radius or values of n and m, when the answer size is too big, the reporting problem has to be subdivided into different subproblems in order to be solved. However, the presented results are obtained without subdivision needs.

References

[1] P. K. Agarwal and J. Erickson, Geometric Range Searching and Its Relatives, Advances in Discrete and Computational Geometry (1997), 1–56.

[2] V. Akman, W. R. Franklin, M. Kankanhalli, C. Narayanaswami, Geometric Computing and the Uniform Grid Data Technique, *Computer Aided Design*, Volume 21, Issue 7 (1989), 410–420.

[3] S. Green, Particle Simulation using CUDA. URL http://developer.download.nvidia.com/compute/cuda/sdk/website/C/src/particles/doc/particles.pdf, last accessed April 2011.

[4] NVIDIA Corporation, OpenCL programming guide for the CUDA architecture. URL http://www.nvidia.com/object/cuda_opencl_new.html, last accessed April 2011.

[5] Shubhabrata Sengupta, Mark Harris and Michael Garland, Efficient Parallel Scan Algorithms for GPUs NVIDIA, 2008, Technical Report NVR-2008-003.

Metaheuristic approaches for MWT and MWPT problems

Maria Gisela Dorzán[1], Edilma Olinda Gagliardi[1], Gregorio Hernández Peñalver[2], Mario Guillermo Leguizamón[1]

[1] Facultad de Ciencias Físico Matemáticas y Naturales, Universidad Nacional de San Luis, San Luis, Argentina
{mgdorzan,oli,legui}@unsl.edu.ar

[2] Facultad de Informática, Universidad Politécnica de Madrid, Madrid, España
gregorio@fi.upm.es

Abstract. It is known that the Minimum Weight Triangulation problem is NP-hard. Also the complexity of the Minimum Weight Pseudo-Triangulation problem is unknown, yet it is suspected to be also NP-hard. Therefore we focused on the development of approximate algorithms to find high quality triangulations and pseudo-triangulations of minimum weight. In this work we propose two metaheuristics to solve these problems: Ant Colony Optimization (ACO) and Simulated Annealing (SA). For the experimental study we have created a set of instances for MWT and MWPT problems, since no reference to benchmarks for these problems were found in the literature. Through experimental evaluation, we assess the applicability of the ACO and SA metaheuristics for MWT and MWPT problems. These results are compared with those obtained from the application of deterministic algorithms for the same problems (Delaunay Triangulation for MWT and a Greedy algorithm respectively for MWT and MWPT).

Introduction

In Computational Geometry there are many optimization problems that are either NP-hard or such that no polynomial algorithms are known to solve them. Examples of these optimization problems are those related to special geometric configurations, such as *triangulations* and *pseudo-triangulations*, which are interesting to investigate due to their use in many fields of application.

Minimizing the total length has been one of the main optimality criteria for triangulations and pseudo-triangulations. Indeed, the Minimum Weight Triangulation (MWT) and the Minimum Weight Pseudo-Triangulation (MWPT) minimize the sum of edge lengths, providing a quality measure for determining how good is a structure. The complexity of computing a minimum weight triangulation has been one of the most longstanding open problems in Computational Geometry, introduced by Garey and Johnson in their open problems list, and various approximation algorithms were proposed over time. It has been the subject of numerous investigations for identifying criteria to include certain edges of the MWT. The paper **[2]** presents the implementations of the LMT heuristic for computing MWT problems, that can compute the "exact" MWT for well-behaved point sets very fast. It is also worth noticing that, to our best knowledge, there are no reports about demonstration of such results. Aichholzer et al. **[1]** disprove the conjecture that the LMT-skeleton coincides with the intersection of all locally minimal triangulations, LMT(P), being P a polygon in the plane, even for a convex polygon. Later, Mulzer and Rote proved in 2006 that MWT construction is an NP-hard problem **[12]**.

The complexity of the MWPT problem is unknown. However, Levcopoulos and Gudmundsson [**8**] show that a 12-approximation of an MWPT can be computed in $O(n^3)$ time. They give an $O(\log n \cdot w(MST))$ approximation of an MWPT, in $O(n \log n)$ time, where $w(MST)$ is the weight of the minimum Euclidean spanning tree, which is a subset of the obtained structure.

Given the inherent difficulty of the above mentioned problems, the approximate algorithms arise as alternative candidates for MWT and MWPT problems. These algorithms can obtain approximate solutions to the optimal solutions, and they can be specific for a particular problem or they can be part of a general applicable strategy in the resolution of different problems. The metaheuristic methods satisfy these properties. These algorithms have a simple implementation and they can efficiently find good solutions for NP-hard optimization problems [**11**]. In this work we use the *Ant Colony Optimization* (ACO) [**4**] and *Simulated Annealing* (SA) [**3, 9, 10**] metaheuristics, and we compare them with Delaunay Triangulation and Greedy algorithms for triangulations and pseudo-triangulations. Previous works about approximations on MWT and MWPT problems using the ACO and SA metaheuristics, were presented in [**5, 7**]. It is also worth noticing that, to the best knowledge of the authors, there are no reports in the literature of extensive experimental evaluations using exact algorithms or metaheuristic techniques.

This paper is organized as follows. In the next section, we review some theoretical aspects of ACO and SA metaheuristics applied to MWT and MWPT problems. Section 2 resumes the experimental evaluation. The last section is devoted to conclusions and future works.

1 ACO and SA metaheuristics applied to MWT and MWPT problems

The ACO metaheuristic involves a family of algorithms in which a colony of artificial ants cooperate in finding good solutions to difficult discrete optimization problems [**4**]. An artificial ant in an ACO algorithm is a stochastic constructive procedure that incrementally builds a solution by adding opportunely defined solution components to a partial solution under construction. Therefore, the ACO metaheuristic can be applied to any combinatorial optimization problem for which a constructive graph can be defined. For details of designs, implementations, and parameter settings of the ACO-MWT and ACO-MWPT algorithms, see [**5, 7**].

The SA metaheuristic tries to minimize the limitation of the local search algorithms, which stops as soon as they find a local extreme. For that, it is allowed to accept solutions of worse quality than the current solution with a certain probability. This probability depends on a parameter T, called temperature, which decreases over the algorithm iterations according to a decrement rule [**3**]. In regards of the design of the algorithms, the parameter settings and the details of implementations for SA-MWT and SA-MWPT algorithms; see [**6**]. SA2P-MWPT is an improved version of SA-MWPT, which involves a double pass under certain criteria, considering the best results obtained in some temperatures.

2 Experimental evaluation

To the best knowledge of the authors, there do not exist collections of instances in the literature for MWT and MWPT problems. Consequently, no benchmarking data are publicly available that allow us to compare our proposal with some other algorithm previously studied. According to that, we design an *instance generator*. Therefore, we have generated respectively a collection of 10 instances of size 40/80/120/160/200; i.e., a total of 50 problem instances. Each instance is called LDn-i where n denotes the size of the i-instance, with $1 \leq i \leq 10$. The instance generator uses different functions of the CGAL Library. The points are randomly generated, uniformly distributed, and, for each point (x, y), the coordinates x, y are in $[0, 1000]$. For the purpose of this work, we assume that there are non-collinear points. The proposed algorithms were implemented in C language and run on a BACO parallel cluster.

In Table 1 we show the results for ACO-MWT, SA-MWT, Greedy Triangulation (GT), and Delaunay Triangulation (DT). The best results were obtained with the SA-MWT algorithm using local retriangulation neighborhood and with different temperature decrement rules (*FSA*, *VFSA*, and geometric decrement with $\alpha = 0.8$).

Especially for pseudo-triangulations, the Greedy Pseudo-Triangulation (GPT) algorithm builds a pseudo-triangulation starting with one face. This face has the edges obtained by the convex hull of the point set P, i.e., $CH(P)$. For the solution construction, the P set is partitioned in faces. This process finishes when all faces are pseudo-triangles without interior points. A face is divided in two faces when there are interior points, or is not a pseudo-triangle. Thus, the partition can be done if i) there are at least one interior point and two points in the border; or ii) there is no interior point, so two points located on the border are chosen. Such selection is performed by selecting those points that generate the edges lead to local minimum weight.

	ACO-MWT	*SA-MWT*	*GT*	*DT*
LD40-1	5493047	**5463745**	5477181	5666348
LD40-2	4661242	**4659552**	**4659552**	4722381
LD40-3	5502567	**5478923**	5489487	5663032
LD40-4	**5745772**	**5745772**	5751867	6289829
LD80-1	6242505	**6220029**	6231682	6462038
LD80-2	7605383	**7581868**	**7581868**	8081573
LD80-3	5836037	**5828344**	5845506	6143637
LD80-4	6217040	**6147234**	**6147234**	6460311

TABLE 1. MWT: The best (smallest) weights obtained with the mentioned algorithms for sets of 40 and 80 points.

Table 2 shows the results according to the smallest weights obtained using the ACO-MWPT, SA-MWPT, SA2P-MWPT and GPT algorithms. The best results for the SA-MWPT and SA2P-MWPT algorithms were obtained with edge-flip neighborhood and with the *FSA* temperature decrement rule.

ACO algorithms were first used to show that the results from Greedy algorithms and Delaunay Triangulation can be improved. Other preliminary results seem to indicate that the SA and SA2P algorithms are more effective techniques for MWT and MWPT. For more details or an extended version of this work, please refer to the authors.

	ACO-MWPT	*SA-MWPT*	*SA2P-MWPT*	*GPT*
LD40-1	6115636	5817042	**4181914**	5312131
LD40-2	4442710	4778701	**3384958**	4292347
LD40-3	5684342	6391410	**4169242**	5794018
LD40-4	5627098	5575409	**5047483**	6245196

TABLE 2. MWPT: The best (smallest) weights obtained with the mentioned algorithms for sets of 40 points.

3 Conclusions

In this work we present the design of approximate algorithms for solving the Minimum Weight Triangulation and Minimum Weight Pseudo-Triangulation problems. Another contribution of this research was the creation of a set of instances for the experimental evaluation, as there are no available instances with special properties for building triangulations and pseudo-triangulations. From this initial experimental phase we obtained preliminary results that will guide future experimentation. Actually, we are in the phase of applying a more methodological approach for the experimental design and running the set of instances for all sets of points mentioned. Since the metaheuristics have proven to behave very well in solving this class of NP-hard problems, there are several directions for further research. We intend to use different parameterizations of the ACO and SA algorithms to adapt and implement other metaheuristics, and to develop hybrid metaheuristics to solve the proposed problems.

References

[1] Oswin Aichholzer, Franz Aurenhammer, and Reinhard Hainz. New results on MWT subgraphs. *Inf. Process. Lett.*, 69:215–219, March 1999.

[2] Ronald Beirouti and Jack Snoeyink. Implementations of the LMT heuristic for minimum weight triangulation. In *Proceedings of the fourteenth annual symposium on computational geometry*, SCG'98, pages 96–105, New York, NY, USA, 1998, ACM.

[3] V. Černý. Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 1985.

[4] M. Dorigo and T. Stützle. *Ant Colony Optimization.* MIT Press, Cambridge, MA, 2004.

[5] M. Dorzán, E. Gagliardi, M. Leguizamón, and G. Hernández Peñalver. Globally optimal triangulations of minimum weight using ant colony optimization metaheuristic. *Journal of Computer Science & Technology*, 10, 2010.

[6] M. Dorzán, E. Gagliardi, M. Leguizamón, and G. Hernández Peñalver. Triangulaciones y pseudotriangulaciones de peso mínimo: resolución aproximada con simulated annealing. In *VII Jornadas de Matemática Discreta y Algorítmica. España*, 2010.

[7] M. Dorzán, E. Gagliardi, M. Leguizamón, M. Taranilla, and G. Hernández Peñalver. Algoritmos ACO aplicados a problemas geométricos de optimización. In *XIII Encuentros de Geometría Computacional*, 2009.

[8] J. Gudmundsson and C. Levcopoulos. Minimum weight pseudo-triangulations. *Comput. Geom.*, 38(3):139–153, 2007.

[9] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[10] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[11] Z. Michalewicz and D. Fogel. *How to Solve It: Modern Heuristics.* Springer, 2004.

[12] W. Mulzer and G. Rote. Minimum weight triangulation is NP-hard. In *Proc. 22nd Annu. ACM Sympos. Comput. Geom*, pages 1–10. ACM Press, 2006.

Solving the minimum vertex floodlight problem with hybrid metaheuristics

Antonio L. Bajuelos[1,4], **Santiago Canales**[2,5], **Gregorio Hernández**[3,5], **Mafalda Martins**[1,6]

[1] Universidade de Aveiro, Portugal
leslie@ua.pt, mafalda.martins@ua.pt

[2] Universidad Pontificia Comillas de Madrid, Spain
scanales@dmc.icai.upcomillas.es

[3] Universidad Politécnica de Madrid, Spain
gregorio@fi.upm.es

Abstract. In this paper we propose four approximation algorithms (metaheuristic based), for the Minimum Vertex Floodlight Set problem. Urrutia et al. [**9**] solved the combinatorial problem, although it is strongly believed that the algorithmic problem is $\mathcal{NP}$-hard. We conclude that, on average, the minimum number of vertex floodlights needed to illuminate a orthogonal polygon with n vertices is $\lceil \frac{n}{4.29} \rceil$.

Introduction

In this paper we address the Minimum Vertex $\frac{\pi}{2}$-Floodlight Set problem (MVF(P) problem). This problem asks for the minimum number of vertex $\frac{\pi}{2}$-floodlights necessary to illuminate a given orthogonal simple polygon P with n vertices (n-ogon, for short [**8**]). A vertex $\frac{\pi}{2}$-floodlight is a source light with an angle of illumination of $\frac{\pi}{2}$ placed on a vertex of an n-ogon. Since this paper only deals with $\frac{\pi}{2}$-floodlights, and for simplicity, the term *floodlight* is used instead of "$\frac{\pi}{2}$-floodlight". It is assumed that the vertex floodlights are edge-aligned and that each reflex vertex has at most two vertex floodlights. Urrutia [**9**] proved that $\lfloor \frac{3n-4}{8} \rfloor$ vertex floodlights are occasionally necessary and always sufficient to illuminate a n-ogon. But for many n-ogons this number is clearly too large. This fact justifies the algorithmic MVF(P) problem. It is strongly believed that this problem is $\mathcal{NP}$-hard. A way to deal with this computational complexity is to develop approximation algorithms to tackle the problem. In general, these approximation methods can be designed specifically to solve the problem (e.g., *greedy strategies*) or can be based on general metaheuristics (e.g., *Simulated Annealing* (SA) and *Genetic Algorithms* (GAs)). There are several works where non-metaheuristc based approximation algorithms were developed to tackle art gallery problems (e.g., [**2, 5, 6, 8**]). Recently, some work has been made on the application of metaheuristic techniques for these problems (see [**1, 3**]).

Our contribution: We present four approximation algorithms, based on general metaheuristics, to tackle the MVF(P) problem. Since the optimal solution to the MVF(P) problem is unknown, we developed a method that allows us to determine a lower bound for our algorithms, as in [**2**]. In this way, we are able to find the approximation ratio of our strategies. Our experiments were performed on a large set of randomly generated orthogonal simple polygons.

[4]Supported by FCT through CIDMA of University of Aveiro.
[5]Partially supported by Projects MTM2008-05043 and HP2008-0060.
[6]Supported by FCT through CIDMA of University of Aveiro and grant SFRH/BPD/66431/2009.

1 Approximation methods

A set of vertex floodlights for an n-ogon P is a *vertex floodlighting set* for P if they illuminate P. We denote a vertex floodlighting set for P by F and its cardinality by $|F|$. Since the existence of an efficient algorithm to determine a minimum-cardinality vertex floodlighting set is unknown, we developed four approximation algorithms to tackle this problem. The first is based on the SA metaheuristic, called M_1; the second is based on the GAs metaheuristic, named M_2, and the last two are hybrid algorithms, designated by M_3 and M_4.

Simulated Annealing Strategy (M_1). A *configuration* is a chain with length $n + r$ (the reflex vertices are duplicated to determine the two possible positions of the floodlight), where the value of each element is 0 (floodlight-"on") or 1 (floodlight-"off"); see Figure 1.

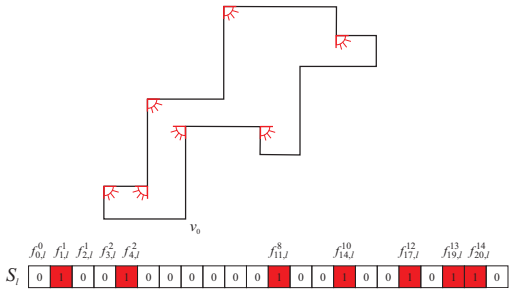


FIGURE 1. A configuration for a 16-ogon.

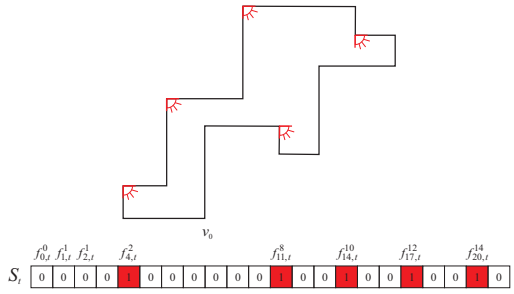


FIGURE 2. Initial configuration.

The *objective function* assigns to each configuration its number of 1's. A *neighbour* of a configuration is generated by switching from 0 to 1, or vice versa, a randomly chosen element. To generate the *initial configuration*, we used the top-left rule **[9]**, that guarantees the illumination of P (see Figure 2). We performed a comparative study taking into account three different *initial temperatures* T_0: (1) $T_0 = n$; (2) $T_0 = n/4$ and (3) $T_0 = 500$. Concerning the *temperature decrement rule*, we made an analysis on three different types of rules: (1) $T_{k+1} = T_0/(1 + k)$; (2) $T_{k+1} = T_0/e^k$ and $T_{k+1} = 0.9 \times T_k$. The *number of iterations in each temperature* T_k is equal to $\lceil T_k \rceil$. Finally, the *termination condition* consists in finishing the search when $T_k \leq 0.005$ or when during 3000 consecutive series of temperatures no new best solution is obtained and the percentage of accepted solutions is less than 2%.

Genetic Algorithms Strategy (M_2). An *individual* is represented by a chain with length $n + r$, where the value of each gene is 0 or 1. We choose the *population size* to be $\lfloor \frac{3n-4}{8} \rfloor$. To create the *initial population*, we generate each of the $\lfloor \frac{3n-4}{8} \rfloor$ individuals in the following way: all of its genes are set to 1; then a gene is randomly selected and its value is set to 0 if the resultant individual is valid; otherwise its value remains 1. The *fitness function* is defined as the number of 1's in each individual. We used the tournament selection method to the genetic operator *selection* and a variant of the single point crossover to the *crossover*. The *mutation* step is relatively simple: for each binary digit it merely flips it from 0 to 1 or vice versa (if the obtained individual is not valid, it is rejected). To *generate a new population*, the worst individual is replaced by the child

obtained at mutation. In order to *evaluate a population*, we consider its fitness as the minimum value of the fitness function when applied to its individuals. Finally, we *stop the search* when the fitness of the population remains unchanged for 500 generations.

Hybrid Strategies (M_3 and M_4). GAs and SA are population-based and single-solution search methods, respectively. Different combinations of these two types of metaheuristics have provided powerful search algorithms. These combinations are known as hybrid metaheuristics [**7**]. To solve the MVF(P) problem, we developed two different hybrid metaheuristics, that fundamentally use a genetic algorithm. However, in the first method, M_3, for the initial population of the genetic algorithm we generate $\lfloor \frac{3n-4}{8} \rfloor$ individuals, running $\lfloor \frac{3n-4}{8} \rfloor$ times the SA metaheuristic. In the second method, M_4, in addition to the classical crossover and mutation operators, we add a new genetic operator based on the SA metaheuristic. Basically the process consists of applying the SA after the crossover operator, in order to refine the solution produced by that operator. After this operation, the mutation operator is applied. Since SA is a genetic operator, it occurs with probability p_{sa}.

Since the optimal solution for the MVFL(P) problem is unknown, we developed a method to compute a lower bound on the optimal number of vertex floodlights for each instance in the performed experiments. For that, we used the notion of *floodlight visibility-independent set*, which is a finite set of points on an n-ogon P, $FIS \subset P$, such that, for all $p, q \in FIS$, p and q are not illuminated by the same floodlight. It can be concluded that the number of points on a maximum-cardinality floodlight visibility-independent set is a lower bound for the optimal number of vertex floodlights on P. However, as far as it is known, the existence of an efficient algorithm to determine this lower bound is unknown. Therefore, we developed a greedy algorithm to find large floodlight visibility-independent sets, which we designated by A_1.

2 Experiments and results

The implementation of our algorithms was done in C/C++ (for MS Visual Studio 2005) on top of CGAL 3.2.1 [**4**]. The above described methods were tested on a PC featuring an Intel(R) Core (TM)2 CPU 6400 at 2.66 GHz and 1 GB of RAM. We performed extensive experiments with the strategies described in the previous section on a large set of randomly generated orthogonal polygons. To generate these polygons, we used the polygon generator developed by O'Rourke (personal communication, 2002). In this section we present our results and conclusions from the experiments. According to Section 1, there are several choices for two of the SA parameters: T_0 and the temperature decrement rule. The different combinations of their values give rise to nine cases. We analyzed these nine cases by comparing the number of vertex floodlights $|F|$, the runtime (in seconds) and the number of iterations performed by each of them. We carried out a statistical study to compare the results obtained by them, but due to lack of space we omit its details. In this study, we concluded that: (i) concerning $|F|$, the case where $T_0 = 500$ and $T_{k+1} = T_0/(1+k)$ is the best one. Therefore, this was the case considered as the SA strategy, i.e., method M_1; (ii) regarding runtime, the case where $T_0 = n/4$ and $T_{k+1} = T_0/e^k$ is the fastest algorithm and, although the returned number of vertex floodlights is worse, it can be used in the hybrid methods. So we used it to generate the initial population in M_3 and as genetic operator in M_4. Then we analyzed and evaluated the

results obtained with our four methods. Table 1 presents the obtained results (averages of 40 n-ogons each one).

n	M_1			M_2			M_3			M_4		
	$\|F\|$	Time	Iterations	$\|F\|$	Time	Iterations	$\|F\|$	Time	Iterations	$\|F\|$	Time	Iterations
30	7.75	13.85	4916.80	8.47	5.70	297.875	7.80	7.15	520.15	6.97	11.52	699.50
50	12.52	39.85	6436.30	14.20	37.37	1071.10	12.27	28.05	580.60	11.62	44.17	683.85
70	17.90	89.22	8295.60	19.97	105.75	1160.30	17.52	78.70	661.90	16.25	127.80	893.60
100	25.12	185.10	10202.00	29.05	290.35	2278.30	25.57	243.20	754.55	23.30	373.55	1061.50
110	27.50	230.62	11049.00	31.40	357.87	1954.80	27.97	324.65	677.77	25.00	485.30	1052.70
130	33.00	340.52	12683.00	37.30	531.40	2360.10	33.30	547.20	783.57	30.12	747.92	1158.50
150	37.65	441.57	13661.00	42.02	763.60	3284.60	38.47	842.50	843.05	34.70	1078.10	1241.40
200	50.25	871.90	17972.00	56.82	1584.50	4568.10	52.22	2163.30	919.05	46.70	2660.20	1520.60

TABLE 1. Results obtained with M_1, M_2, M_3 and M_4.

We performed a statistical analysis to check the differences between the solutions obtained with them (again, we omit its details), and we concluded that: concerning the obtained solutions, the hybrid method M_4 is the best one and the method M_2 the worst one. The methods M_1 and M_3 can be considered equal. Consequently, we continued our study considering M_4 as the best strategy. To infer about the average of the minimum number of vertex floodlights needed to illuminate an orthogonal polygon, we used the least squares method and the following linear adjustment was obtained, with a correlation factor of 0.99: $f(x) = 0.2328x - 0.1091 \approx \frac{x}{4.29}$. Thus, it can be concluded that, on average and approximately, the minimum number of vertex floodlights needed to illuminate an n-ogon was observed to be $\lceil \frac{n}{4.29} \rceil$. In order to get a quantitative measure on the quality of the calculated $|F|$, the floodlights visibility-independent sets were computed on our instances (the eight sets of polygons described above). The ratio between the smallest F (obtained with M_4) and the largest *FIS* (obtained with A_1) never exceeded 2, which implies that algorithm M_4 has an approximation ratio less than or equal to 2.

References

[1] M. Abellanas, E. Alba, S. Canales and G. Hernández, Solving the illumination problem with heuristics, in: *Numerical Methods and Applications*, Vol. 4310 of LNCS, Springer (2006), 205–213.

[2] Y. Amit, J. S. B. Mitchell and E. Packer, Locating Guards for Visibility Coverage of Polygons, in: *Proceedings of the Workshop on Algorithm Engineering and Experiments*, 2007, 1–15.

[3] A. L. Bajuelos, A. M. Martins, S. Canales and G. Hernández, Metaheuristic approaches for the minimum vertex guard problem, in: *The Third IEEE International Conference on Advanced Engineering Computing and Applications in Sciences*, IEEE Computer Society (2009), 77–82.

[4] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

[5] S. Ghosh, Approximation algorithms for art gallery problems in polygons, in: *Discrete Applied Mathematics*, **158** (2010), 718–722.

[6] M. C. Couto, C. C. Souza and P. J. Rezende, An exact and efficient algorithm for the orthogonal art gallery problem, in *Proceedings of the XX Brazilian Symposium on Computer Graphics and Image Processing*, Washington, DC, USA, IEEE Computer Society, 2007, 87–94.

[7] E. G. Talbi, A taxonomy of hybrid metaheuristics, *Journal of Heuristics* **8** (2002), 541–564.

[8] A. P. Tomás, A. L. Bajuelos and F. Marques: Approximation algorithms to minimum vertex cover problems on polygons and terrains, Vol. 2657 of LNCS, Springer (2003), 869–878.

[9] J. Urrutia, Art gallery and illumination problems, in: J.-R. Sack and J. Urrutia, eds., *Handbook of Computational Geometry*, Elsevier, 2000, 973–1027.

A geometrical approach for automatic building texture mapping

M. D. Robles Ortega[1], **L. Ortega**[1], **F. R. Feito**[1]

[1] Departamento de Informática de la Universidad de Jaén, España
mrobles@ujaen.es, lidia@ujaen.es, ffeito@ujaen.es

Abstract. Creating a 3D model of large cities from 2D GIS data and its associated DEM is a complex task. After generating the geometry, a texturization process is needed in order to obtain a realistic scene. There are some requirements to create a believable texturization, like, for example, the building texture should not be crossed by the road surface geometry. In this paper we propose an automatic method to place correctly any building image considering the street slope. We describe all the process to compute the right texture coordinates. Our method allows to reuse the pictures for different buildings.

Introduction

Nowadays there is a growing demand for virtual urban environments in areas such as films, tourism, and games, among others. There are some different solutions for modelling virtual cities like, for example, procedural modelling or L-Systems. In order to obtain realistic scenes, integrating terrain morphology is needed. We have used a method which generates a 2.5D model using 2D GIS data for each urban entity: buildings, streets and crossroads, and also integrates DEM information [1].

The next step in this process is the texturization of all elements in the scene and, more specifically, the buildings. Some authors have used real photographs to obtain more realism in façades [2, 3]. Nevertheless, in these works it is assumed that the city lies on flat land, so the slope of the streets has not been considered. Both these techniques create the 3D model from images, while we propose using texture mapping.

Texture mapping is a shading technique for image synthesis in which a texture picture is mapped onto a surface in a three dimensional scene, much as wallpaper is applied to a wall [4]. Therefore, we work with two related coordinate systems [5]: (1) *Texture space*, the 2D space of surface textures, and (2) *Object space*, the 3D coordinate system in which the 3D geometry (the urban environment) is defined. Our method determines a realistic *parametrization of the surface*, that is, a correct correspondence between the 2D texture space and the 3D object space.

To make the texturing process easier, the buildings have been divided into two parts: top and ground floors. The mapping process in the first category is trivial and does not have any restrictions. However, for ground floors special requirements should be taken into account in order to avoid non-realistic situations. As shown in Figure 1, results are non-realistic if a gate is located above ground level and does not allow the natural entrance to the building, or if a front door or a frieze are crossed by the asphalt street.

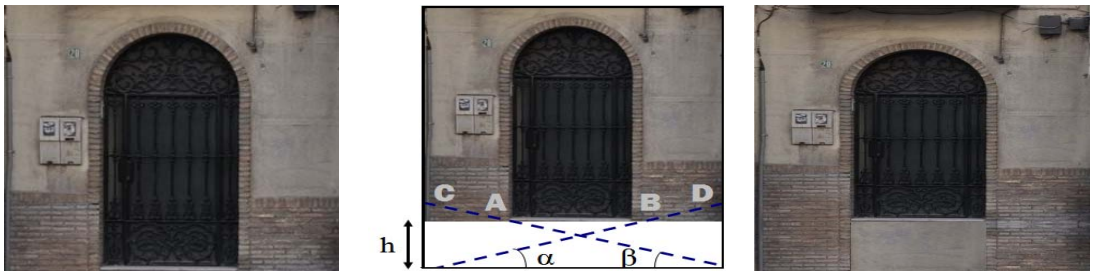
[1]This work has been partially granted by the European Union by means of the ERDF funds, under the research project TIN2007-67474-C03-03 and by the Consejería de Innovación, Ciencia y Empresa of the Junta de Andalucía, under the research project P07-TIC-02773.

FIGURE 1. Non-realistic situations.

1 Preparing the textures

The set of textures with their main features has been stored in a database. To obtain a correct texture placement, only four control points to determine the maximum gradient are needed. Next we explain the process to obtain them. Figures 2(a) and 2(b) show an original texture and its control points, respectively. As can be seen, the points A and B are the corner gates, and C and D are two points situated in the right and left image ends. C and D heights are computed considering the façade elements like the frieze or the windows, and their location in the real world. For instance, if C or D were situated over or just below the frieze, the situation would not be realistic because in any real building façade there is a space between the frieze and the street surface. Evidently, if there were a window in the façade, it should not be crossed by the street surface.

The maximum allowed gradient for each texture is determined by angles $\angle DB = \alpha$ and $\angle CA = \beta$, as depicted in Figure 2(b). Nevertheless, the texture would be incomplete if it were located on a street with this maximum gradient. To avoid this situation, an extra slide of texture with width h should be added in the bottom of the image (Figure 2(c)). Since real photographs can be out of proportion, textures in most cases should be adjusted to maintain the appropriate size. Because the gradient can be affected by the change of size, the maximum and minimum value for α and β are stored in the database.

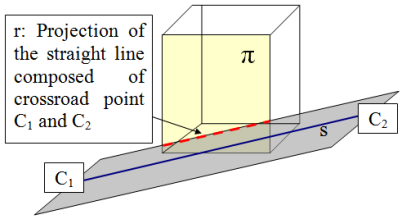


(a) Original texture (b) Control points (c) Final texture

FIGURE 2. Pre-processing of the texture.

2 How to place the texture correctly

Once textures have been pre-processed, the next step is to determine the correct texture coordinates to map the image onto the building. As input data, we have a wall-building



(1) Compute the plane π associated to the edge of the building
(2) Compute the straight line s through the two crossroad points (C_1 and C_2)
(3) Project s line onto plane π, obtaining r
(4) Mapping the texture onto the wall building (Figure 4)

FIGURE 3. Computing the line through crossroad points C_1 and C_2 projection.

polygon (a set of polylines determining its footprint and its height), and a 3D segment line representing the portion of street between two crossroads points with a specific gradient.

For the classic texture mapping, texture coordinates (u, v), $u \in [0, 1]$, $v \in [0, 1]$ are assigned to each vertex. We use affine transformations because placing the texture correctly on the building is possible using only scales and translations. Considering points in texture space like $p_t = (u, v, q)$ and points in the object space like $p_o = (x, y, w)$, the 2D affine mapping matrix can be written algebraically as:

$$p_o = p_t\, M_{ot};$$
$$\begin{pmatrix} x & y & 1 \end{pmatrix} = \begin{pmatrix} u & v & 1 \end{pmatrix} M_R\, M_S\, M_T.$$

Notice that we have chosen $w = q = 1$ without loss of generality. As the perspective of all textures has been previously corrected, an additional rotation transformation is not needed. Therefore, M_R corresponds to the identity matrix. The scale matrix M_S contains the scale factors for both vertical and horizontal dimension to adjust texture width and height with the building wall. Finally, M_T represents the translation. It is assumed that the texture is on the same plane as the wall, so only a vertical transformation is needed. This factor is obtained by computing the intersection between the street line and the building wall, as Figure 3 summarizes. Next, we explain the process. First, the plane associated to the edge of the building π and the line s through the two crossroad points C_1 and C_2 are computed (steps 1, 2). Then, s is projected onto the plane, obtaining the straight line r (step 3). The final step is mapping the texture image onto the wall surface (see Figure 4). In the beginning the texture is scaled to have the same width and height as the building wall (step a). If the texture was directly mapped onto the surface, the position of the gate would be (x_B, y_B). However, as the object space image shows, the gate position is not realistic to allow the access to the buildings. To obtain the right position, the texture should be vertically translated towards point F (step b). The y-coordinate of F can be easily computed using similarity of triangles. Finally, the value of the required vertical translation V should be obtained using the formula $V = y_B - y_F - y_E$ (step c). The position of point E should be included because the straight line r intersects with π at a specific height (y_E).

3 Results

Figure 5 shows some screenshots of the images we have obtained. As can be seen, the scene is realistic because all doors and windows are placed correctly. We also have created a database to store all the data related to the textures.

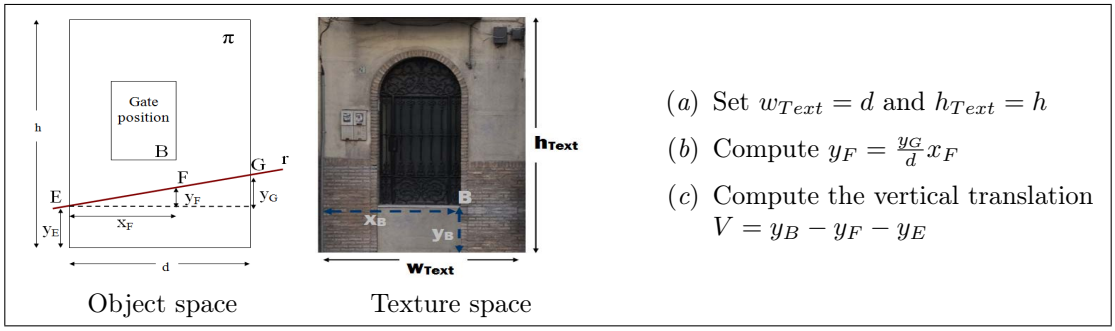


FIGURE 4. Mapping the texture onto the wall building.

FIGURE 5. Some pictures of the textured buildings.

4 Conclusions and future work

In this paper we have presented an automatic method which allows to create realistic mappings of façade textures in buildings situated in non horizontal streets. We have described how to determine a set of control points for each texture which determines the maximum gradient that an image allows. We also explain the process of computing the correct texture coordinates to avoid crossings between doors, windows or friezes and the road surface. Our algorithm can be used to create automatic approaches to texturize whole hilly cities using only a reduced number of textures. As future work we want to include new images in our database.

References

[1] M. D. Robles Ortega, L. Ortega, F. Feito, A. Coelho and A. Augusto de Sousa, Hacia una ciudad virtual, *Congreso Español de Informática Gráfica* (2010), 289–292 (in Spanish).

[2] J. Xiao, T. Fang, P. Zhao, M. Lhuillier and L. Quan, Image-based street-side city modeling. *ACM Transaction on Graphics (TOG), Proceedings of ACM SIGGRAPH Asia*, **28(5)**, 2009. Article 114.

[3] P. Müller, G. Zeng, P. Wonka and L. Van Gool, Image-based procedural modeling of façades, *Proceedings of ACM SIGGRAPH 2007 – ACM Transactions on Graphics*, **26(3)**, 2007.

[4] E. E. Catmull, *A subdivision algorithm for computer display of curved surfaces*. Ph.D. Thesis. Department of Computer Science, University of Utah, December 1974.

[5] P. S. Heckbert, *Fundamentals of texture mapping and image warping*. Master's thesis, UCB/CSD 89/516, CS Division, University of California, Berkeley, June 1989.

A generalization of the source unfolding of convex polyhedra

Erik D. Demaine[1], **Anna Lubiw**[2]

[1] MIT
edemaine@mit.edu

[2] University of Waterloo
alubiw@uwaterloo.ca

Abstract. We present a new method for unfolding a convex polyhedron into one piece without overlap, based on shortest paths to a convex curve on the polyhedron. Our "sun unfoldings" encompass source unfolding from a point, source unfolding from an open geodesic curve, and a variant of a recent method of Itoh, O'Rourke, and Vîlcu.

Introduction

The easiest way to show that any convex polyhedron can be unfolded is via the *source unfolding* from a point s, where the polyhedron surface is cut at the *ridge tree* of points that have more than one shortest path to s; see **[2]**. The unfolding does not overlap because the shortest paths from s to every other point on the surface develop to straight lines radiating from s, forming a star-shaped unfolding.

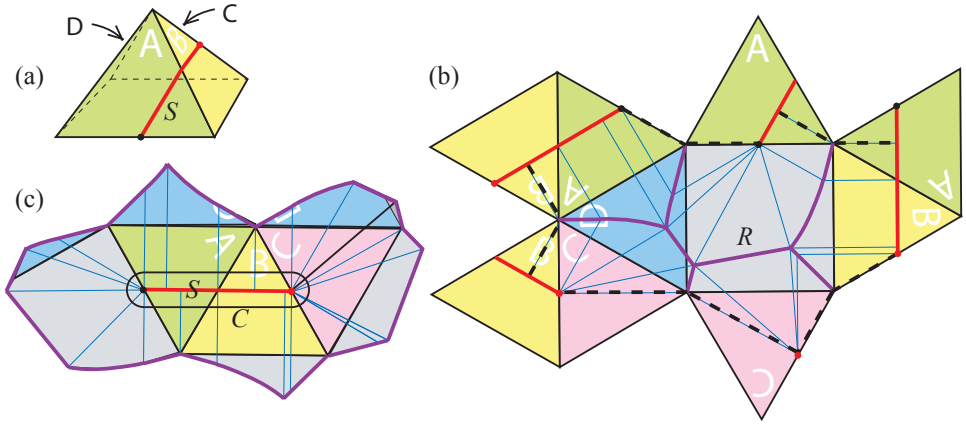


FIGURE 1. Source unfolding from an open geodesic: (a) a pyramid with an open geodesic curve S crossing two faces; (b) the ridge tree R lies in two faces, the base and face D (the dashed lines, together with segments of S delimit a "dual" unfolding where the paths are attached to R); (c) the source unfolding showing paths emanating radially from the open geodesic, and showing the convex curve C relevant to sun unfolding.

Our main result is a generalized unfolding, called a *sun unfolding*, that preserves the property that shortest paths emanate in a radially monotone way, although they no longer radiate from a point. We begin with an easy generalization where the point is replaced

by a curve S that unfolds to a straight line segment (an *open geodesic curve*). Cutting at the ridge tree of points that have more than one shortest path to S produces an unfolding in which the shortest paths from S radiate from the unfolded S; see Figure 1. (All proofs can be found in the long version of the paper.)

For our general sun unfolding, the paths emanate radially, not from a point or a segment, but from a tree S, and the paths are not necessarily shortest paths from S. We define both S and the paths based on a convex curve C on the surface of the polyhedron. Let S be the ridge tree of C on the convex side and let R be the ridge tree of C on the other side. Let $\mathcal{G}$ be the set of all shortest paths to C, where we glue together any paths that reach the same point of C from opposite sides. We prove that the paths emanate in a *radially monotone way* from the unfolded S, and hence that the polyhedron unfolds into a non-overlapping planar surface if we make the following cuts: cut R and, for every vertex v on the convex side of C, cut a shortest path from v to C and continue the cut across C, following a geodesic path, until reaching R. See Figure 2.

Our result generalizes source unfolding from a point or an open geodesic, by taking C to be the locus of points at distance ϵ from the source. See the curve C in Figure 1(c). Our result is related to recent work of Itoh, O'Rourke, and Vîlcu on "star unfolding via a quasigeodesic loop" [**4**]. A quasigeodesic loop is a special case of a convex polygonal curve. Itoh et al. cut the polyhedron at the loop, unfold both halves (keeping R and S intact) and attach the resulting two pieces. Their unfolding of the convex side is the same one that we use. See Figure 3. Itoh, O'Rourke, and Vîlcu also have an interesting alternative unfolding where the convex curve C remains connected (developing as a path) while S and R are cut [**3, 5**]. This is possible only for special convex polygonal curves.

1 Sun unfolding

We define *sun unfolding* of a convex polyhedron P relative to a closed convex curve C on P. For the purpose of this note, we consider only curves composed of a finite number of line segments and circular arcs. In the long version of the paper we discuss more general convex curves. The curve C splits P into two "halves", the convex or *interior* side C_I, and the *exterior* side C_E. For a point c on C (denoted $c \in C$), let $\alpha_I(c)$ be the surface angle of C_I between the left and right tangents at c, and let $\alpha_E(c)$ be the surface angle of C_E between those tangents. Then $\alpha_I(c) + \alpha_E(c) \leq 2\pi$, with equality unless c is a vertex of P. Also, $\alpha_I(c) \leq \pi$. A point c with $\alpha_I(c) < \pi$ is called an *internal corner* of C. A point $c \in C$ with $\alpha_E(c) < \pi$ is called an *external convex corner* of C.

The *ridge tree* (a.k.a. "cut locus") in C_I (or C_E) is the closure of the set of points that have more than one shortest path to C. Let S be the ridge tree of C in C_I, and let R be the ridge tree in C_E. Among all shortest paths from points of C_I to C, let $\mathcal{G}_I$ be the maximal ones. Among all shortest paths from points of C_E to C, let $\mathcal{G}_E$ be the maximal ones. If $c \in C$ has $\alpha_I(c) = \alpha_E(c) = \pi$, then we concatenate together the unique paths of $\mathcal{G}_I$ and $\mathcal{G}_E$ that are incident to c. Let $\mathcal{G}$ be the resulting set of paths, together with any leftover paths of $\mathcal{G}_I$ and any leftover paths of $\mathcal{G}_E$.

Lemma 1.1. *Both R and S are trees. Every vertex of P lies in R or S (or both). Every internal corner of C is a leaf of S. Every external convex corner of C is a leaf of R. Every path of $\mathcal{G}$ goes from S to R and includes a point of C. The surface of P is covered by S, R, and $\mathcal{G}$. Furthermore, any point not on S or R is in a unique path of $\mathcal{G}$.*

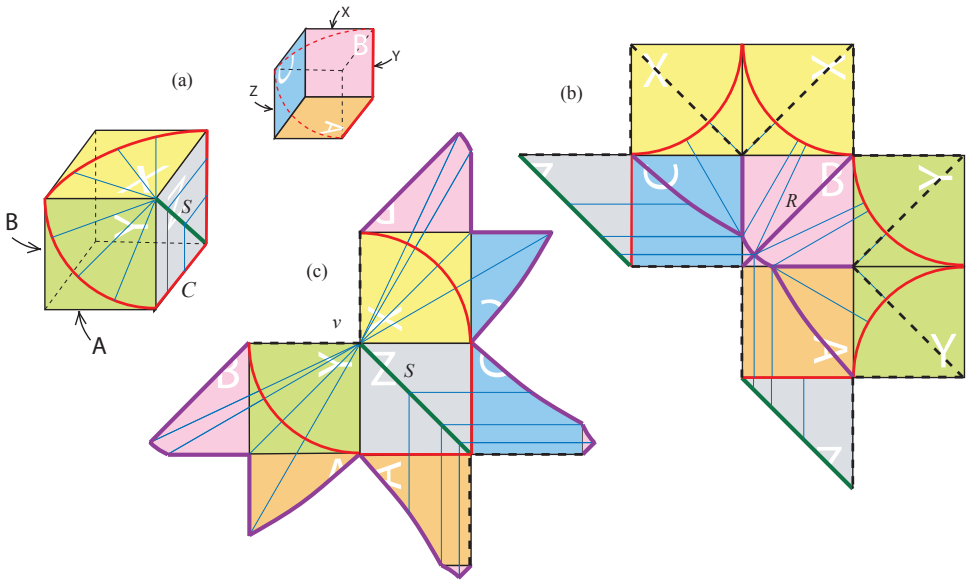


FIGURE 2. Sun unfolding with respect to a convex curve: (a) a cube with a convex curve C and the ridge tree S on the convex side; (b) the ridge tree R on the non-convex side of C (the dashed lines delimit a "dual" unfolding where the paths are attached to R); (c) the sun unfolding showing paths emanating radially from S. Note that the two vertices on S require cuts $\gamma(\cdot)$ to R, shown with dashed lines.

Let v be a vertex of P. If v is not in R, then it is in S, and we let $\gamma(v)$ be a path of $\mathcal{G}$ incident to v. The choice of $\gamma(v)$ is not unique in general, but we fix one $\gamma(v)$. Observe that each $\gamma(v)$ is a path from v to R, consisting of a shortest path from v to C possibly continued geodesically to R. We define *sun cuts* with respect to C to consist of R and the paths $\gamma(v)$, for v a vertex of P in $C_I \cup C$. Note that a vertex on C may be a leaf of R, in which case $\gamma(v)$ has length 0.

Theorem 1.2. *Let C be a closed convex curve on the surface of a convex polyhedron P, such that C is composed of a finite number of line segments and circular arcs. Then sun cuts with respect to C unfold the surface of P into the plane without overlap.*

To prove the theorem, we first show that the sun cuts form a tree that reaches all vertices of P —hence the surface unfolds to the plane. To show that the unfolded surface does not overlap, we prove by shrinking C and applying induction that S unfolds without overlap and that the paths of $\mathcal{G}$ emanate from the unfolded S in a *radially monotone way*, defined as follows. Make a tour clockwise around the unfolded S, travelling in the plane an infinitesimal distance away from the unfolded S. See Figure 3(b).

2 Conclusion

Our sun unfolding generalizes one of the basic unfolding methods for convex polyhedra, namely the source unfolding from a point. The other basic unfolding method is the *star unfolding* from a point s, where the polyhedron surface is cut along a shortest path from every vertex to s [**1**]. This is dual to the source unfolding in that the shortest paths are

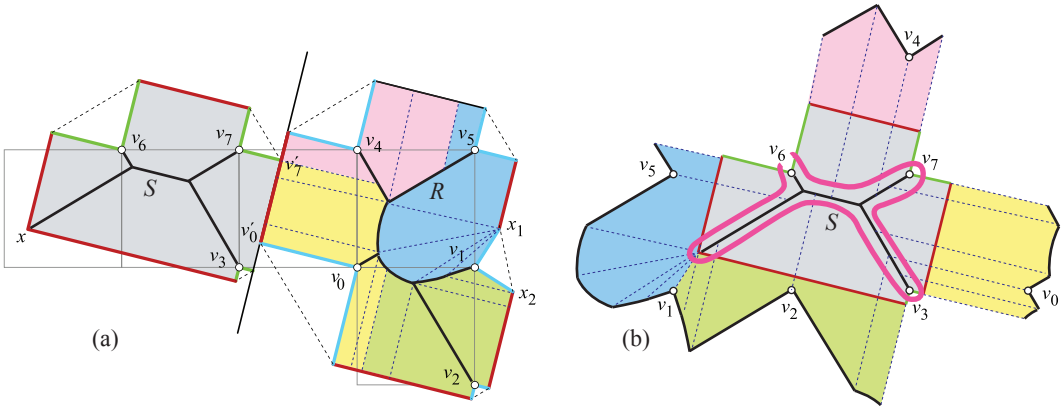


FIGURE 3. The sun unfolding with respect to a geodesic loop Q on a cube, based on an example from Fig. 1 of Itoh et al. [**4**]: (a) the ridge trees S and R on the two sides of the curve Q (superimposed on the unfolding from [**4**]); (b) the sun unfolding with respect to Q, showing a tour around S and the paths emanating in a radially monotone way from S. The faint horizontal and vertical lines in (a) are the edges of the cube.

attached in one case to s (for source unfolding) and in the other case to the ridge tree (for star unfolding). The dual of our sun unfolding would be to attach the paths of $\mathcal{G}$ to the ridge tree R and cut the ridge tree S and paths of $\mathcal{G}$ from vertices to S. See for example Figure 1(b) and Figure 2(b). We conjecture that this unfolds without overlap. A first step would be to prove that the star unfolding from an open geodesic unfolds without overlap.

Acknowledgments. We thank Joseph O'Rourke for helpful discussions.

References

[1] B. Aronov and J. O'Rourke. Nonoverlap of the star unfolding. *Discrete and Computational Geometry*, 8:219–250, 1992.

[2] E. D. Demaine and J. O'Rourke. *Geometric Folding Algorithms: Linkages, Origami, Polyhedra*. Cambridge University Press, 2007.

[3] J. Itoh, J. O'Rourke, and C. Vîlcu. Source unfoldings of convex polyhedra with respect to certain closed polygonal curves. In *Proc. 25th European Workshop Comput. Geom.*, pages 61–64, 2009.

[4] J. Itoh, J. O'Rourke, and C. Vîlcu. Star unfolding convex polyhedra via quasigeodesic loops. *Discrete and Computational Geometry*, 44:35–54, 2010.

[5] J. O'Rourke and C. Vîlcu. Conical existence of closed curves on convex polyhedra. *CoRR*, 2011, abs/1102.0823.

Continuous flattening of convex polyhedra

Jin-ichi Itoh[1], **Chie Nara**[2], **Costin Vîlcu**[3]

[1] Faculty of Education, Kumamoto University, Kumamoto, 860-8555, Japan
j-itoh@kumamoto-u.ac.jp

[2] Liberal Arts Education Center, Aso Campus, Tokai University, Aso, Kumamoto, 869-1404, Japan
cnara@ktmail.tokai-u.jp

[3] Institute of Mathematics of the Romanian Academy, P.O. Box 1-764, 70700 Bucharest, Romania
Costin.Vilcu@imar.ro

Abstract. A *flat folding* of a polyhedron is a folding by creases into a multilayered planar shape. It is an open problem by E. Demaine et al. that every flat folded state of a polyhedron can be reached by a continuous folding process. The first and the second named authors showed that each Platonic polyhedron can be continuously flat folded onto an original face. Here we prove that every convex polyhedron possesses infinitely many continuous flat folding processes. Moreover, we give a sufficient condition under which every flat folded state of a convex polyhedron can be reached by a continuous folding process.

Introduction

We use the terminology *polyhedron* for a polyhedral surface which is permitted to touch itself but not self-intersect (where a doubly covered polygon is considered a polyhedron). A *flat folding* of a polyhedron is a folding by creases into a multilayered planar shape.

The results presented here are related to the following problem by E. Demaine et al. (see Open Problem 18.1 in [**4**]): *Can every flat folded state of a polyhedron be reached by a continuous folding process?*

Cauchy's theorem says that any convex polyhedron is rigid (i.e., if two convex polyhedra P, P' are combinatorially equivalent and their corresponding faces are congruent, then P and P' are congruent). R. Connelly provided an example of a flexible (non convex) polyhedon, and I. Sabitov [**7, 8**] proved that the volume of any polyhedron is invariant under flexing. (That is, if there is a continuous family of polyhedra $\{P_t : 0 \leq t \leq 1\}$ such that, for every t, the corresponding faces of P_0 and P_t are congruent, then the volumes P_0 and P_t are equal for all $0 \leq t \leq 1$.) Sabitov's theorem implies that, if a polyhedron P is flattened by a continuous folding process (see Definition 1.1) with polyhedra $\{P_t : 0 \leq t \leq 1\}$, then the crease pattern in P for $\{P_t : 0 \leq t \leq 1\}$ is an infinite set of line segments.

The existence of flat folded states was proved by the method of disk packing for polyhedra homeomorphic to the 2-sphere (see §18.3 in [**4**]), and by the method of straight skeleton for some very special classes of convex polyhedra (see §18.4 in [**4**]).

Our Section 1 is devoted to preliminaries. We also briefly present there (Theorem 1.2) the method to continuously flat folding the Platonic polyhedra onto an original face, proposed by the first and the second named authors in [**5**]. Section 2 contains our main results. We propose a method to continuously flat folding general convex polyhedra (Theorem 2.1), and we give a sufficient condition under which every flat folded state of convex polyhedron can be reached by a continuous folding process (Theorem 2.5). We employ Alexandrov's gluing theorem and the structure of cut loci for the proofs.

1 Preliminaries

Definition 1.1. Let P be a polyhedron in the Euclidean space $\mathbb{R}^3$. We say that P *is flattened by a continuous folding process* if there is a continuous family of polyhedra $\{P_t : 0 \le t \le 1\}$ satisfying the following conditions:

(1) for each $0 \le t \le 1$, there is a polyhedron P'_t obtained from P by subdividing some faces of P (i.e., some faces of P'_t may be included in the same face of P, but P'_t is congruent to P) such that P_t is combinatorially equivalent to P'_t and the corresponding faces of P'_t and P_t are congruent,
(2) $P_0 = P$, and
(3) P_1 is a flat folded polyhedron.

We call P_1 a *flat folded polyhedron* (or *state*) of P.

In the case of Platonic polyhedra, the first and second named authors proved **[5]**:

Theorem 1.2. *For each Platonic polyhedron (the regular tetrahedron, the cube, the regular octahedron, the regular dodecahedron, or the regular icosahedron) there is a continuous flat folding process onto an original face.*

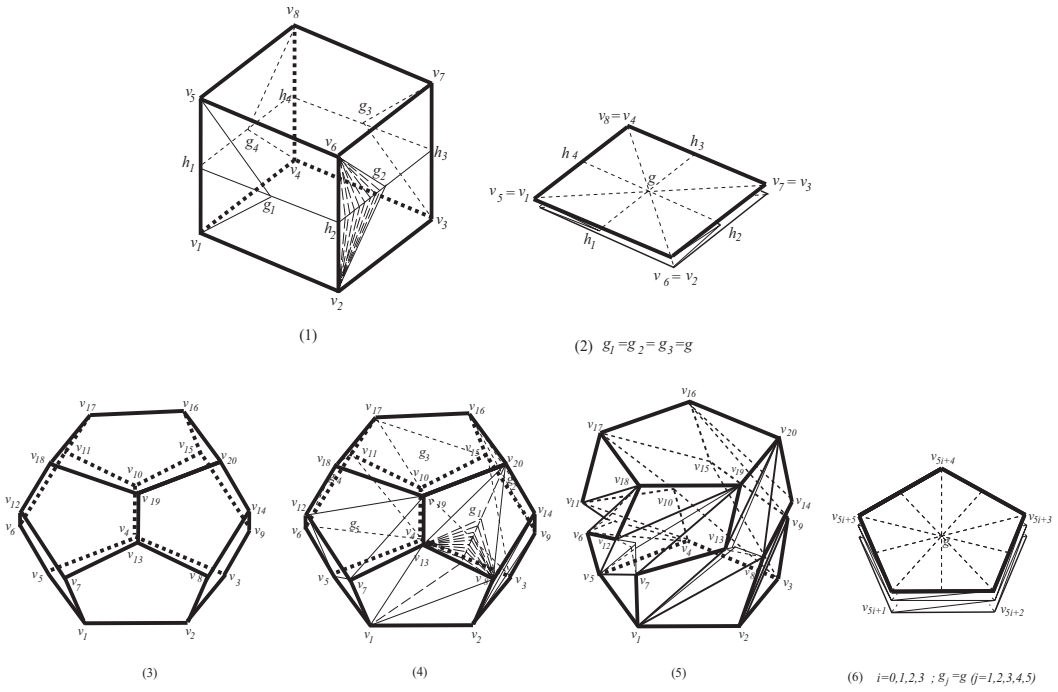


FIGURE 1. A continuous flat folding for the cube and for the regular dodecahedron.

Figure 1 illustrates continuous flat folding processes for the cube and for the regular dodecahedron, each of which is folded onto an original face. Theorem 1.2 was proved by using a key lemma: any rhombus is folded into a shape as showed in Figure 2(2), with distances $|f(b)f(d)| = l$ and $|f(a)f(c)| = m$ for any given $0 \le l \le |bd|$ and $0 \le m \le |bd|$, where we denote by $|xy|$ the Euclidean metric distance between $x, y \in \mathbb{R}^3$.

Our main tools here are the cut locus, described below, and Alexandrov's gluing theorem (see **[2]**, p. 100), saying that gluing polygons to form a topological sphere in

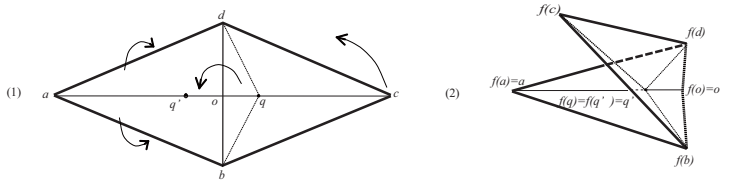


FIGURE 2. An example of a folded rhombus by an intrinsic isometry f.

such a way that at most 2π angle is glued at any point, results in a unique convex polyhedron.

Definition 1.3. Let P be a convex polyhedron. The *cut locus* $C(x) = C(x, P)$ of the point x in P is defined as the set of endpoints (different to x) of all nonextendable shortest paths (on the surface P) starting at x.

It is known that $C(x)$ is a tree whose leaves are precisely the vertices of P, excepting (if the case) x and those vertices of P which are interior to $C(x)$; the junction points in $C(x)$ are joined to x by as many shortest paths as their degree in the tree; the edges of $C(x)$ are shortest paths on P (see Lemma 2.4 in [**1**]).

Assume P has n vertices; then $C(x)$ is a tree with $O(n)$ vertices, and it can be determined in time $O(n^2)$ by the use of the algorithm of J. Chen and Y. Han [**3**] (see [**6**] for a public implementation).

Figure 3(1) shows the cut locus $C(a)$ of the vertex a of the cube.

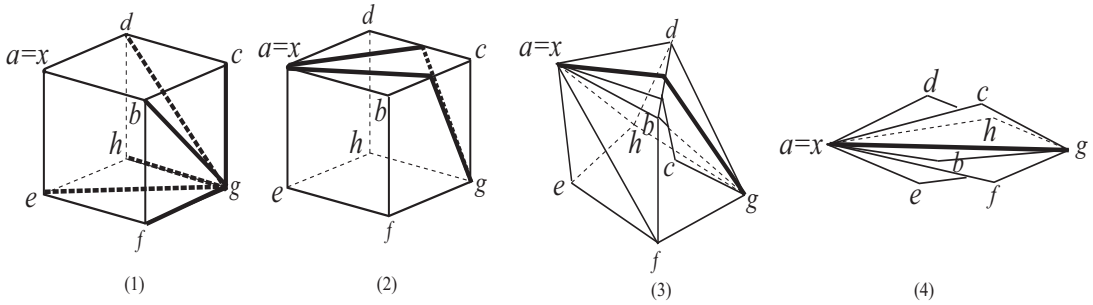


FIGURE 3. (1) The cut locus of $x = a$. (2) Two shortest paths joining x to g (of the existing six), separating c. (3) The resulting polyhedron after cutting along these two shortest paths and gluing. (4) The resulting flat folded polyhedron.

2 Main results

Theorem 2.1. *For every convex polyhedron there exist infinitely many continuous flat folding processes.*

Sketch of proof. Let P be a convex polyhedron and x an arbitrary point in P.

Step 1. Determine the cut locus $C(x)$, which is a tree (see §1).

Step 2. Flatten the regions of P corresponding to *external edges of* $C(x)$ (i.e., edges incident to leaves).

Lemma 2.2. *Each external edge E of $C(x)$ corresponds to a unique loop on P, composed by two shortest paths from x and bounding a region T of P enclosing precisely one vertex of P, the vertex incident to E. Moreover, T can be flattened to a doubly covered triangle.*

Figure 3(2) shows the region of the cube corresponding to the external edge cg in $C(a)$, bounded by two shortest paths from a to g, and Figure 3(3) shows the result after flattening this region.

The remaining part of P, after flattening all Ts as above, is realized as a convex polyhedron Q by Alexandrov's gluing theorem. The result after one iteration of Step 2 consists of Q and doubly covered triangles, as many as the external edges of $C(x)$.

Lemma 2.3. *The cut locus $C(x, Q)$ of x on Q is (isometric to) the truncation of the cut locus $C(x, P)$ of x on P with respect to the cut and glue process.*

Iterating Step 2 we flatten regions corresponding to $O(n)$ external edges (see §1), so the flattening process ends. Figure 3(4) shows the flat folded polyhedron finally obtained from P after flattening all such regions.

Step 3. The continuity of our flat folding process is covered by the next result.

Definition 2.4. A *cut doubly covered convex polygon* consists of two copies of a convex polygon glued along some of their corresponding edges.

A flat folded state P_f of a convex polyhedron is called *simple* if it can be decomposed into cut doubly covered convex polygons whose vertices are vertices of P_f.

Notice that all flat folded states obtained by the use of Theorem 2.1 are simple, while Theorem 1.2 provides examples of non simple flat folded states.

Theorem 2.5. *Every simple flat folded state of a convex polyhedron can be reached by a continuous folding process.*

Sketch of proof. Let P_f be a simple folded state of a convex polyhedron P, decomposed into cut convex doubly covered convex polygons. Write each cut doubly covered polygon, and consequently P_f, as a continuous family of loops. By Alexandrov's gluing theorem and Blaschke's convergence theorem, we can cut and glue along each of these loops, to obtain a continuous family of convex polyhedra $P_t\,(0 \leq t \leq 1)$ in $\mathbb{R}^3$. Each P_t has one or several "wings": doubly covered convex polygons attached to it (to its exterior).

References

[1] P. K. Agarwal, B. Aronov, J. O'Rourke and C. A. Schevon, Star unfolding of a polytope with applications, *SIAM J. Comput.* **26** (1997), 1689–1713.

[2] 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.

[3] J. Chen and Y. Han, Shortest path on a polyhedron, *Proc. 6th Ann. ACM Sympos. Comput. Geom.* (1990), 360–369.

[4] E. D. Demaine and J. O'Rourke, *Geometric Folding Algorithms, Linkages, Origami, Polyhedra*, Cambridge University Press, 2007.

[5] J. Itoh and C. Nara, Continuous flat foldings of Platonic polyhedra, submitted.

[6] B. Kaneva and J. O'Rourke, An implementation of Chen and Han's shortest path algorithm, *Proc. 12th Canadian Conf. Compt. Geom.* (2000), 139–146.

[7] I. Sabitov, The volume of polyhedron as a function of its metric, *Fundam. Prikl. Mat.* **2**(4) (1996), 1235–1246.

[8] I. Sabitov, The volume as a metric invariant of polyhedra, *Discrete Comput. Geom.* **20** (1998), 405–425.

Empty disks supported by a point set

Kiyoshi Hosono[1], **Masatsugu Urabe**[1]

[1] Department of Mathematics, Tokai University, 3-20-1 Orido, Shimizu, Shizuoka, 424-8610 Japan
hosono@scc.u-tokai.ac.jp, qzg00130@scc.u-tokai.ac.jp

Abstract. For a given point set P in the plane, we consider the minimum number of empty disks $\mathcal{D}$ such that each point of P lies on the boundary of some disk of $\mathcal{D}$.

Introduction

Assume given any point set P in the plane. For a disk D in the plane, let P' be a nonempty subset of P on the boundary of D. Then a disk D is said to *be supported* by P'. We call a disk D supported by P' the *bubble* by P' if D is *empty*, that is, no point of P lies inside D. The *bubble set* $\mathcal{D}$ of P is a set of bubbles such that every point of P supports at least one disk of $\mathcal{D}$.

Let $b(P)$ denote the minimum number of bubbles over all bubble sets of P, and define the *bubble number* $B(n)$ as the maximum value of $b(P)$ over all sets P with n points. We estimate the bubble number in this talk.

1 Result

We present the following result.

Theorem 1.1. *For any n point set in the plane with $n \geq 14$,*

$$\left\lceil \frac{n}{2} \right\rceil \leq B(n) \leq \left\lfloor \frac{2n-2}{3} \right\rfloor.$$

We give a sketch of proof. Any collinear n points realize the lower bound of $B(n)$. To prove the upper bound, we use the following lemma [**1**].

Lemma 1.2. *Let G be a 3-connected planar simple graph with $\delta(G) \geq 3$ and $|G| \geq 14$. Then the size of a maximum matching $M(G)$ in G is*

$$|M(G)| \geq \left\lceil \frac{|G|+4}{3} \right\rceil.$$

Let G be a graph of the Delaunay triangulation of P. We remark that the closure of the circumcircle of any triangle in G is a bubble. By adding an additional vertex to G outside the hull and making it adjacent to every vertex on the hull, we construct a 3-connected maximal planar graph G'. Note that $\delta(G') = 3$. By Lemma 1.2, we have $\left\lceil \frac{(n+1)+4}{3} \right\rceil$ matchings in G'. Now, the disk corresponding to a triangle with a matching is a bubble, and an unsaturated vertex corresponds to a very small disk. Hence we obtain the desired result.

CRM Documents, vol. **8**, Centre de Recerca Matemàtica, Bellaterra (Barcelona), 2011

2 Discussion

(i) The lower bound of $B(n)$ is also given by the points on the vertices of a $(2k-1)$-regular polygon and the point on the center for $n = 2k$.

(ii) The problem of $B(n)$ can be generalized, in an obvious way, to higher dimensions. Let $B^d(n)$ denote the maximum of minimum number of empty spheres in $\mathbb{R}^d, d \geq 2$. Another interesting problem is to study the similar question for *disjoint* bubbles or spheres. Given a point set P in $\mathbb{R}^d$, let $s^d(P)$ be the minimum number of sets in a partition into disjoint spheres of P for $d \geq 2$. Define $S^d(n) = \max\{s^d(n)\}$ over all sets P of n points. In general, the inequality $S^d(n) \geq B^d(n)$ holds for every positive integer n. It would be nice to generalize values of n, d.

We conjecture that $B^2(n) = S^2(n) = n/2$ for any set of n points.

References

[1] T. Nishizeki and I. Baybars, Lower bounds on the cardinality of the maximum matchings of planar graphs, *Discrete Math.* **28** (1979), 255–267.

A note on the number of empty triangles

Alfredo García[1]

[1] Departamento de Métodos Estadísticos and IUMA, Universidad de Zaragoza, Zaragoza, 50009, Spain
olaverri@unizar.es

Abstract. Let P be a set of n points on the plane, in general position, H of them placed on the boundary of the convex hull of P. In this note we prove that there is a well defined family of empty triangles —the family of empty triangles not generated by an empty pentagon— containing exactly $n^2 - 5n + H + 4$ empty triangles. Notice that this result immediately implies a slight improvement on the lower bound on the number of empty triangles that every set of n points in the plane must determine.

Introduction

In a classic paper, Bárány and Füredi [**3**] proved that given a set P of n points in the plane, in general position, the number of empty triangles determined by P is at least $n^2 - O(n \lg n)$. To our knowledge (see [**1**]) this lower bound has not been improved. In the same paper, they proved that the expected number of empty triangles for a set of random points is $2n^2 + O(n \lg n)$, this last result implying that there are sets of points having $\leq 2n^2 + O(n \lg n)$ empty triangles. In fact, Bárány and Valtr [**2**] constructed sets of n points in general position containing $\leq 1.6196n^2 + o(n^2)$ empty triangles.

In this note, we slightly improve the lower bound given in [**3**], by proving that any set of points P contains at least

$$n^2 - 5n + H + 4 + \lfloor (n-4)/6 \rfloor$$

empty triangles, where H is the number of points on the boundary of the convex hull of P.

Probably more interesting that this new lower bound is the result that there is a well defined family of empty triangles —let us call them "triangles not generated from an empty pentagon"— containing an invariant number of empty triangles —exactly $n^2 - 5n + 4 + H$ triangles.

This family of empty triangles that we are going to count appears in a paper of Pinchasi et al. [**5**], and we use for this family the name "empty triangles not generated by an empty pentagon", used by them. In that paper, given an empty pentagon Q spanned by P, they call triangle generated by Q the empty triangle whose vertices are the top vertex p of Q and the two vertices of Q not adjacent to p. They study the number of empty triangles not generated from an empty pentagon, proving that this number is at most $(n-2)(n-3) + H'$, where $H' < n$ is a new geometric parameter (different from H and depending on the set P of points and also on the chosen direction to explore the points). However, they did not realize that, in fact, the number of empty triangles of that family is an invariant, and they did not provide any lower bound for that number. As in that paper, we are going to denote by $X_k(P)$ the number of empty convex k-gons determined by P.

[2]Partially supported by research grants MTM2009-07242 and E58-DGA.

1 Counting empty triangles

Let us first define exactly what type of empty triangles we are counting. Suppose that the n points of $P = \{p_1, p_2, \ldots, p_n\}$ are sorted in increasing order of the ordinate y, that is, p_1 is at the bottom, and p_n is at the top. Given an empty triangle $\Delta = p_i p_j p_k$, $(i < j < k)$, we call region A of that triangle the (bounded) region of the plane placed below the horizontal line passing through p_k, line l_k, and placed between Δ and the ray $p_i p_j$. See Figure 1. In the same way, we will call region C the (unbounded) region of the plane placed below l_k, and limited by Δ and the ray $p_j p_i$. We will say that region A (C) of a triangle is empty if it does not contain points of P.

Let us denote by F_e the family of empty triangles such that both zone A and zone C contain points of P. Notice that given an empty pentagon Q, the triangle generated by Q contains points in both zones A and C, so, it belongs to the family F_e. Reciprocally, a triangle $\Delta = p_i p_j p_k$ of F_e is generated by the empty convex pentagon Q formed by these three vertices of Δ, plus the closest point to Δ in region A and the closest point to Δ in region C, that is to say, F_e coincides with the family of empty triangles generated by an empty pentagon. The complementary family, the family of empty triangles such that region A or region C (or both) is empty, let us denote it by F_o, is the one with an invariant number of triangles.

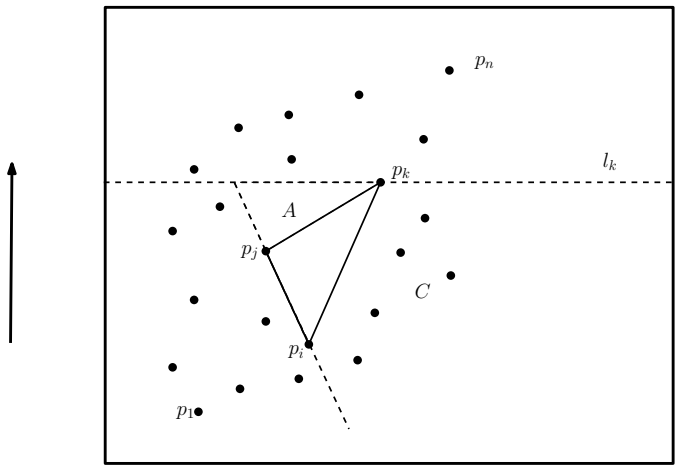


FIGURE 1. Regions A and C of an empty triangle.

Now, the main result.

Theorem 1.1. *Let P be a set of n points in general position, H of them placed on the boundary of the convex hull of P. Then, the number of empty triangles of P such that region A or region C (or both) is empty equals*

$$n^2 - 5n + H + 4.$$

Proof. First of all let us denote by T the triangulation obtained by the greedy method, (exploring the points in its natural order $p_1, p_2, \ldots, p_n$). Initially T contains only the triangle p_1, p_2, p_3; then we add to T the triangles (placed outside of the convex hull of the previous points) with top point p_4, next the triangles with top point p_5 and so on. In this process, when we are adding triangles with top vertex p_k, a triangle $\Delta = p_i p_j p_k$

is added to T if and only if p_k lies in one half-plane of the line p_ip_j and all the other points p_h $(h < k)$ lie in the other half-plane. Therefore, that triangle Δ is added to T if and only if both regions A and C (of Δ) are empty, or, in other words, the $2n - H + 2$ triangles of the greedy triangulation T are precisely the empty triangles of P such that region A and region C are both empty.

Now we are going to form three families of empty triangles. First, for each point p_i $(1 \leq i \leq n-2)$ consider the rays emanating from p_i and passing through the $n-i$ points placed above p_i. Let us order clockwise these rays, and let $p_{i_1}, p_{i_2}, \ldots, p_{i_{n-i}}$ be the points of P placed in that order on the rays. Clearly, since the space between consecutive rays does not contain points of P, region A of each triangle $p_ip_{i_j}p_{i_{j+1}}$ $(j = 1, n-i-1)$ is empty. Let us call F_1 this first family of empty triangles. Notice that $|F_1| = (n-1)(n-2)/2$.

Suppose now that, from the above configuration of rays emanating from p_i, we remove p_n, the top point of P. If the ray p_ip_n is not the first one nor the last one of the above configuration of rays, that is $p_n = p_{i_j}$, $j \neq 1, j \neq n-i$, then we can form a new empty triangle $p_ip_{i_{j-1}}p_{i_{j+1}}$. We can repeat this process removing the following top point p_{n-1}, then point p_{n-2}, and so on, until removing point p_{i+2}. After removing each point, if the corresponding ray is not the first nor the last ray, we obtain a new triangle. Let us call F_2 the family of triangles obtained by this method (this family can be empty). By construction region A of each triangle of F_2 is empty, and a triangle of F_2 cannot belong to F_1. As the process of removing top points is made $n-i-1$ times for each point p_i, we have

$$|F_2| + |\text{Extreme rays}| = (n-1)(n-2)/2,$$

where |Extreme rays| is the number of times that an extreme ray (first or last ray) is found.

Now consider the case when we have to remove vertex p_k, $k > i+1$, and ray $r_k = p_ip_k$ is the last ray or the first one. Without loss of generality, suppose that ray r_k is the last one. Notice that, at this moment of the process, we are only considering points in the range $p_i, p_{i+1}, \ldots, p_k$, and we are supposing that ray r_k is the last one in this subset of rays. Let p_ip_j be the ray placed just before ray r_k, and consider the triangle $\Delta = p_ip_jp_k$. If region C of this triangle is nonempty, then rotating clockwise the ray p_ip_k we first reach a point p_h $(h < i)$ on that region (now the ray p_ip_h is going downward). In this case, consider the empty triangle $p_hp_ip_k$. By construction, zone C of this new triangle is empty; however, zone A is nonempty because point p_j is in that zone. Therefore, this new triangle $p_hp_ip_k$ does not belong to $F_1 \cup F_2$. Doing this construction, when possible, we obtain a new family of triangles F_3 (perhaps empty). Again, by construction, notice that

$$|\text{Extreme rays}| = |F_3| + |\text{Cases with } C = \emptyset|.$$

Finally, observe that region A of triangle $\Delta = p_ip_jp_k$ is empty, so, if region C of this triangle is also empty, this situation happens, if and only if, the triangle $\Delta = p_ip_jp_k$ belongs to T, the greedy triangulation, that is $2n - H - 2$ times. Summarizing, we have

$$|F_1| + |F_2| + |F_3| + 2n - H - 2 = (n-1)(n-2),$$

as claimed. □

2 Some implications

- Notice that, given a set of points P, the families $F_o = F_1 \cup F_2 \cup F_3$ and F_e depend on the direction $\vec{d}$ that we take to order the points (upward in our case), yet the numbers $|F_o|$ and $|F_e|$ do not depend on the chosen direction $\vec{d}$.
- As we said, given a triangle $\Delta = p_i p_j p_k$ of F_e we can associate to it the empty convex pentagon Q formed by these three vertices of Δ, plus the closest point to Δ in region A and the closest point to Δ in region C. We can check that different triangles of F_e have associated different pentagons, so the number of empty pentagons of P is at least $|F_e|$, or, in other words,
$$X_5(P) \geq X_3(P) - (n^2 - 5n + H + 4).$$
- In the same way, given an empty pentagon Q, there is a unique empty triangle generated by Q. However, different pentagons can generate a same triangle of F_e. In any case, since, for $n \geq 10$, P must contain empty pentagons (see [**4**]), we can immediately deduce the following bound for the number of empty triangles that every set of n points in the plane must determine:
$$X_3(P) \geq n^2 - 5n + H + 4 + \lfloor (n-4)/6 \rfloor.$$

Proof. The first 10 points, $p_1, \ldots, p_{10}$, contain an empty pentagon Q_1; the last four points of this group plus the following six points $p_{11}, \ldots, p_{16}$ contain a different empty pentagon Q_2, and adding six new points we obtain another new pentagon Q_3, and so on. Notice that the top points of each one of these pentagons $Q_1, Q_2 \ldots$ are all of them different, therefore the triangles generated by $Q_1, Q_2 \ldots$ are also different. Thus F_e contains at least $\lfloor (n-4)/6 \rfloor$ triangles. □

References

[1] O. Aichholzer. [Empty] [colored] k-gons – Recent results on some Erdős–Szekeres type problems. In *Proc. XIII Encuentros de Geometría Computacional*, (2009) 43–52. Prensas universitarias de Zaragoza, Spain.

[2] I. Bárány, P. Valtr. Planar point sets with a small number of empty convex polygons. *Stud. Sci. Math. Hung.* **41(2)** (2004), 243–269.

[3] I. Bárány, Z. Füredi. Empty simplices in Euclidean space, *Canadian Math. Bull.* **30** (1987), 436–445.

[4] H. Harborth. Konvexe Fünfecke in ebenen Punktmengen, *Elem. Math.* **33** (1978), 116–118.

[5] R. Pinchasi, R. Radoicic, M. Sharir. On empty convex polygons in a planar point set. *J. Comb. Theory, Ser. A*, **113(3)** (2006) 385–419.

Delaunay triangulation of imprecise points, preprocess and actually get a fast query time

Olivier Devillers[1]

[1] INRIA Sophia Antipolis Méditerranée, France
http://www.inria.fr/sophia/members/Olivier.Devillers/

Abstract. We propose a new algorithm to preprocess a set of n disjoint unit disks in $O(n \log n)$ expected time, allowing to compute the Delaunay triangulation of a set of n points, one from each disk, in $O(n)$ expected time. This work reaches the same asymptotical theoretical complexities as previous results on this problem, but our algorithm is much simpler and efficient in practice.

1 Introduction

Löffler and Snoeyink **[8]** proposed an algorithm that preprocesses a set of disjoint unit disks in the plane in $O(n \log n)$ time and computes the Delaunay triangulation of an instance in $O(n)$ time. This algorithm has a reasonably simple description but uses as a building block the linear time construction of the constrained Delaunay triangulation of a simple polygon **[4]**, which makes the result mainly theoretical. Buchin et al. **[2]** proposed a simpler solution, which uses the split of a Delaunay triangulation in linear time **[3]**. This solution remains a bit heavy in practice; indeed, in the preprocessing phase, they compute a Delaunay triangulation of $8n$ points (at the center and on the boundary of the disks), then the points of the instance are added in linear time to get a triangulation of $9n$ points, and finally this triangulation is split in the triangulation of the instance and the $8n$-point triangulation again. In the same paper, a different algorithm based on quadtrees is proposed allowing overlapping disks of different radii.

Contribution. In this paper we preprocess in $O(n \log n)$ randomized expected time a set of n disjoint unit disks, allowing the computation of the Delaunay triangulation of an instance taken in these disks in randomized expected $O(n)$ time. Compared to previous algorithms **[2, 8]**, theoretical asymptotic complexity is not improved, but the proposed algorithm is much simpler —so simple that its description fits in a dozen of lines.

Moreover, the algorithm is quite efficient in practice and uses only the classical predicates for Delaunay triangulation. Our benchmarks conclude that we can process an instance much faster than with the Delaunay hierarchy **[6, 10]** and faster than the incremental algorithm inserting the points in spatial sorting order **[1, 5]**.

The algorithm works for any set of circles (overlapping, different radii) and generalize to balls in higher dimensions, but to yield a good complexity the analysis requires that the imprecise points are unit disks in the plane, possibly overlapping a constant number of times (at most k disks have a common intersection). This analysis can be extended to unit balls in higher dimensions under suitable hypotheses. For disks of different radii overlapping at most twice, we provide a pathological example where our algorithm reaches a quadratic behavior. For disjoint disks of different radii, the analysis remains open.

[1]Supported by ANR grant Triangles (ANR-07-BLAN-0319).

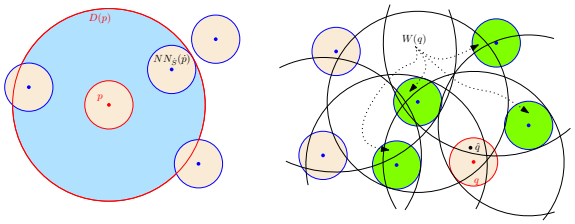


FIGURE 1. Definitions of $D(p)$ (big disk on the left) and $W(q)$ (darker disks on the right)

2 Algorithm

Notations

- Given a point set P in the plane, let DT_P denote its Delaunay triangulation, NN_P its nearest neighbor graph, and $NN_P(v)$ the nearest neighbor of $v \in P$ in $P \setminus \{v\}$.
- For a graph G and a vertex v of G, $d^\circ_G(v)$ is the degree of v in G.
- If p denotes an imprecise point, $\dot{p}$ denotes the center of p and $\hat{p}$ an instance of p. Let $S_k = \{p_1, p_2, \ldots, p_k\}$, $\dot{S}_k = \{\dot{p}_1, \ldots, \dot{p}_k\}$, $\hat{S}_k = \{\hat{p}_1, \ldots, \hat{p}_k\}$, $S = S_n$, $\dot{S} = \dot{S}_n$, $\hat{S} = \hat{S}_n$.
- In the case where S is a set of disjoint unit disks, given $p \in S$, we define $D(p)$ to be the disk with center $\dot{p}$ and radius $|\dot{p}NN_{\dot{S}}(\dot{p})| + 1$, that is, $D(p)$ is the interior of the circle centered at $\dot{p}$ tangent to the nearest disk in S and containing it (see Fig. 1 left).
- Given an instance $\hat{S}$, we also define $W(q) = \{p \in S \setminus \{q\} \,:\, \hat{q} \in D(p)\}$ (see Fig. 1 right).
- $|W|$ denotes the size of a set W.

Preprocessing. First we assume that the indices in $S = \{p_1, p_2, \ldots, p_n\}$ enumerate the disks in a random order (otherwise reorder the disks according to a random permutation).

We compute $DT_{\dot{S}}$ incrementally, inserting the points in the order of their indices. Furthermore, after inserting $\dot{p}_k$, we compute the index $h(k)$ such that $NN_{\dot{S}_k}(\dot{p}_k) = \dot{p}_{h(k)}$. The index $h(k)$ is called the *hint* for p_k. Using incremental randomized construction, it can be done in $O(n \log n)$ expected time [**6**] (including the computation of $h(k)$ for all k).

Instance processing. Now, given an instance $\hat{S}$, we compute $DT_{\hat{S}}$ incrementally, inserting the points in the order of their indices. The location of $\hat{p}_k$ in $DT_{\hat{S}_{k-1}}$ is done by a straight walk starting at $\hat{p}_{h(k)}$.

Complexity. *The expected cost of constructing $DT_{\hat{S}}$ is linear.*

Sketch of proof (complete proof in [**7**]). By usual backward analysis, it is enough to prove that the insertion of the last point $\hat{p}$ is done in expected constant time.

Let $\hat{x}$ be the starting point of the straight walk in $DT_{\hat{S}\setminus\{\hat{p}\}}$ to insert $\hat{p}$. As seen in the algorithm description, we have $\dot{x} = NN_{\dot{S}}(\dot{p})$. The cost of locating and inserting $\hat{p}$ is split in three parts:

- The cost c_1 of visiting the triangles incident to $\hat{x}$ in $DT_{\hat{S}\setminus\{\hat{p}\}}$ (to find the first one crossed by line segment $\hat{x}\hat{p}$),
- the cost c_2 of visiting the triangles crossed by line segment $\hat{x}\hat{p}$, and
- the cost c_3 of modifying the triangulation to update $DT_{\hat{S}\setminus\{\hat{p}\}}$ into $DT_{\hat{S}}$.

The cost c_3 is $d^\circ_{DT_{\hat{S}}}(\hat{p})$, since $\hat{P}$ is a random point in $\hat{S}$ the expected value of c_3 is less than 6. The cost c_1 is is $d^\circ_{DT_{\hat{S}\setminus\{\hat{p}\}}}(\hat{x}) \leq d^\circ_{DT_{\hat{S}}}(\hat{x}) + d^\circ_{DT_{\hat{S}}}(\hat{p})$. Although $\hat{x}$ is not a random point, it can be shown to be random enough to have an expected degree less than 36, thus the expected value of c_3 is less than 42. The cost c_2 is related to the average size of $|W(q)|$ for $q \in S$ and one can prove that the expected value of c_2 is less than 132. ◇

3 Experiments

Algorithms proposed in previous works [**2, 8**] are not implemented and are rather complicated, thus we have no doubt that our solution is better in practice. The aim of this section is to compare our algorithm and a direct computation of the Delaunay triangulation of the instance by a state-of-the-art algorithm without preprocessing.

For comparison, we use Sewchuk's code: Triangle [**9**] and the best implementations available in CGAL [**10**] where the points are ordered along a space-filling curve and then inserted in the *spatial sort* order [**5**]. The point location is done by a straight walk from the previous point (point location should be fast since it starts at a point nearby).

Point sets. We experiment on random disjoint unit disks. Each set contains n disjoint imprecise points in a $4\sqrt{n} \times 4\sqrt{n}$ square. Each center of an imprecise point is generated at random in the square, then the point is kept if its distance to previously kept points is greater than 1. Points are generated until a set of n points is obtained. A point instance is generated at random in each imprecise point. Point sets with size ranging between 10^3 and 10^8 points have been generated.

Platform. Experiments have been done on the following platform:

— CGAL 3.8 (`gcc 4.3.2` release mode),
— Linux-FC10,
— timings obtained with the `CGAL::Timer`,
— 3.00 GHz processor 32 GByte RAM.

Results. In the following table, we report the average time for a point insertion and the average number of triangles visited during the straight walks used in point locations.

2D random imprecise points	running time (μs) per point ♯ **visited triangles per point** ♯ cache miss ⋆ per point					
n	10^3	10^4	10^5	10^6	10^7	10^8
Shewchuk (divide & conquer)	0.98 5.3	1.12 2.3	1.05 2.8	1.67 20	2.34 39	
spatial sort + insert in order	1.2 **3.74** 14	0.92 **3.63** 3.6	0.90 **3.71** 3.1	0.98 **3.67** 5.3	1.12 **3.55** 7.7	1.36 **3.71**
locate from **hint** in random order	0.9 **2.83** 14	0.81 **2.80** 3.5	1.25 **2.77** 9.7	3.1 **2.75** 27	4.1 **2.75** 32	6.1 **2.74**
locate from **hint** in spatial sort order	0.9 **2.82** 14	0.78 **2.80** 3.5	0.64 **2.77** 3.3	0.66 **2.76** 3.8	0.67 **2.75** 3.8	0.79 **2.75**

⋆ number of cache miss obtained by `valgrind --tool=cachegrind` emulator

We first observe that the average numbers of triangles visited during the walk to locate a point from its hint (our method) or its predecessor in spatial sort (CGAL method) are both a small constant independent of the size of the point set. The number of visited triangles is around 2.8 for the hint and about 3.7 for spatial sort. Thus our theoretical linear complexity is confirmed by the experiments.

Unfortunately, a similar behavior is not observed for the running time. Our interpretation is that the random insertion order is demanding more and more to the cache memory management when the input size increases. Since the spatial sort order inserts the new point near the previous one, relevant triangles are already loaded in the cache memory and it reaches a better running time, even if the length of the walk is longer than for our method. We combine the advantages of both methods by using the spatial sort order to preprocess the imprecise points and to process the instance with our method. The results are satisfactory and our method is clearly faster than direct computation.

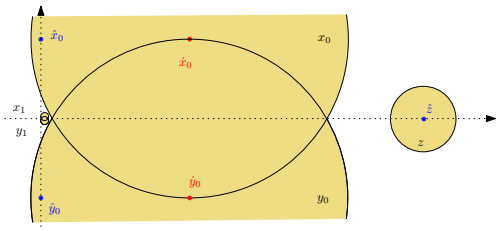


FIGURE 2. At most two disks overlapping with complexity $\Omega(n^2)$

4 Beyond disjoint unit disks in the plane

The algorithm does not need the disks to be disjoint nor have unit radius —these hypotheses are only useful for the proof of complexity. In fact, if we allow unit disks overlapping at most k times, the Delaunay triangulation of an instance is computed in $O(kn)$ time.

If the unit radius hypothesis is removed, the proof of complexity fails. Indeed, it is possible to design an example of n disks overlapping at most twice such that the algorithm takes quadratic time (see Fig. 2). In that example, the hint for x_i is y_i with probability $\frac{1}{2}$ (and vice versa), while $\hat{S}$ can be arranged close to the y axis so that there are i points between x_i and y_i.

The analysis extends to higher dimensions under additional hypotheses on the data, that are usual for random incremental construction. We get that, if S is such that, for a random sample R of size r, the expected sizes of $DT_{\dot{R}}$ and $DT_{\hat{R}}$ are both $O(r)$, then S can be preprocessed in $O(n \log n)$ time so that the Delaunay triangulation of an instance is computed in $O(n)$ time. Our experiments in 3D show a running time a bit smaller than spatial sort. The smallness of the gain is mainly due to the fact that, in 3D, the relative weight of point location compared to the modification of the data structure is smaller than in 2D.

Acknowledgments. The author thanks Sylvain Lazard for fruitful discussions.

References

[1] 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.

[2] Kevin Buchin, Maarten Löffler, Pat Morin, and Wolfgang Mulzer. Preprocessing imprecise points for delaunay triangulation: Simplified and extended. *Algorithmica*, 2011. http://www.cs.princeton.edu/~wmulzer/pubs/idt_algorithmica.pdf.

[3] Bernard Chazelle, Olivier Devillers, Ferran Hurtado, Mercè Mora, Vera Sacristán, and Monique Teillaud. Splitting a Delaunay triangulation in linear time. *Algorithmica*, 34:39–46, 2002. http://hal.inria.fr/inria-00090664.

[4] F. Chin and C. A. Wang. Finding the constrained Delaunay triangulation and constrained Voronoi diagram of a simple polygon in linear time. *SIAM J. Comput.*, 28:471–486, 1998. http://portal.acm.org/citation.cfm?id=299868.299877.

[5] Christophe Delage. Spatial sorting. In *CGAL User and Reference Manual*. CGAL Editorial Board, 3.8 edition, 2011. http://www.cgal.org/Manual/3.8/doc_html/cgal_manual/packages.html#Pkg:SpatialSorting.

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

[7] Olivier Devillers. Delaunay triangulation of imprecise points, preprocess and actually get a fast query time. Research Report 7299, INRIA, 2010. http://hal.inria.fr/inria-00485915.

[8] Maarten Löffler and Jack Snoeyink. Delaunay triangulation of imprecise points in linear time after preprocessing. *Comput. Geom. Theory Appl.*, 43:234–242, 2009. http://linkinghub.elsevier.com/retrieve/pii/S0925772109000583.

[9] J. R. Shewchuk. Triangle: Engineering a 2d quality mesh generator and Delaunay triangulator. http://www.cs.cmu.edu/~quake/triangle.html.

[10] Mariette Yvinec. 2D triangulations. In *CGAL User and Reference Manual*. CGAL Editorial Board, 3.8 edition, 2011. http://www.cgal.org/Manual/3.8/doc_html/cgal_manual/packages.html#Pkg:Triangulation2.

Geometric study of the weak equilibrium in a weighted case for a two-dimensional competition game

Javier Rodrigo[1], **M. Dolores López**[2]

[1] Departamento de Matemática Aplicada, E.T.S. de Ingeniería, Universidad Pontificia Comillas de Madrid
jrodrigo@upcomillas.es

[2] Departamento de Matemática e Informática Aplicadas a la Ingeniería Civil de la E.T.S.I. Caminos, Canales y Puertos, Universidad Politécnica de Madrid
marilo.lopez@upm.es

Abstract. This paper investigates an abstract game of competition between two players who want to earn the maximum number of points from a finite set of points in the plane. It is assumed that the distribution of these points is not uniform, so an appropriate weight to each position is assigned. The existence of Nash equilibrium and an approximate equilibrium in the game is studied by means of Computational Geometry techniques.

Introduction

This work generalizes the study of Nash equilibrium in a weighted competitive game with two players. This generalization is performed by means of a new definition of equilibrium that is weaker than the classical one ([**1**]).

The two players choose their positions t_1, t_2 in the plane to attract the largest possible weight of a given set of n weighted points in the plane $H = \{p_1, \ldots, p_n\}$. The perpendicular bisector of the players' locations partitions the plane into two regions. Each player is considered to capture those points which lie closer to his position than to that of the other player. The payoff of each player is then the total weight of the captured points ([**2, 5, 6, 7**]).

In formal terms, consider a strategic game $G = (N, X, \Pi)$, where $N = \{1, 2\}$ is the set of two players, $X = \mathbb{R}^2 \times \mathbb{R}^2$ is the strategy space, and Π^i, $i = 1, 2$ is the payoff function of each player.

The description of the payoff function in the game presented here is given by:

- $\Pi^1(t_1, t_2)$ = weight of the points p_i such that $d(p_i, t_1) \leq d(p_i, t_2)$;
- $\Pi^2(t_1, t_2)$ = weight of the points p_i such that $d(p_i, t_1) > d(p_i, t_2) = n - \Pi^1(t_1, t_2)$ if $t_1 \neq t_2$,

where $d(p_i, t)$ represents the Euclidean distance between the points p_i, t. In the case where $t_1 = t_2$, $\Pi^1(t_1, t_2) = \Pi^2(t_1, t_2) = \frac{n}{2}$ is defined.

Therefore, if we define the weight of a point p_i as $w(p_i) = k_i$, with $k_i > 0$ and $k_1 + \cdots + k_n = n$, then the payoff of the first player will be the sum of the weights of all positions located in the same half-plane as t_1, including the points on the bisector. The second player follows the same pattern, except for positions on the bisector.

CRM Documents, vol. **8**, Centre de Recerca Matemàtica, Bellaterra (Barcelona), 2011

Along the paper, the following definitions apply:

(1) The weight of a subset $\{p_{i_1}, \ldots, p_{i_k}\}$ of H is $w(\{p_{i_1}, \ldots, p_{i_k}\}) = \sum_{j=1}^{k} w(p_{i_j})$.
(2) m_i stands for the i-th weight bigger than $\frac{n}{2}$ that can be attained by a subset of H. For example, m_1 is the minimum weight bigger than $\frac{n}{2}$.
(3) C_α is the intersection of the convex hulls of the subsets of H with weight bigger than α.

The structure of the paper is as follows: In Section 1, necessary and sufficient conditions are presented for the Nash equilibrium positions to exist. Section 2 proposes an analogous study as previous section for a definition of equilibrium which is weaker than the classical one.

1 Nash equilibrium

Definition 1. A strategy profile (t_1^0, t_2^0) is a *Nash equilibrium* if $\Pi^1(t_1, t_2^0) \leq \Pi^1(t_1^0, t_2^0)$ and $\Pi^2(t_1^0, t_2) \leq \Pi^2(t_1^0, t_2^0)$ for all $(t_1, t_2) \in \mathbb{R}^2 \times \mathbb{R}^2$.

Necessary and sufficient conditions are presented for the Nash equilibrium positions to exist. These conditions are stated in terms of geometric features such as convex hulls ([**3**]). For the proofs of these conditions, see [**4**].

1.1 Necessary conditions

Let us see the following necessary condition on the payoff functions of the players:

Proposition 1. *In the presented game, the Nash equilibrium positions* (t_1, t_2) *must satisfy* $\Pi^1(t_1, t_2) = \Pi^2(t_1, t_2) = \frac{n}{2}$.

1.2 Necessary and sufficient conditions

Proposition 2. *There exist Nash equilibrium positions in the game if and only if* $C_{\frac{n}{2}}$ *is not the empty set. Furthermore, the equilibrium positions are the positions* (t_1, t_2) *such that* t_1, t_2 *are in the intersection.*

1.3 Uniqueness

We will now show that $C_{\frac{n}{2}}$ contains at most a single point, unless the n points of H lie on a single line.

Proposition 3. *If the* n *points of* H *are not collinear, then* $C_{\frac{n}{2}}$ *is either a point or the empty set.*

Remark. In the degenerate case where all the points are in a single line and there exists a combination of points with weight $\frac{n}{2}$, the intersection of the convex hulls may be an infinite set.

This analysis leads to the following conclusion:

Proposition 4. *The equilibrium in the present game, if it exists, is the unique point* (t, t) *for some* $t \in \mathbb{R}^2$. *In other words, both players will choose the same position, except in cases where the points of* H *are aligned.*

2 Weak equilibrium

To avoid the situation presented in previous section, a definition of equilibrium which is weaker than the classical one is proposed:

Definition 2. A strategy profile (t_1^0, t_2^0) is a *weak equilibrium* if $\Pi^1(t_1, t_2^0) \leq K_1$ and $\Pi^2(t_1^0, t_2) \leq K_2$ for all $(t_1, t_2) \in \mathbb{R}^2 \times \mathbb{R}^2$, where K_1 is the least weight greater than $\Pi^1(t_1^0, t_2^0)$ of a subset of H, and K_2 is the least weight greater than $\Pi^2(t_1^0, t_2^0)$.

A geometric analysis is developed here that extends the concepts presented in the previous section and underlies the search for the equilibrium positions, if they exist, according to the new definition.

Proposition 5. *In a weak equilibrium position (t_1^0, t_2^0), the following is necessarily satisfied: $\Pi^i(t_1^0, t_2^0) \geq n - m_1$, $i = 1, 2$.*

Remark. This proposition implies that the payoffs in a position of weak equilibrium must be of $n - m_1$ for a player and m_1 for the other one, or $\frac{n}{2}$ for both players. It can be noted that $n - m_1$ is the greatest payoff smaller than $\frac{n}{2}$ that a player can obtain.

The following proposition gives a necessary and sufficient condition for the existence of weak equilibrium positions:

Proposition 6. *If $m_1 < n$, then there exist weak equilibrium positions in the presented game if and only if $C_{m_1} \neq \varnothing$.*

Now Propositions 5 and 6 are applied to classify the weak equilibrium positions, as follows. The possible weak equilibrium positions in the proposed game are:

(1) (t_1, t_2) with t_1, t_2 belonging to C_{m_1}, $\Pi^i(t_1, t_2) = \frac{n}{2}$ for $i = 1, 2$.
(2) (t_1, t_2) with one of the points, say t_1, belonging to C_{m_1}, the other point, t_2, belonging to C_{m_2}, $\Pi^1(t_1, t_2) = m_1$, and $\Pi^2(t_1, t_2) = n - m_1$ (if there does not exist any combination of points of H with weight $\frac{n}{2}$).
(3) (t_1, t_2) with one of the points, say t_1, belonging to C_{m_2}, the other point, t_2, belonging to $C_{\frac{n}{2}}$, $\Pi^1(t_1, t_2) = n - m_1$, and $\Pi^2(t_1, t_2) = m_1$ (if there exists a combination of points of H with weight $\frac{n}{2}$).

Now the problem of determining the maximum number of points of H that can belong to C_{m_1} is considered. This maximum number gives a greater number of weak equilibrium positions in which at least one of the players chooses a position in H. This can be interesting for practical purposes. Concretely, the following statement is developed.

Find the maximum number of points of H that can belong to C_{m_1} for all the possible configurations of n points in the plane and all the possible assignations of weights to these points satisfying $m_2 < n$. This maximum number is denoted by max (the condition $m_2 < n$ implies that $n > 2$).

Now an upper bound for max that is bounded in n is given:

Proposition 7. *The inequality* $\max \leq 3$ *holds.*

In the following example of equally weighted n points with n an even number greater than four, this upper bound is attained, so $\max = 3$ for even n, $n > 4$.

Example. Let n be an even number greater than four, and consider a collection of $n - 3$ points $p_1, \ldots, p_{n-3}$ in convex position. If three new points p_{n-2}, p_{n-1}, p_n are located in the intersection of the convex hulls of the subsets of $\{p_1, \ldots, p_{n-3}\}$ with $\left[\frac{n-3}{2}\right] + 2$ points,

and a weight 1 is attached to each point, then the set of n points $H = \{p_1, \ldots, p_n\}$ has three points in C_{m_1}: If $\frac{n}{2} + 2$ points of H not containing every point of p_{n-2}, p_{n-1}, p_n are considered, then there will be at least $\frac{n}{2} = \left[\frac{n-3}{2}\right] + 2$ points of $p_1, \ldots, p_{n-3}$, so p_{n-2}, p_{n-1}, p_n are in the convex hull. Consequently, these three points are in the intersection of the convex hulls of the subsets of H with $\frac{n}{2} + 2$ points. This last set is C_{m_1}.

Remark. An analogous example with 3 points in C_{m_1} can be seen for odd n, $n > 5$: $n-3$ points $p_1, \ldots, p_{n-3}$ in the vertices of a regular polygon with weight $\frac{n}{n+1}$ each, a point p_{n-2} with weight $\frac{2n}{n+1}$ in the intersection of the convex hulls of the subsets of $\{p_1, \ldots, p_{n-3}\}$ with $\frac{n-3}{2}+2$ points, and two points p_{n-1}, p_n with weight $\frac{n}{n+1}$ each and in the intersection of the convex hulls of the subsets of $\{p_1, \ldots, p_{n-2}\}$ with $\left[\frac{n-2}{2}\right]+2$ points. Then for odd n, $n > 5$, we also have max = 3, so max = 3 for $n \geq 6$.

3 Conclusions

A discrete two-dimensional weighted competition model has been proposed and analyzed using geometric strategies that find the Nash equilibrium positions if they exist and ensure their uniqueness. In spite of this, Nash equilibrium in the majority of the situations studied has been found not to exist. To resolve this situation, a weakened definition of equilibrium has been presented, which ensures for each player that the other cannot improve his payoff by more than one quantity if he changes his position. This new definition of equilibrium can be useful in cases which have no Nash equilibrium. Also, to give the game a general scope, a weight to each point of the finite set the players fight for has been assigned.

The study of the existence and locations of Nash equilibrium and weak equilibrium positions has here been expanded in scope by applying techniques from computational geometry such as the intersection of convex hulls, which can be used because of the discrete nature of the game.

References

[1] M. Abellanas, M. López, J. Rodrigo, I. Lillo, Weak equilibrium in a spatial model, 2010, DOI: 10.1007/s00182-010-0241-y

[2] R. Aurenhammer, R. Klein, Voronoi diagrams, in: *Handbook of Computational Geometry*, Elsevier Science Publishers B.V., North-Holland, Amsterdam, 2000, 201–290.

[3] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf, *Computational Geometry – Algorithms and Applications*, 2nd ed., Springer, New York, 1997.

[4] I. Lillo, M. López, J. Rodrigo, A geometric study of the Nash equilibrium in a weighted case, *Applied Mathematical Sciences* **1** (2007), 2715–2725.

[5] A. Okabe, B. Boots, K. Sugihara, S. Chiu, *Spatial Tessellations Concepts and Applications of Voronoi diagrams*, John Wiley & Sons, Chichester, 2000.

[6] D. Serra, C. Revelle, Market capture by two competitors: The preemptive location problem, *Journal of Regional Science* **34** (1994), 549–561.

[7] M. Smid, Closest point problems in Computational Geometry, in: *Handbook of Computational Geometry*, Elsevier Science Publishers B.V., North-Holland, Amsterdam, 2000, 879–931.

Hyperbolic Delaunay triangulations and Voronoi diagrams made practical

Mikhail Bogdanov, Olivier Devillers, Monique Teillaud

INRIA Sophia Antipolis – Méditerranée, France
FirstName.LastName@inria.fr

Abstract. We show how to compute Delaunay triangulations and Voronoi diagrams of a set of points in hyperbolic space in a very simple way. While the algorithm follows from [**7**], we elaborate on arithmetic issues, observing that only rational computations are needed. This allows an exact and efficient implementation.

1 Introduction

As D. Eppstein states: "Hyperbolic viewpoint may help even for Euclidean problems" [**8**].[1] He gives two examples: the computation of 3D Delaunay triangulations of sets lying in two planes [**4**], and optimal Möbius transformation and conformal mesh generation [**3**]. Hyperbolic geometry is also used in applications like graph drawing [**9, 11**].

Several years ago, we showed that the hyperbolic Delaunay triangulation and Voronoi diagram can easily be deduced from their Euclidean counterparts [**4, 7**]. As far as we know, this was the first time when the computation of hyperbolic Delaunay triangulations and Voronoi diagrams was addressed. Since then, the topic appeared again in many publications. Onishi and Takayama write that they "rediscover the algorithm of [**7**]", in a way that they consider as more natural (i.e., their proofs rely only on computations) [**13**]. Nielsen and Nock transform the computation of the Voronoi diagram in the Klein model to the computation of an Euclidean power diagram [**12**]; however, even when the input sites have rational coordinates, the weighted points on which the power diagram is computed have algebraic coordinates. Other references can be found in [**15**] (which does not mention [**7**], though).

None of the above papers shows interest in practical aspects, especially arithmetic aspects, which are well known to be crucial for exactness and efficiency of implementations. In this paper, we show that our earlier approach allows to use only very simple arithmetic computations. Moreover, the proofs are purely geometric and avoid any computation. We first recall some background on hyperbolic geometry (Subsection 2.1) and on the space of circles (Subsection 2.2). Section 3 details algorithmic and arithmetic aspects of the computation of hyperbolic Delaunay triangulations and Voronoi diagrams. Section 4 gives a quick overview of the implementation.

Due to lack of space, the current presentation is restricted to the 2D case, but the results hold in any dimension.

This work was partially supported by the ANR (*Agence Nationale de la Recherche*) under the "Triangles" project of the *Programme blanc* (No. BLAN07-2_194137), http://www.inria.fr/sophia/geometrica/collaborations/triangles/.

[1] See also http://www.ics.uci.edu/~eppstein/pubs/geom-hyperbolic.html.

2 Background

2.1 The Poincaré disk

We refer the reader to basic geometry books for an introduction to hyperbolic geometry (e.g., [**2**, Chapter 19], [**16, 17**]). The transformations between the various models of the hyperbolic space are recalled in [**12**].

In the Poincaré disk model, the hyperbolic plane $\mathbb{H}^2$ is represented as the unit disk of $\mathbb{R}^2$. The set $\mathcal{H}_\infty$ of points of $\mathbb{H}^2$ at infinity is represented as the unit circle of $\mathbb{R}^2$. The set of finite points of $\mathbb{H}^2$ is the interior of the unit disk.

Hyperbolic lines are represented either as portions of lines of $\mathbb{R}^2$ that are orthogonal to $\mathcal{H}_\infty$ or as portions of circles of $\mathbb{R}^2$ that are orthogonal to $\mathcal{H}_\infty$. Hyperbolic circles are circles of $\mathbb{R}^2$ contained in the unit disk and that do not intersect $\mathcal{H}_\infty$. The hyperbolic center of a circle C is the unique point (or circle with null radius) in $\mathbb{H}^2$ of the linear pencil of circles of $\mathbb{R}^2$ defined by C and $\mathcal{H}_\infty$.

2.2 The space of circles

The space of circles sets up a correspondence between circles of $\mathbb{R}^2$ and points of $\mathbb{R}^3$ [**2**, Chapter 20], [**7**]: the circle C of $\mathbb{R}^2$ with center $c = (x_c, y_c)$ and radius r is associated to the point $\sigma_C = (x_c, y_c, z_c = x_c^2 + y_c^2 - r^2)$ in $\mathbb{R}^3$. A point $p = (x_p, y_p)$ of $\mathbb{R}^2$, seen as a circle of null radius, is thus associated to the point $\sigma_p = (x_p, y_p, x_p^2 + y_p^2)$ on the unit paraboloid $\Pi : (z = x^2 + y^2)$ of $\mathbb{R}^3$.

In the space of circles, we are considering polarity relatively to Π, i.e., orthogonality with respect to the symmetric bilinear form $\phi_\Pi((x_1, y_1, z_1), (x_2, y_2, z_2)) = x_1x_2 + y_1y_2 - \frac{1}{2}(z_1 + z_2)$ associated to the quadratic form $Q_\Pi(x, y, z) = x^2 + y^2 - z$. In this setting, for a circle C of $\mathbb{R}^2$, π_C denotes the polar hyperplane of σ_C in $\mathbb{R}^3$; each point $\sigma_{C'}$ of π_C represents a circle C' that is orthogonal to C. For a point $p \in \mathbb{R}^2$, π_p is the hyperplane tangent to Π at point $\sigma_p \in \Pi$; each point $\sigma_{C'}$ of π_p represents a circle C' that passes through p. The unit circle $\mathcal{H}_\infty$ of $\mathbb{R}^2$ (i.e., the infinite line of $\mathbb{H}^2$) is represented as the point σ_∞ of coordinates $(0, 0, -1)$ in $\mathbb{R}^3$. Its polar hyperplane is the plane π_∞ of equation $(z = 1)$ in $\mathbb{R}^3$.

3 Algorithmic and arithmetic aspects

Let $\mathcal{P}$ be a set of points in $\mathbb{H}^2$, i.e., in the unit disk of $\mathbb{R}^2$. We assume coordinates of points in $\mathcal{P}$ to be rational.

To compute the hyperbolic Delaunay triangulation $DT_\mathbb{H}(\mathcal{P})$ of $\mathcal{P}$, it is enough to compute the Euclidean Delaunay triangulation $DT_\mathbb{R}(\mathcal{P})$, and to filter out the triangles of $DT_\mathbb{R}(\mathcal{P})$ whose circumscribing circle intersects $\mathcal{H}_\infty$. It directly follows that the complexity of computing $DT_\mathbb{H}(\mathcal{P})$ is of the same order as $DT_\mathbb{R}(\mathcal{P})$, namely $\Theta(n \log n)$.

Many standard algorithms allow to compute $DT_\mathbb{R}(\mathcal{P})$ using only rational computations, since the two elementary predicates *orientation*(p, q, r) and *in_circle*(p, q, r, s) boil down to computing signs of determinants on the coordinates of the points p, q, r, s. Testing whether a circle defined by two rational points intersects the unit circle $\mathcal{H}_\infty$ is also done by rational computations only. So, the combinatorial part of $DT_\mathbb{H}(\mathcal{P})$ is easily computed using only rational computations.

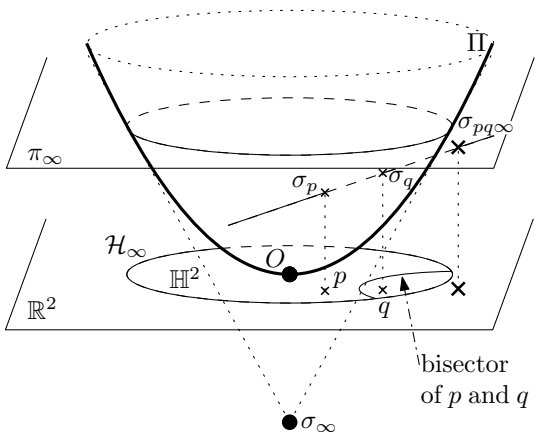
Let us now focus on the geometric embedding of $DT_\mathbb{H}(\mathcal{P})$ and of the dual Voronoi diagram $VD_\mathbb{H}(\mathcal{P})$. We first remark that, straightforwardly from the above definition, a

circle $C \subset \mathbb{R}^2$ has rational equation if and only if its associated point σ_C in the space of circles has rational coordinates.

For a triangle pqr of $DT_{\mathbb{H}}(\mathcal{P})$, the hyperbolic edge $e_{\mathbb{H}}(pq)$ is the arc between p and q of the hyperbolic line through p and q, i.e., the circle through p and q that is orthogonal to $\mathcal{H}_\infty$. The set of circles passing through p and q is the line $\delta_{pq} = \pi_p \cap \pi_q$ in the space of circles, and δ_{pq} is also the polar line of line $(\sigma_p\sigma_q)$. The circle through p and q that is orthogonal to $\mathcal{H}_\infty$ is thus associated to the intersection $\delta_{pq} \cap \pi_\infty$. So, $DT_{\mathbb{H}}(\mathcal{P})$ can be geometrically embedded using only rational computations.

Proposition 3.1. *The bisector of two points of $\mathcal{P}$ is a hyperbolic line, whose equation in $\mathbb{R}^2$ is rational.*

Proof. Let p and q be two points of $\mathcal{P}$, i.e., rational points of $\mathbb{R}^2$ inside the unit disk. We are going to construct their hyperbolic bisector as the locus of hyperbolic centers of circles passing through p and q.



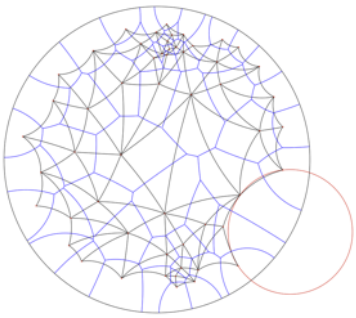
The hyperbolic center of a given circle $\sigma_C \in \delta_{pq}$ is the intersection of the line $\sigma_C\sigma_\infty$ with Π, so the locus of such centers is the intersection $E_{pq\infty}$ with Π of the plane $P_{pq\infty}$ containing δ_{pq} and σ_∞. The polar point $\sigma_{pq\infty}$ of $P_{pq\infty}$ represents a circle, so $E_{pq\infty}$ is in fact associated to the set of points of $\mathbb{R}^2$ on this circle. And $\sigma_{pq\infty}$ is the intersection of the polars of σ_∞ and δ_{pq}, i.e., the intersection $\pi_\infty \cap (\sigma_p\sigma_q)$.

To complete the proof, it is enough to notice that all geometric constructions above manipulate only rational objects: rational points, and intersection of a rational plane with a rational line. □

A vertex of $VD_{\mathbb{H}}(\mathcal{P})$ is the intersection of two hyperbolic bisectors, i.e., the intersection of two circles whose equation is rational, so its coordinates are algebraic numbers of degree 2. A Voronoi vertex can alternatively be computed in the following way: it is the hyperbolic center of a circle C_{pqr} circumscribing a Delaunay triangle pqr, associated with point σ_{pqr} in the space of circles; the hyperbolic center of C_{pqr} is thus the intersection $(\sigma_{pqr}\sigma_\infty) \cap \Pi$ (from the last sentence of Subsection 2.1). Infinite edges of the Voronoi diagram intersect $\mathcal{H}_\infty$ in points whose coordinates are also algebraic numbers of degree 2. So, all edges of $VD_{\mathbb{H}}(\mathcal{P})$ are arcs of rational circles whose endpoints are algebraic numbers of degree 2.

4 Implementation

The algorithm was implemented using CGAL [**1**]. The class `Delaunay_hyperbolic_triangulation_2` derives from the class `CGAL::Delaunay_triangulation_2<GT>`, which computes $DT_{\mathbb{R}}(\mathcal{P})$ [**18**]. It just has to mark all triangles of $DT_{\mathbb{R}}(\mathcal{P})$ that are not in $DT_{\mathbb{H}}(\mathcal{P})$. The template parameter `GT` is the *geometric traits*, which provides the algorithm with elementary predicates and constructions.



Only predicates are used for the computation of $DT_{\mathbb{R}}(\mathcal{P})$. They do not need to be modified to compute $DT_{\mathbb{H}}(\mathcal{P})$, which is thus computed exactly and efficiently using filtered rational arithmetics. Our preliminary experiments show an extra cost of 9% to extract $DT_{\mathbb{H}}(\mathcal{P})$ from $DT_{\mathbb{R}}(\mathcal{P})$ (i.e., $DT_{\mathbb{H}}(\mathcal{P})$ is constructed in less than 1 sec for 1 million points, on a MacBookPro 2.6 GHz). Only geometric embeddings need constructions, which must be replaced in `GT` by constructions presented in Section 3; if they are only used for drawing, they do not need to be exact, but some CGAL, CORE or LEDA exact algebraic numbers can be used if necessary.

One of our goals is to compute hyperbolic periodic Delaunay triangulations, which appears much more difficult than in the Euclidean setting [**6**], even for the simplest case of a group of four hyperbolic translations in $\mathbb{H}^2$ [**5**, Section 4.4], which would already be useful for various applications [**14, 10**].

References

[1] CGAL, Computational Geometry Algorithms Library. `http://www.cgal.org`.

[2] M. Berger. *Geometry* (vols. 1-2). Springer-Verlag, 1987.

[3] M. Bern and D. Eppstein. Optimal Möbius transformations for information visualization and meshing. In *7th Workshop on Algorithms and Data Structures*, volume 2125 of *Lecture Notes in Comp. Sci.*, Providence, Rhode Island, USA, 2001. Springer-Verlag. `http://arxiv.org/abs/cs.CG/0101006`.

[4] Jean-Daniel Boissonnat, André Cérézo, Olivier Devillers, and Monique Teillaud. Output-sensitive construction of the Delaunay triangulation of points lying in two planes. *Internat. J. Comput. Geom. Appl.*, 6(1):1–14, 1996. `http://www-sop.inria.fr/prisme/publis/bcdt-oscdt-96.ps.gz`.

[5] Manuel 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/`.

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

[7] Olivier Devillers, Stefan Meiser, and Monique Teillaud. The space of spheres, a geometric tool to unify duality results on Voronoi diagrams. In *Proc. 4th Canad. Conf. Comput. Geom.*, pages 263–268, 1992. Long version: Report 1620, INRIA, 1992, `http://hal.inria.fr/inria-00074941`.

[8] D. Eppstein. Hyperbolic geometry, Möbius transformations, and geometric optimization, 2003. Invited talk at MSRI Introductory Workshop on Discrete and Computational Geometry.

[9] D. Eppstein and M. T. Goodrich. Succinct greedy graph drawing in the hyperbolic plane. In *Proc. 16th Int. Symp. Graph Drawing*, volume 5417 of *Lecture Notes in Computer Science*, pages 14–25, Heraklion, Crete, 2008. IEEE Transactions on Computing, `http://arxiv.org/abs/0806.0341`.

[10] G. Faye, P. Chossat, and O. Faugeras. Some theoretical results for a class of neural mass equations. Technical report, 2010. `http://arxiv.org/abs/1005.0510`.

[11] Tamara Munzner. Exploring large graphs in 3D hyperbolic space. *IEEE Computer Graphics and Applications*, 18(4):18–23, 1998.

[12] Frank Nielsen and Richard Nock. Hyperbolic Voronoi diagrams made easy. In *Proc. 10th International Conference on Computational Science and its Applications*, pages 74–80, Los Alamitos, CA, USA, 2010. IEEE Computer Society. `http://doi.ieeecomputersociety.org/10.1109/ICCSA.2010.37`.

[13] K. Onishi and N. Takayama. Construction of Voronoi diagrams on the upper half-plane. *IEICE Trans. Fundamentals*, E79-A(4):533–539, 1996.

[14] Guodong Ron, Miao Jin, and Xiaohu Guo. Hyperbolic centroidal Voronoi tessellation. In *Proc. ACM Symposium on Solid and Physical Modeling*, pages 117–126, Haifa, Israel, 2010.

[15] Toshihiro Tanuma, Hiroshi Imai, and Sonoko Moriyama. Revisiting hyperbolic Voronoi diagrams from theoretical, applied and generalized viewpoints. *International Symposium on Voronoi Diagrams in Science and Engineering*, pages 23–32, 2010.

[16] W. P. Thurston. Three dimensional manifolds, Kleinian groups, and hyperbolic geometry. *Bull. Amer. Math. Soc.*, 6(3):357–381, 1982.

[17] P. M. H. Wilson. *Curved Spaces*. Cambridge University Press, Cambridge, 2008.

[18] Mariette Yvinec. *2D Triangulations*, 3.8 edition, 2010. CGAL User and Reference Manual.

Improving shortest paths in the Delaunay triangulation

Manuel Abellanas[1], Mercè Claverol[2], Gregorio Hernández[1], Ferran Hurtado[3], Vera Sacristán[3], Maria Saumell[3], Rodrigo I. Silveira[3]

[1] Facultad de Informática, Universidad Politécnica de Madrid
{mabellanas,gregorio}@fi.upm.es

[2] Departament de Matemàtica Aplicada IV, Universitat Politècnica de Catalunya
merce@ma4.upc.edu

[3] Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya
{ferran.hurtado,vera.sacristan,maria.saumell,rodrigo.silveira}@upc.edu

Abstract. We study a problem about shortest paths in Delaunay triangulations. Given two nodes s, t in the Delaunay triangulation of a point set P, we look for a new point p that can be added, such that the shortest path from s to t in the Delaunay triangulation of $P \cup \{p\}$ improves as much as possible. We study properties of the problem and give efficient algorithms to find such a point when the graph-distance used is Euclidean and for the link-distance. Several other variations of the problem are also discussed.

1 Introduction

There are many applications involving communication networks where the underlying physical network topology is not known, too expensive to compute, or there are reasons to prefer to use a logical network instead. An example of an area where this occurs is ad-hoc networks, where nodes can communicate with each other when their distance is below some threshold. Even though the routing is done locally, to avoid broadcasting to all neighbors every time a packet needs to be sent, some logical network topology and routing algorithm must be used.

The Delaunay triangulation is often used to model the overlay topology [**4, 6**] due to several advantages: it provides locality, scales well, and in general avoids high-degree vertices, which can create serious bottlenecks. In addition, several widely-used *localized* routing protocols guarantee to deliver the packets when the underlying network topology is the Delaunay triangulation [**3**]. Furthermore, there are localized routing protocols based on the Delaunay triangulation where the total distance traveled by any packet is never more than a small constant factor times the network distance between source and destination (e.g., [**3**]). Since the Delaunay triangulation is known to be a spanner [**5**], in the case of geometric networks this guarantees that all packets travel at most a constant times the minimum travel time.

In this paper we consider the problem of improving a geometric network, with a Delaunay triangulation topology, by augmenting it with additional nodes. In particular, we aim at improving the shortest path on the Delaunay network between two given nodes s and t. Adding new nodes to a Delaunay network produces changes in the network topology that can result in equal, shorter, or longer shortest paths between s and t (an example where adding a point shortens the path is shown in Figure 1, left).

CRM Documents, vol. **8**, Centre de Recerca Matemàtica, Bellaterra (Barcelona), 2011

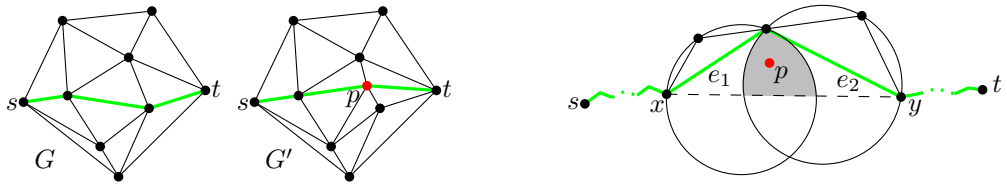


FIGURE 1. Left: shortest path between s and t before and after adding p; the latter is shorter. Right: any point inserted in the shaded region, like p, improves $SP(s,t)$, shown in green.

We restrict ourselves to the scenario where at most one node can be added to the network, which can be placed anywhere on the plane. The goal is to find a location for the new node that improves the shortest path between s and t as much as possible. We are not aware of any previous work on this problem.

Notation. The input to the problem is a set of n points $P = \{p_1, \ldots, p_n\}$, and two distinguished points $s, t \in P$. The points represent the locations of the network nodes.

We will use G to denote the Delaunay graph of P. Thus, G has the points in P as vertices, and an edge (p_i, p_j) between two vertices if and only if there is a circle through p_i, p_j that does not contain any point from P in its interior. We assume that the points in P are in general position: no three points are collinear and no four points are cocircular. Hence G represents the Delaunay triangulation of P. Moreover, we also assume that the edge $(s,t) \notin G$, otherwise the distance between s and t in G would be optimal. Note that we use G to refer to both the graph and the triangulation.

The shortest path on G between s and t will be denoted by $SP_G(s,t)$. The length of such path, defined as the sum of the Euclidean lengths of its edges, will be denoted by $|SP_G(s,t)|$, although we will omit G if possible. The straight line segment between two points x and y will be denoted by $\overline{xy}$, and its Euclidean length by $|\overline{xy}|$.

Finally, we will use G'_p to denote the Delaunay graph of $P \cup \{p\}$, for some $p \notin P$ (we will omit p when clear from the context).

2 Properties and observations

We begin the paper by analyzing some geometric properties of the problem.

When a new point p is inserted in P, some edges of the Delaunay triangulation might disappear and new edges, all incident to p, appear. The edges of G that are affected by the insertion of p belong to Delaunay triangles whose circumcircles contain p. In particular, all triangles in G whose circumcircle contains p get new edges in G', connecting their vertices to p. If p is outside the convex hull of P, some additional edges might appear.

A first question that one may ask is whether it is always possible to improve a shortest path by adding one point to G. There are situations in which it is easy to obtain *some* improvement:

Lemma 2.1. *Let e_1 and e_2 be two consecutive edges of $SP_G(s,t)$. Let C_1 and C_2 be two Delaunay circles through the extremes of e_1 and e_2, respectively. If C_1 and C_2 are not tangent and $C_1 \cap C_2$ is on the side in which the edges form the smallest angle, then the length of $SP_G(s,t)$ can always be reduced by inserting one point.*

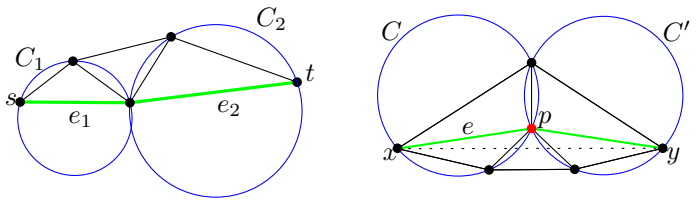


FIGURE 2. Left: example where $SP(s,t)$ cannot be improved by adding one point. Right: one of the situations described in Lemma 3.1.

Even though we omit the proof of the previous lemma, the idea can be seen in Figure 1 (right). If we insert p in the shaded region, then (x,p) and (p,y) will be Delaunay edges in G', shortening the portion of $SP(s,t)$ between x and y. However, some shortest paths cannot be improved by adding one point, as shown in Figure 2 (left).

Lemma 2.2. *It is sometimes impossible to improve $SP_G(s,t)$ by inserting only one point.*

On the other hand, it is easy to see that two points always suffice to improve $SP(s,t)$ (details are given in [**1**]).

3 Finding a point that gives the maximum improvement

Next we present an algorithm that computes a point p such that $|SP_{G'_p}(s,t)|$ is minimum. The correctness of the algorithm is based on the following lemmas, proved in [**1**]. We use the facts that we only need to look at $O(k)$ possible candidate points, where k is the number of pairs of Delaunay circles that intersect ($k \in \Theta(n^2)$ in the worst case) and that the shortest paths from s and t need to be computed only once.

Lemma 3.1. *Let p be a point that gives the maximum improvement, and let x, y be the points in G such that $SP_{G'_p}(s,t)$ includes (x,p) and (p,y) as consecutive edges. Then p lies on the segment $\overline{xy}$, or on the intersection of a Delaunay circumcircle through x, and another through y.*

Lemma 3.2. *Let p be a point, and let $x \in P$ such that $(x,p) \in G'_p$. If $SP_{G'_p}(s,p)$ includes (x,p), then $|SP_{G'_p}(s,p)| = |SP_G(s,x)| + |\overline{xp}|$. Otherwise, $|SP_{G'_p}(s,p)| \leq |SP_G(s,x)| + |\overline{xp}|$.*

Algorithm. The previous lemmas imply that in order to find an optimal point p it suffices to analyze each pair of intersecting Delaunay circles of G. We first precompute the shortest path trees from s and from t. Then we use an output-sensitive algorithm to compute all pairs of intersecting circles.

For each pair of intersecting circles (C_1, C_2), we proceed as follows. Each circle corresponds to a Delaunay triangle from G. Let the two triangles be t_1, t_2. For each pair of vertices $x \in t_1$ and $y \in t_2$, we first check if $\overline{xy}$ intersects $C_1 \cap C_2$. If it does, we take p as any point on $(\overline{xy} \cap C_1 \cap C_2)$. Otherwise, we check the two points where C_1 and C_2 intersect, and use the one that gives the shortest path from x to y. See Figure 2 (right).

If the length of $SP(s,t)$ improves by using p, we update this information. In the end we output the point that gave the shortest path, if it improves over $SP_G(s,t)$, or report that no point can improve it.

The running time of the algorithm is dominated by the time needed to find all pairs of intersecting circles. Using an algorithm like Balaban's [**2**], we obtain an $O(n \log n + k)$ running time, with $O(n)$ space.

Theorem 3.3. *Given the Delaunay triangulation of n points, and two vertices s and t, a point whose insertion gives the maximum improvement in $SP(s,t)$ can be found in $O(n \log n + k)$ time, where k is the number of pairs of intersecting Delaunay circles.*

Related problems. We show in [**1**] that, based on the previous lemmas, some other related problems can also be solved efficiently. In addition, a different proximity graph can be used (e.g., Gabriel or nearest neighbor graph) by applying the general approach of partitioning the plane into regions such that inserting a point anywhere inside one region produces the same topological change to the structure.

4 Using the link-distance

An interesting variant of the problem arises when the metric used to measure distances on G is the link-distance: the length of a path is defined by its number of edges. This metric is also interesting in networking applications, since it measures the number of hops. As before, we are interested in adding one new point to G such that the link-distance between s and t is minimized as much as possible. Using a data structure based on additively-weighted Voronoi diagrams of the circle centers, this problem can be solved more efficiently than the previous one. In [**1**] we prove:

Theorem 4.1. *Given the Delaunay triangulation of n points, and two vertices s and t, a point whose insertion gives the maximum improvement in $SP(s,t)$, under the link-distance, can be found in $O(n \log^2 n)$ time.*

5 Discussion

We have studied the problem of adding a point to a Delaunay triangulation, such that it improves a shortest path as much as possible. In [**1**] we also show how to solve several other related problems. Perhaps the most intriguing question left open is whether the decision problem (Is there a point that improves $SP(s,t)$?) can be solved faster than the optimization problem.

Acknowledgments. Research initiated during the 6th Iberian Workshop on Computational Geometry, held in Aveiro. M.A. and G.H. were partially supported by project MTM2008-05043. M.C., F.H., V.S. and M.S. were partially supported by projects MTM-2009-07242 and Gen. Cat. DGR 2009SGR1040. M.A., G.H., M.C. and F.H. were also supported by project HP2008-0060. R.I.S. was supported by the Netherlands Organisation for Scientific Research (NWO).

References

[1] M. Abellanas, M. Claverol, G. Hernández, F. Hurtado, V. Sacristán, M. Saumell, and R. I. Silveira, Improving shortest paths in the Delaunay triangulation, manuscript in preparation.

[2] I. J. Balaban, An optimal algorithm for finding segments intersections, in *Proc. SoCG'95*, 1995, 211–219.

[3] P. Bose and P. Morin, Online routing in triangulations, *SIAM J. Comput.*, **33** (2004), 937–951.

[4] E. Buyukkaya and M. Abdallah, Efficient triangulation for P2P networked virtual environments, in *Proc. NetGames'08*, 2008, 34–39.

[5] D. P. Dobkin, S. J. Friedman, and K. J. Supowit, Delaunay graphs are almost as good as complete graphs, *Discrete Comput. Geom.*, **5** (1990), 399–407.

[6] J. Liebeherr, M. Nahas, and W. Si, Application-layer multicasting with Delaunay triangulation overlays, *IEEE J. Sel. Areas Comm.*, **20** (2002), 1472–1488.

Compact grid representation of graphs

J. Cáceres[1], C. Cortés[2], C. I. Grima[2], M. Hachimori[3], A. Márquez[2], R. Mukae[4], A. Nakamoto[4], S. Negami[4], R. Robles[2], J. Valenzuela[2]

[1] University of Almeria (Spain)
jcaceres@ual.es

[2] University of Seville (Spain)
{ccortes,grima,almar,rafarob,jesusv}@us.es

[3] Tsukuba University (Japan)
hachi@sk.tsukuba.ac.jp

[4] Yokohama National University (Japan)
{mkerij,nakamoto,negami}@edhs.ynu.ac.jp

Abstract. A grid representation of a graph maps vertices to grid points and edges to line segments that avoid grid points but the extremes. We study how many lines of the plane are needed to give a grid representation of a given graph.

Introduction

Graph drawing applies topology and geometry to derive suitable representations of graphs. Particularly, grid representations of graphs have attracted the attention of many researchers (see, for example **[3, 5, 7]**).

A *grid point* is a point of the plane having integer coordinates. A line segment S joining two grid points is said to be *primitive* if the only grid points in s are its extremes.

A graph G is said to be *grid locatable* (or *locatable* for short) if each vertex is represented by a grid point and each edge by a primitive segment joining its extremes; in other words, G is a subgrah of the visibility graph of the integer lattice (in the sense defined, for example, in **[9]**). The graph G is said to be *properly embeddable in grid* (*p-embeddable* for short) if the segments representing edges do not cross each other.

Independently in **[6, 8]** the following characterization is shown

Theorem 0.1. **[6, 8]** *A graph G is locatable if and only if G is 4-colorable.*

In their proofs **[6, 8]**, they show that only 4 lines are needed in order to represent any 4-colorable graph. However, of course, not always the four lines are needed. In this work, we deal with this problem, namely, how many lines are needed to locate or p-embed a given graph? In fact, as we just have mentioned, in both papers, an upper bound to the number of lines needed to locate a graph is given.

Corollary 0.2. **[6, 8]** *A graph G with chromatic number $\chi(G) \leq 4$ can be located in, at most, $\chi(G)$ lines.*

We shall see here that the upper bound provided by Corollary 0.2 is not optimal in some cases.

A trivial but useful observation is that an edge joining (x_1, y_1) and (x_2, y_2) is primitive if and only if $|x_1 - x_2|$ and $|y_1 - y_2|$ are relatively prime to each other.

CRM Documents, vol. **8**, Centre de Recerca Matemàtica, Bellaterra (Barcelona), 2011

1 p-embedding a graph in the grid

Although it is not known whether any planar graph can be p-embedded in the plane (see [**6**]), some partial results are known. Thus, in [**8**] it is proven that any outerplanar graph can be p-embedded in the grid. Even more, we can prove the following result

Proposition 1.1. *Any plane bipartite graph can be p-embedded in the grid.*

Proof. In [**1**] it is given a method to embed a quadrangulation in the grid. That method is based on two 2-book representations of any quadrangulation, and they assign to each vertex the coordinates as its relative positions in the spines of both 2-books. But we can choose the x-coordinate of each point verifying that all the differences of x-coordinates being relatively prime with the other ones. In this way, we guarantee the condition of primitiveness. □

Additionally, a complete characterization can be given on the graphs that can be p-embedded into two lines.

Proposition 1.2. [**4**] *A graph G is p-embeddable into two lines if and only if G can be extended to a maximal outerplanar graph such that its dual is a path.*

It is known that, if a graph is locatable in the plane, then it is locatable in at most four lines. However, it is hopeless to try to obtain a similar result regarding p-embeddability. This can be obtained since it is well known that some graphs need linear grids in both dimensions to be embedded in the plane. But if we deal with primitive segments, we can obtain a similar result even for trees.

Lemma 1.3. *Given an integer number n, it is possible to find a tree that cannot be p-embedded in n lines.*

Proof (sketch). It is not difficult to see that the number of lines needed to p-embed any balanced n-ary tree tends to infinity as n goes to infinity. □

Thus the main question remaining open in this section is whether it is possible to p-embed any planar graph in the plane.

2 Locating a graph in the grid

Given that it is known that a graph is locatable if and only if it is 4-colourable (see [**6, 8**]), in this section we consider minimizing the number of lines in such a representation. In other words, given a concrete 4-colorable graph, how many lines are needed to represent it?

Although Theorem 0.1 gives an upper bound for the number of lines needed to represent a graph, that bound is of course not always optimal. For instance,

Proposition 2.1. *Any outerplanar graph can be located in two lines.*

Proof (sketch). The proof can be obtained by induction, taking into account that any maximal outerplanar graph has always an ear. Figure 1 shows how to add an ear to an edge that is in the outerface (the tiny triangles in the central image mean that one of those points is in the outerface; then we move —if needed— all the drawing in order to leave room for the new vertex). □

In fact, we can describe the structure of any graph locatable in two or three lines.

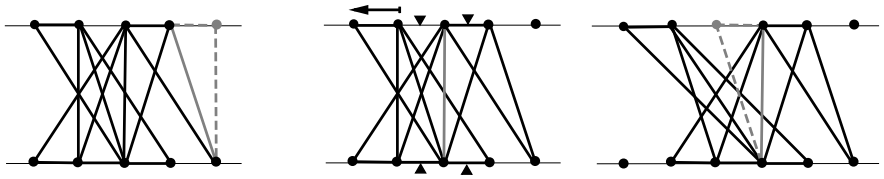


FIGURE 1. How to add a vertex of degree 2.

Theorem 2.2. *Let $G = (V, E)$ be a graph such that $\chi(G) \leq 4$. Then,*

(1) *G is locatable in two lines if and only if V can be partitioned into two subsets V_1, V_2, such that the subgraph of G induced by V_i is a disjoint union of paths.*
(2) *G is locatable in three lines if and only if V can be partitioned into three subsets V_1, V_2, V_3, such that the subgraph of G induced by V_1 is a disjoint union of paths, and V_2 and V_3 are independent sets.*

We can use Theorem 2.2 to obtain examples of planar graphs that need the four lines of Theorem 0.1 to be located in the plane. For instance, the graph depicted in Figure 2. Basically there is only one 4-coloring and there is no possible assignment of the colors avoiding vertices of degree 3 in the subgraph induced by two colors.

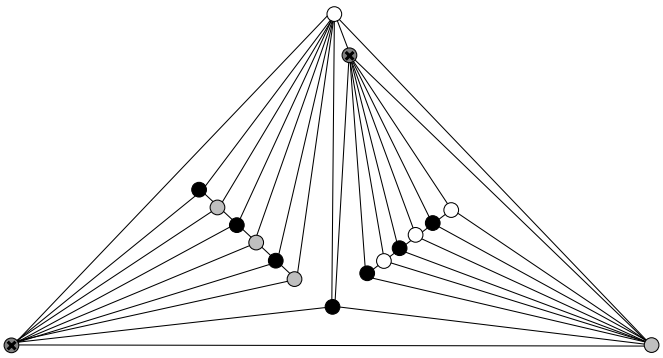


FIGURE 2. A planar graph that needs four lines to be located.

Additionally, Theorem 2.2 is one of the main tools used to prove the following results.

Theorem 2.3. *If $\Delta(G) \leq 3$ (the maximum degree of G), then G is locatable in two lines.*

Proof (sketch). We have to obtain a partition of the vertices in G verifying the hypothesis of Theorem 2.2 (1). In a first step, if G is not K_4, that can be located trivially in two lines, we color G with 3 colors. Then, if there are vertices of degree 3 in the subgraph of G induced by colors 1 and 2, this means that there is a vertex v with the color 1 with three neighbors with the color 2 (or the other way around). In that case, we can change the color of v to 3.

So, if we assume that there is not vertex of degree 3 in the subgraph of G induced by colors 1 and 2, we have to check that there is no cycle in such a subgraph. If such a cycle exists, we can change the color of one of the vertices to a new color 4. The only problem could be now to find a vertex of degree 3 in the subgraph of G induced by colors 3 and 4,

but that case can be avoided again by changing the color of that vertex to 1. After all this process, V_1 will have all the vertices with color 1 or 2 and V_2 the vertices with color 2 or 4. □

Observe that, in the previous theorem, G could be non-planar. It is possible to relax the condition on the maximum degree of G, but adding the additional hypothesis of planarity.

Theorem 2.4. *Let G be a planar graph with $\Delta(G) \leq 4$. Then G is locatable in three lines.*

Proof (sketch). The proof is in some way similar to the proof of Theorem 2.3. Starting with an actual embedding of $G = (V, E)$, we complete it by adding extra dummy edges to the faces of even length. In this way, we are sure that in a 4-coloring of G we have no face with only two colors. We are going to change this initial coloring in order to obtain a partition of V fulfilling the conditions of Theorem 2.2 (2). We assign vertices with colors 1 or 2 to V_1, vertices with color 3 to V_2 and vertices with color 4 to V_3. As in Theorem 2.3, with a simple change, we can avoid vertices of degree 3 or 4 in the subgraph $\langle V_1 \rangle$ induced by V_1. In the case of a cycle, we can change the colors using Kempe's chains starting in the cycles containing the most points in their interior. Now, some other cycles can be obtained, but those cycles are going to have less points in their interior, so we can apply again the same method, if needed. □

References

[1] L. Barriere, C. Huemer. 4-Labelings and grid embeddings of plane quadrangulations. In *Graph Drawing. 17th International Symposium (GD 2009)*. LNCS Vol. 5849, Springer-Verlag, 2009.

[2] R. L. Brooks. On colouring the nodes of a network. *Proc. Cambridge Philosophical Society, Math. Phys. Sci.* **37** (1941), 194–197.

[3] M. Chrobak and S. Nakano. Minimum width grid drawings of plane graphs. *Computational Geometry* **11(1)** (1998), 29–54.

[4] S. Cornelsen, T. Schank, and D. Wagner. Drawing graphs on two and three lines. In *M. Goodrich, S. Kobourov, eds.: Graph Drawing, 10th International Symposium (GD 2002)*. LNCS Vol. 2528, Springer-Verlag, 2002, 31–41.

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

[6] D. Flores and F. J. Zaragoza. Every four-colorable graph is isomorphic to a subgraph of the visibility graph of the integer lattice. In *CCCG 2009*, 17–19.

[7] J. Kára, A. Pór, and D. R. Wood. On the chromatic number of the visibility graph of a set of points in the plane. *Discrete and Computational Geometry* **34(3)** (2005), 497–506.

[8] A. Nakamoto and S. Negami. Proper grid representation of graphs. Preprint.

[9] A. Por and D. R. Wood. On visibility and blockers, *J. Computational Geometry* **1(1)** (2010), 29–40.

Large angle crossing drawings of planar graphs in subquadratic area

Patrizio Angelini[1], Giuseppe Di Battista[1], Walter Didimo[2], Fabrizio Frati[1,3], Seok-Hee Hong[4], Michael Kaufmann[5], Giuseppe Liotta[2], Anna Lubiw[6]

[1] Dipartimento di Informatica e Automazione, Roma Tre University, Italy
{angelini,gdb,frati}@dia.uniroma3.it

[2] Dipartimento di Ingegneria Elettronica e dell'Informazione, Perugia University, Italy
{walter.didimo,liotta}@diei.unipg.it

[3] School of Basic Sciences, École Polytechnique Fédérale de Lausanne, Switzerland

[4] School of Information Technologies, University of Sydney, Australia
shhong@it.usyd.edu.au

[5] Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, Germany
mk@informatik.uni-tuebingen.de

[6] Cheriton School of Computer Science, University of Waterloo, Canada
alubiw@uwaterloo.ca

Abstract. This paper describes algorithms for computing non-planar drawings of planar graphs in subquadratic area such that: (i) edge crossings are allowed only if they create big enough angles; (ii) the maximum number of bends per edge is bounded by a (small) constant.

Introduction

Recent cognitive experiments in graph visualization show that the human understanding of a graph layout is negatively affected by edge crossings that form acute angles, while edge crossings that form angles from about $\pi/3$ to $\pi/2$ guarantee good readability properties [**7, 8**]. As a consequence, in the last two years a large body of papers has been devoted to the study of non-planar drawings where such "sharp angle crossings" are forbidden.

A *Right Angle Crossing drawing (RAC drawing)* is a drawing where edges cross only at right angles. RAC drawings have been introduced in [**4**], where it is shown that: (i) every graph admits a RAC drawing with *curve complexity* 3, that is, with at most 3 bends per edge; (ii) straight-line RAC drawings have at most $4n - 10$ edges (and this bound is tight). Arikushi *et al.* [**2**] proved that even RAC drawings with curve complexity 1 or 2 have a linear number of edges. A *Large Angle Crossing drawing (LAC_α drawing)* is a relaxation of a RAC drawing, in which crossing edges form angles of at least α, where α is a given number in the interval $(0, \pi/2)$. LAC_α drawings have been independently introduced in [**3**] and [**6**].

In this paper we investigate RAC and LAC_α drawings of planar graphs in subquadratic area and constant curve complexity, where the area of a drawing is the area of the smallest rectangle containing it. In [**1**] it is proved that straight-line RAC drawings of planar graphs may require $\Omega(n^2)$ area, and that every graph with vertex degree at most 6 (at most 3) admits a RAC drawing with curve complexity 2 (curve complexity 1, resp.) in

[2]This work started during the Bertinoro Workshop on Graph Drawing 2010. The work was partially supported by the ESF project 10-EuroGIGA-OP-003 GraDR "Graph Drawings and Representations", by the MIUR of Italy, under projects AlgoDEEP, prot. 2008TFBWL4, and FIRB, "Advanced tracking system in intermodal freight transportation", grant no. RBIP06BZW8, by the Swiss National Science Foundation, grant no. 200021-125287/1, and by the Centre Interfacultaire Bernoulli (CIB) of EPFL.

$O(n^2)$ area. In [**4**] a simple algorithm is given for computing a RAC drawing with curve complexity 3 in $O(n^4)$ area for every graph. In [**3**] it is proved that every graph admits a RAC drawing with curve complexity 4 in $O(n^3)$ area. Concerning LAC_α drawings, in [**3**] it is also proved that every graph admits such a drawing with curve complexity 1 in $O(n^2)$ area, for any given $\alpha \in (0, \pi/2)$.

We present new advances in the study of the trade-off between curve complexity and area requirement of RAC and LAC_α drawings. Our main contributions are the following:

- We describe two different algorithms for constructing RAC drawings of d-degree planar graphs with curve complexity 4 in area $O(dn \log^2 n)$ and $O(n\sqrt{dn})$, respectively. Note that, if d is bounded by a sublinear function, both the area bounds are subquadratic.
- We describe an algorithm for constructing LAC_α drawings of every planar graph with curve complexity 2 in $O(n^{5/3})$ area, hence providing the first subquadratic area bound for planar graphs in a model that is reasonable from the user perspective.

1 Preliminaries

An *exact edge-separator* in a graph $G(V, E)$ is a set of edges $E' \subseteq E$ whose removal partitions G into two subgraphs $G_1(V_1, E_1)$, $G_2(V_2, E_2)$ with $|V_1| \leq \lfloor n/2 \rfloor$, $|V_2| \leq \lceil n/2 \rceil$.

Lemma 1.1 (Diks et al. [**5**]). *Every n-vertex d-degree planar graph has an exact edge separator of size $2\sqrt{2dn}$.*

The *cut-width* of G is the smallest integer k such that the vertices of G can be arranged in a linear layout $v_1, \ldots, v_n$ so that, for every i, at most k edges have one endpoint in $v_1, \ldots, v_i$ and the other in $v_{i+1}, \ldots, v_n$. The following is a consequence of Lemma 1.1.

Lemma 1.2 ([**5**]). *The cut-width of an n-vertex d-degree planar graph is $O(\sqrt{dn})$.*

2 RAC Drawings

In this section we provide two different techniques for constructing RAC drawings of bounded-degree planar graphs with curve complexity 4. The first technique is adapted from the classical method of Leiserson [**9**] for orthogonal drawings of 4-degree graphs, enhanced by a careful choice of bend points close to each vertex.

Theorem 2.1. *Every n-vertex d-degree planar graph admits a RAC drawing with curve complexity 4 in $O(dn \log^2 n)$ area.*

Proof sketch. The drawing is constructed recursively by computing an exact separator of the graph, by drawing the two subgraphs obtained when removing the separating edges, and by placing such drawings either horizontally or vertically next to each other.

For each vertex v, reserve two vertical lines $r(v)$ and $l(v)$, and two horizontal lines $t(v)$ and $b(v)$, to the right, to the left, above and below v, respectively. See Fig. 1(a). Suppose that the two drawings are placed horizontally, the other case being analogous. Reserve a vertical channel of size $2\sqrt{2dn}$ between the two drawings for the separating edges. Each edge (v, w), with v in the left subgraph and w in the right subgraph, is routed by placing two bends on $r(v)$ and $l(w)$, connected through a sequence of a horizontal, a vertical, and a horizontal segment. See Fig. 1(b).

The curve complexity of the drawing is 4 and all the segments that are not axis-parallel are drawn between $r(v)$, $l(v)$, $t(v)$, and $b(v)$ without crossings. Hence, the drawing is RAC. As the edge separator has size $2\sqrt{2dn}$, the area bound follows. □

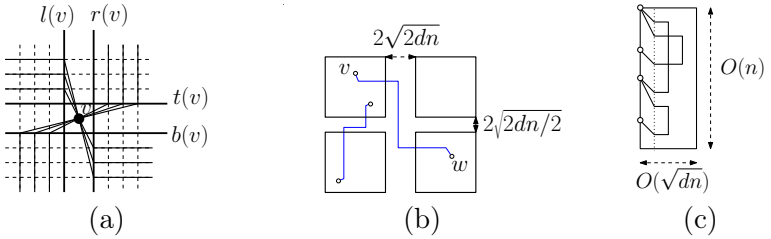


FIGURE 1. (a) The routing of the edges around a vertex v. (b) The layout of Theorem 2.1. (c) The layout of Theorem 2.2.

Note that, when d is $o(n/\log^2 n)$, the area bound obtained in Theorem 2.1 is subquadratic, while it is superquadratic when d is $O(n)$. In view of this, we present a second drawing technique achieving an area bound of $O(n\sqrt{dn})$, which is quadratic when d is $O(n)$ and subquadratic when d is $o(n)$, that works better for high-degree graphs. In particular, it outperforms the one in Theorem 2.1 whenever d is greater than $O(n/\log^4 n)$.

Theorem 2.2. *Every n-vertex d-degree planar graph G admits a RAC drawing with curve complexity 4 in $O(n) \times O(\sqrt{dn})$ area.*

Proof sketch. The drawing is constructed inside an axis-parallel rectangle $R(G)$, with the vertices lying on the left side of $R(G)$, the first and the last bend of each edge lying on the vertical line immediately to the right of the left side of $R(G)$, and the edges between such two bends being composed of one vertical and two horizontal segments. Refer to Fig. 1(c). As only axis-parallel segments might cross, the drawing is RAC. As each edge uses two horizontal lines, the height of $R(G)$ is $O(n)$, while the width is determined by the cut-width, which is $O(\sqrt{dn})$. □

3 LAC Drawings

In this section we provide an algorithm for constructing LAC_α drawings of planar graphs with curve complexity 2 in subquadratic area for any angle $\alpha < \pi/2$.

First, we provide an algorithm to construct LAC_α drawings of bounded-degree graphs with curve complexity 2 in subquadratic area. The construction, depicted in Fig. 2(a), is analogous to the one of Theorem 2.2. Namely, the vertices are placed on the left side of a rectangle according to the optimal cut-width order, so that any two consecutive of them are separated by a horizontal line. Each edge (u, v) is routed by placing one bend on the line below u and one on the line below v joined by a vertical segment, so that at least k vertical lines lie between the left side of $R(G)$ and the two bends, where $k = \frac{1}{\pi/2-\alpha}$ is a constant. The bounded cut-width implies the subquadratic area bound, while the distance of at least k implies large angle crossings.

Lemma 3.1. *Every n-vertex d-degree planar graph G admits a LAC_α drawing with curve complexity 2 in $O(n) \times O(\sqrt{dn})$ area, for any $0 \leq \alpha < \pi/2$.*

We now prove the subquadratic bound for LAC_α drawings of planar graphs.

Theorem 3.2. *Every n-vertex planar graph G admits a LAC_α drawing with curve complexity 2 in $O(n^{1.667})$ area for any $0 \leq \alpha < \pi/2$.*

Proof sketch. We say that a vertex is *heavy* if its degree is greater than or equal to $n^{1/3}$ and it is *light* otherwise. Let H and L denote the sets of heavy and light vertices, respectively.

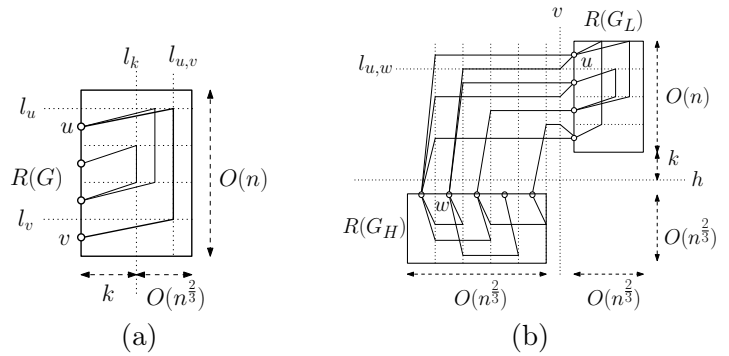


FIGURE 2. (a) Layout of Lemma 3.1. (b) Layout of Theorem 3.2.

Note that $|H| = O(n^{2/3})$ and $|L| = O(n)$. Let G_H and G_L be the subgraphs induced by H and L, respectively. By Lemma 3.1, they admit LAC_α drawings with curve complexity 2 inside rectangles $R(G_H)$ and $R(G_L)$, where $R(G_H)$ has height $O(|H|) = O(n^{2/3})$ and width $O(\sqrt{n^{2/3}n^{2/3}}) = O(n^{2/3})$, as the maximum degree of G_H is less than $n^{2/3}$ due to $|H| = O(n^{2/3})$, while $R(G_L)$ has height $O(|L|) = O(n)$ and width $O(\sqrt{n^{1/3}n}) = O(n^{2/3})$.

Refer to Fig. 2(b). Consider a horizontal line h and a vertical line v. Rotate $R(G_H)$ in such a way that the vertices are on the upper side and place it with its upper side one unit below h and its right side one unit to the left of v. Place $R(G_L)$ with its lower side k units above h, where $k = \frac{1}{\pi/2-\alpha}$, and its left side one unit to the right of v. Edges connecting a vertex u in L to a vertex w in H are routed inside the smallest rectangle $R(G)$ containing both $R(G_L)$ and $R(G_H)$ by placing first reaching from u the topmost free point of line v, then by moving horizontally till intersecting the vertical line to the right of w, and by finally reaching w.

As crossing may only happen between a horizontal segment and a segment connecting two points with horizontal distance 1 and vertical distance at least k, the constructed drawing is a LAC_α drawing with curve complexity 2. The height of $R(G)$ is $O(n)$, as it is equal to the height of $R(G_L)$, which is $O(n)$, plus the height of $R(G_H)$, which is $O(n^{2/3})$, plus a constant number of $k+1$ lines, while the width of $R(G)$ is $O(n^{2/3})$, as it is equal to the width of $R(G_L)$, which is $O(n^{2/3})$, plus the width of $R(G_H)$, which is $O(n^{2/3})$, and the statement follows. □

References

[1] P. Angelini, L. Cittadini, G. Di Battista, W. Didimo, F. Frati, M. Kaufmann, A. Symvonis. On the perspectives opened by Right Angle Crossing drawings, *JGAA, Special Issue on Sel. Pap. from GD'09* **15(1)** (2011), 53–78.

[2] K. Arikushi, R. Fulek, B. Keszegh, F. Moric, C. D. Tóth. Graphs that admit Right Angle Crossing drawings, in: *WG'10*, LNCS, 6410, 2010, 135–146.

[3] E. Di Giacomo, W. Didimo, G. Liotta, H. Meijer. Area, curve complexity, and crossing resolution of non-planar graph drawings, in: *GD'09*, LNCS, 5849, 2010, 15–20.

[4] W. Didimo, P. Eades, G. Liotta. Drawing graphs with Right Angle Crossings, in: *WADS'09*, LNCS, 5664, 2009, 206–217.

[5] K. Diks, H. Djidjev, O. Sýkora, I. Vrto. Edge separators of planar and outerplanar graphs with applications, *J. Algorithms* **14** (1993), 258–279.

[6] V. Dujmović, J. Gudmundsson, P. Morin, T. Wolle. Notes on large angle crossing graphs, in: *CATS'10*, 2010, 19–24.

[7] W. Huang. Using eye tracking to investigate graph layout effects, in: *APVIS'07*, 2007, 97–100.

[8] W. Huang, S.-H. Hong, P. Eades. Effects of crossing angles, in: *PacificVis'08*, 2008, 41–46.

[9] C. E. Leiserson. Area-efficient graph layouts (for VLSI), in: *FOCS'80*, IEEE, 1980, 270–281.

Euclidean arrangements of n pseudolines with one n-gon are stretchable

J. Leaños[1], C. Ndjatchi-Mbe-Koua[2], L. M. Rivera-Martínez[3]

[1] Unidad Académica de Matemáticas, Universidad Autónoma de Zacatecas, México
jleanos@mate.reduaz.mx

[2] Coordinación de Ingeniería Mecatrónica y Energía, Universidad Politécnica de Zacatecas, México
ndjatchi@upz.edu.mx

[3] Faculty of Mathematics, University of Vienna, Austria
luismanuel.rivera@univie.ac.at

Abstract. In [**4**] it was proved that, if a simple Euclidean arrangement of pseudolines has no (≥ 5)-gons, then it is stretchable. In the opposite direction, it is known that not every simple Euclidean arrangement with three (≥ 5)-gons is stretchable; see [**8**]. Thus the following question arises naturally: Are the simple Euclidean arrangements with one or two (≥ 5)-gons stretchable? In this paper we give a large class of stretchable simple Euclidean arrangements with one (≥ 5)-gon. More precisely, we prove that, if $\mathcal{L}$ is a Euclidean arrangement of n pseudolines with one n-gon, then $\mathcal{L}$ is stretchable. We also prove that the number of such arrangements is $\Omega(2^{n/2})$.

Introduction

A simple noncontractible closed curve in the projective plane $\mathbb{P}$ is a *pseudoline*, and an *arrangement of pseudolines* is a collection $\mathcal{B} = \{p_0, p_1, \ldots, p_n\}$ of pseudolines that intersect (necessarily cross) pairwise exactly once. Since $\mathbb{P} \setminus p_0$ is homeomorphic to the Euclidean plane $\mathbb{E}$, we may regard $\{p_1, \ldots, p_n\}$ as an *arrangement of pseudolines* in $\mathbb{E}$ (and regard $p_1, \ldots, p_n$ as *pseudolines* in $\mathbb{E}$). An arrangement is *simple* if no point belongs to more than two pseudolines. The cell complex of an Euclidean arrangement in $\mathbb{P}$ has both bounded and unbounded cells. As in [**4**], we are only interested in bounded cells (whose interiors are the *polygons* or *faces*). Thus it is clear what is meant by a *triangle*, a *quadrilateral*, or, in general, an *n-gon* of the arrangement. Any m-gon with $m \geq n$ is called a ($\geq n$)-gon.

An *arrangement of lines* in $\mathbb{E}$ is a collection of straight lines, no two of them parallel. Thus, every arrangement of lines is an arrangement of pseudolines. On the other hand, not every arrangement of pseudolines is *stretchable*, that is, equivalent to an arrangement of lines, where two arrangements are *equivalent* if they generate isomorphic cell decompositions of $\mathbb{E}$. Every arrangement of eight pseudolines is stretchable [**3**], but there is a simple non-stretchable arrangement (Figure 1) of nine pseudolines [**8**], which is unique up to isomorphism [**3**]. Since the arrangement in Figure 1 is non-stretchable and has only three (≥ 5)-gons, it follows that not every simple Euclidean arrangement with three (≥ 5)-gons is stretchable. In the opposite direction, in [**4**] was proved that every simple Euclidean arrangement with no (≥ 5)-gons is strechable. With these facts in mind, it is natural to ask what is the role of the (≥ 5)-gons in the stretchability of simple Euclidean

[3]Partially supported by an ERC Grant.

arrangements. This work is a first effort in this direction. In particular, here we study the stretchability of certain class of Euclidean arrangements with one (≥ 5)-gon.

Stretchability questions are typically difficult: deciding stretchability is NP-hard [**9**] even for simple arrangements [**10**]. The concept of stretchability is particularly relevant because of the close connection between arrangements of pseudolines and rank 3 oriented matroids: on this ground, the problem of stretchability of arrangements is equivalent to the problem of realizability for oriented matroids (see [**1, 7**]).

Let $\Im$ denote the set of Euclidean arrangements of n pseudolines with one n-gon. It follows from the definition of $\Im$ that every arrangement of $\Im$ must be simple. Our aim is to prove that every element of $\Im$ is stretchable and that the number of non-isomorphic arrangements of $\Im$ with m pseudolines grows exponentially with m.

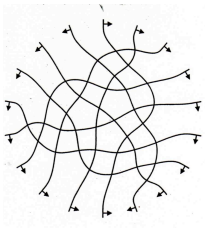


FIGURE 1. A simple non-stretchable arrangement with three (≥ 5)-gons.

1 Results

Our main results are the following:

Theorem 1.1. *If $\mathcal{L}$ is an arrangement of $\Im$, then $\mathcal{L}$ is stretchable.*

Theorem 1.2. *The number of non-isomorphic Euclidean arrangements of n pseudolines with one n-gon is $\Omega(2^{n/2})$.*

To prove Theorem 1.1, we need two lemmas. Lemma 1.4 is a consequence of the proof of Proposition 1.2 in [**2**] and the definition of $\Im$. Lemma 1.5 establishes structural properties of the elements of $\Im$. Lemmas 1.3 and 1.5 are both easy to see.

Since Theorem 1.1 is trivial for $\mathcal{L} \in \Im$ with $|\mathcal{L}| \leq 4$, we may assume that $|\mathcal{L}| \geq 5$.

Lemma 1.3. *If $\mathcal{L} \in \Im$, then the $|\mathcal{L}|$-gon of $\mathcal{L}$ is the unique (≥ 5)-gon of $\mathcal{L}$.*

Lemma 1.4. *If $\mathcal{L} \in \Im$, then the $|\mathcal{L}|$-gon of $\mathcal{L}$ has at most two edges that are adjacent to an unbounded cell of $\mathcal{L}$.*

Lemma 1.5. *Let $\mathcal{L} \in \Im$. If an edge e of the $|\mathcal{L}|$-gon of $\mathcal{L}$ is not adjacent to an unbounded cell of $\mathcal{L}$, then e is also an edge of a triangle of $\mathcal{L}$.*

We now give a sketch of the proof of Theorem 1.1. See [**5**] for further details.

1.1 Proof of Theorem 1.1

The proof is by induction on $|\mathcal{L}|$. In [**3**] it was shown that any Euclidean arrangement with 8 pseudolines is stretchable. So we may assume that $|\mathcal{L}| = n \geq 9$ and that every arrangement of $\Im$ with $j < n$ pseudolines is stretchable. Let P be the n-gon of $\mathcal{L}$. By Lemma 1.4, P has at most two edges that are adjacent to an unbounded cell of $\mathcal{L}$. Hence,

P has three consecutive edges, say a', b' and c', which are not adjacent to an unbounded cell of $\mathcal{L}$. By Lemma 1.5, each of a', b' and c' is an edge of a triangle of $\mathcal{L}$. Let A, B and C be, respectively, the triangles of $\mathcal{L}$ that are adjacent to a', b' and c'. Let a, b and c be the supporting pseudolines of a', b' and c', respectively. Assume without loss of generality that a, b and c are directed in such a way that P lies to the left of each of them. See Figure 2.

For $x \in \{a, b, c\}$ and $\ell = 1, \ldots, n-1$, let x_ℓ be the ℓ-th pseudoline of $\mathcal{L} \setminus \{x\}$ that is crossed by x. Since $\mathcal{L}$ is simple, the labels $a_1, \ldots, a_{n-1}$; $b_1, \ldots, b_{n-1}$; and $c_1, \ldots, c_{n-1}$ are well-defined. We assume from now on that $a_k = c$, $b_t = a$ and $c_r = a$.

Let X, Y, W and Z be the four unbounded regions defined by a and c (see the small drawing in Figure 2). Note that $P \subset X$. Since P has an edge in every pseudoline of $\mathcal{L}$, then Z (respectively, Y) contains no crossings between pseudolines of $\{a_1, \ldots, a_{k-2}\}$, (respectively, $\{c_{r+2}, \ldots, c_{n-1}\}$). Using these facts, we can obtain Equations (1) and (2) by a straightforward argument.

$$b_{t-j} = \begin{cases} a_{k-1-j} = c_{r-j} & \text{if } j = 1, 2, \ldots, r-1, \\ a_{k-1-j} & \text{if } j = r, r+1, \ldots, t-1. \end{cases} \tag{1}$$

$$b_{t+1+i} = \begin{cases} c_{r+1+i} = a_{k+i} & \text{if } i = 1, 2, \ldots, n-k-1, \\ c_{r+1+i} & \text{if } i = n-k, n-k+1, \ldots, n-t-2. \end{cases} \tag{2}$$

From Equations (1) and (2) it follows that $\mathcal{L}$ looks like in Figure 2.

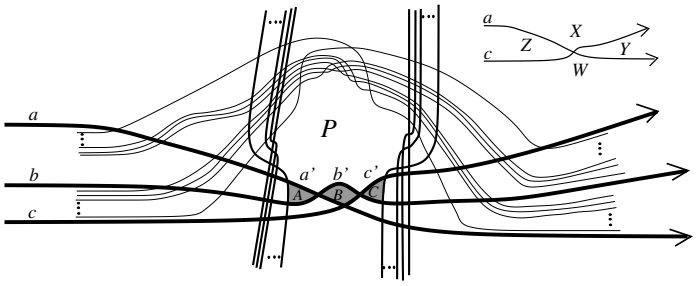


FIGURE 2. The structure of an arrangement of $\Im$.

Clearly, $\mathcal{L} \setminus \{b\}$ belongs to $\Im$ and, by induction, it is stretchable. Let $\mathcal{L}_b^*$ be an arrangement of straight lines, which is equivalent to $\mathcal{L} \setminus \{b\}$. If θ denotes an element of $\mathcal{L} \setminus \{b\}$, we denote by θ^* the corresponding element in $\mathcal{L}_b^*$. For $s = 1, \ldots, k-r-1$, let m_s be the slope of a_s^*. Let $H = \{a_1, \ldots, a_{k-r-1}\}$ (the thin pseudolines in Figure 2 are the elements of H). Since any two lines of $\mathcal{L}_b^*$ intersect exactly once, $\mathcal{L}_b^*$ has no lines with equal slopes. Moreover, since the crossing of any two lines of H^* is in X^*, $m_0 < m_1 < \cdots < m_{k-r-1} < m_{k-r}$, where m_0 and m_{k-r} are, respectively, the slopes of a^* and c^*. Let d^* be the line with slope $(m_{k-t-1} + m_{k-t})/2$ through $v = a^* \cap c^*$. Since $\mathcal{L}_b^*$ and $\mathcal{L} \setminus \{b\}$ are equivalent and $m_0 < m_1 < \cdots < m_{k-r}$, d^* crosses the lines of $\mathcal{L}_b^* \setminus \{a^*, c^*\}$ in the exact same order in which b crosses the pseudolines of $\mathcal{L} \setminus \{a, b, c\}$. Also, note that d^* can be perturbed in such a way that: (i) the order in which d^* crosses the lines of $\mathcal{L}_b^* \setminus \{a^*, c^*\}$ is preserved, and (ii) d^* intersects X^*. Finally, note that the arrangement of lines obtained by such a perturbation of d^* is equivalent to $\mathcal{L}$, as desired.

1.2 Proof of Theorem 1.2

A 2-colored necklace with $2m$ beads in which opposite beads have different colors is a *self-dual* necklace. An example of a self-dual necklace is shown in Figure 3 *i*). Two self-dual necklaces are isomorphic if one can be obtained from the other by rotation or reflection or both. In [**6**] it was proved that the number of non-isomorphic self-dual necklaces is $\Omega(2^{m/2})$. So, it is enough to exhibit a one-to-one correspondence between the set of self-dual necklaces and a subset of $\Im$. Let C be a self-dual necklace with $2m \geq 6$ beads colored 0 and 1, and let P be the regular polygon of $2m$ sides. Now we extend every edge of P to both sides in such a way that each pair of non-parallel segments intersect. See Figure 3 *ii*). Finally, we intersect every pair of parallel segments according to color 1 of C as shown in Figure 3 *iii*). By construction, the resultant arrangement P_C belongs to $\Im$. It is easy to see that distinct necklaces generate distinct arrangements.

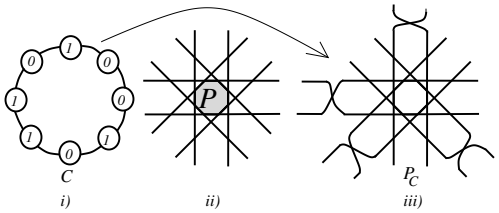


FIGURE 3. How to associate an arrangement of $\Im$ to each self-dual necklace with $2m$ beads.

Remark 1.6. An anonymous referee has observed that the elements of $\Im$ are nothing but the extensions of the cyclic arrangement by one new line that does not cross the n-gon and consequently the number of non-isomorphic Euclidean arrangements of n pseudolines with one n-gon is $\Omega(2^{n-1}/n)$.

References

[1] A. Björner, M. Las Vergnas, B. Sturmfels, N. White, and G. Ziegler, *Oriented matroids*, Encyclopedia of Math. and its Applications, **46**. Cambridge University Press, Cambridge, 1993.

[2] S. Felsner and K. Kriegel, *Triangles in Euclidean arrangements*, J. Disc. and Comp. Geom. **22** (1999), no. 3, 429–438.

[3] J. E. Goodman and R. Pollack, *Proof of Grünbaum's conjecture on the stretchability of certain arrangements of pseudolines*, Journal of Combinatorial Series A **29** (1980), no. 3, 385–390.

[4] J. Leaños, M. Lomeli, C. Merino, G. Salazar, and J. Urrutia, *Simple Euclidean arrangements with no* (≥ 5)*-gons*, J. Disc. and Comp. Geom. **38** (2007), no. 3, 595–603.

[5] J. Leaños, C. Ndjatchi Mbe Koua, and L. M. Rivera-Martínez, *Simple Euclidean arrangements with one* (≥ 5)*-gon*, arXiv:1008.4598v1 [math.co].

[6] E. M. Palmer and R. W. Robinson, *Enumeration of self-dual configurations*, Pacific J. Math. **110** (1984), 203–221.

[7] J. Richter-Gebert and G. M. Ziegler, *Oriented matroids*, Handbook of Disc. and Comp. Geom., 2nd ed. (J. E. Goodman and J. O'Rourke, eds.), pp. 129–151. Chapman & Hall/CRC, FL, 2004.

[8] G. Ringel, *Teilungen der Ebene durch Geraden oder topologische Geraden*, Math. Z. **64** (1956), 79–102.

[9] P. W. Shor, *Stretchability of pseudolines is NP-hard*, Applied Geometry and Discrete Mathematics, DIMACS Ser. Discrete Math. Theoret. Comput. Sci. **4**, 531–554. AMS Providence, RI, 1991.

[10] P. Bose, H. Everett, and S. Wishmath, *Properties of arrangements*, Int. Journal of Comp. Geom. **13(6)** (2003), 447–462.

Quadrangulations on 3-colored point sets with Steiner points

Sho Kato[1], **Ryuichi Mori**[1], **Atsuhiro Nakamoto**[1]

[1] Department of Mathematics, Yokohama National University, Yokohama 240-8501, Japan
uz2s-ktu@asahi-net.or.jp, ryu-chan@uriwari.com, nakamoto@ynu.ac.jp

Abstract. Let P be a point set on the plane, and consider whether P is *quadrangulatable*, that is, whether there exists a 2-connected bipartite plane graph G with each edge a straight segment such that $V(G) = P$, and the boundary of the unbounded face of G coincides with $\mathrm{Conv}(P)$, and that each bounded face of G is quadrilateral. It is easy to see that it can be done if an even number of vertices of P appear on $\mathrm{Conv}(P)$. Hence we give a k-coloring of P and consider the same problem so that no edge joins two vertices of P with the same color. In this case, we always assume that the number of the points of P lying on $\mathrm{Conv}(P)$ is even and that any two consecutive points on $\mathrm{Conv}(P)$ have distinct colors. However, there is a 2-colored non-quadrangulatable point set P. So we introduce *Steiner points*, each of which can be put in any position of the interior of $\mathrm{Conv}(P)$ and may be colored by any of the k colors. When $k = 2$, Alvarez et al. proved that if a point set P on the plane consists of $\frac{n}{2}$ red and $\frac{n}{2}$ blue points in general position, then adding Steiner points Q with $|Q| \leq \lfloor \frac{n-2}{6} \rfloor + \lfloor \frac{n}{4} \rfloor + 1$, $P \cup Q$ is quadrangulatable, but there exists a non-quadrangulatable 3-colored point set no matter how many Steiner points are added. In this paper, we define the winding number for a 3-colored point set P, and prove that a 3-colored point set P in general position with a finite number of Steiner points Q added is quadrangulatable if and only if the winding number of P is zero. When $P \cup Q$ is quadrangulatable, we prove that $|Q| \leq \frac{7n+34m-48}{18}$, where $|P| = n$ and the number of points of P in $\mathrm{Conv}(P)$ is $2m$.

1 Introduction

Let P be a point set arranged on the plane in *general position*, that is, no three points of P lie on a straight line. Let $\mathrm{Conv}(P)$ denote the convex hull of P. A *quadrangulation* on P is a 2-connected bipartite plane graph G with each edge a straight segment such that $V(G) = P$, and that the boundary of the unbounded face of G coincides with $\mathrm{Conv}(P)$, and that each bounded face of G is quadrilateral. We say that P is *quadrangulatable* if there exists a quadrangulation on P. Figure 1 shows a point set P and a quadrangulation on P.

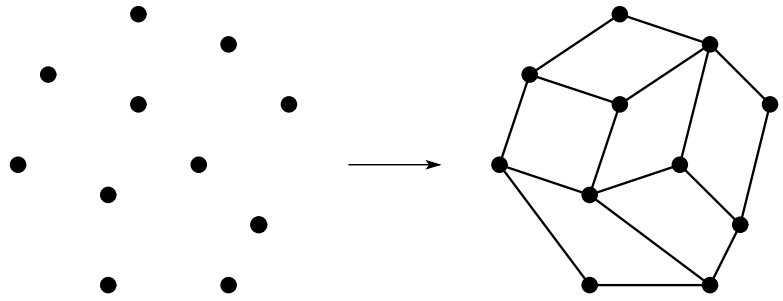


FIGURE 1. A quadrangulatable point set P.

In this paper, we discuss whether a given point set P is quadrangulatable or not. It is easy to see that not all point sets P are quadrangulatable, and that a necessary

condition for P to be quadrangulatable is that $|P| \geq 4$ and $\mathrm{Conv}(P)$ contains an even number of points. It was proved in **[3, 9]** that these conditions are also sufficient for P to be quadrangulatable. (Some papers focus on quadrangulations on point sets with all bounded faces convex **[4, 6]**.) For those problems, we refer the reader to the survey **[10]**.

A *k-colored* point set P is one in which a k-coloring $c\colon P \to \{1, 2, \ldots, k\}$ is fixed. Let us study whether P is quadrangulatable so that no edge joins two points of P with the same color. For this problem, we have to assume that $\mathrm{Conv}(P)$ contains an even number of points and that any two consecutive points on $\mathrm{Conv}(P)$ have distinct colors. It is easy to see that, for any $k \geq 2$, there exists a k-colored point set which is not quadrangulatable **[5]**. See the point set on the left side of Figure 2, in which each of the black inner vertices b_3 and b_4 must be joined to the white vertices w_1 and w_2, but they cannot be done simultaneously. Therefore, we introduce *Steiner points* for P, which are auxiliary points put in the interior of $\mathrm{Conv}(P)$ and each of which may be colored by any of $\{1, 2, \ldots, k\}$. See the right side of Figure 2.

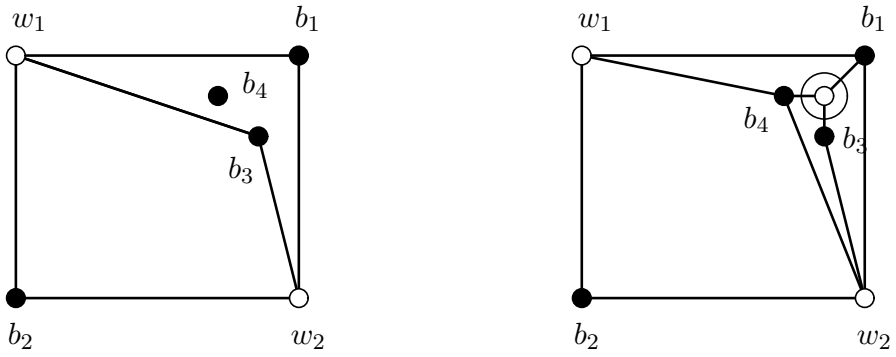


FIGURE 2. A non-quadrangulatable 2-colored point set P and a quadrangulation on $P \cup Q$ with Steiner points Q added.

For 2-colored point sets, Alvarez et al. proved the following theorem.

Theorem 1 (Alvarez, Sakai and Urrutia **[1]**)**.** *Let P be a 2-colored point set on the plane in general position with $|P| = n \geq 4$ in which $\frac{n}{2}$ points are red and $\frac{n}{2}$ points are blue. Then, adding a set Q of at most $\lfloor \frac{n-2}{6} \rfloor + \lfloor \frac{n}{4} \rfloor + 1$ Steiner points in the interior of* $\mathrm{Conv}(P)$*, $P \cup Q$ is quadrangulatable.*

In the same paper, Alvarez et al. constructed a 2-colored point set P with n points requiring at least $\frac{n}{3}$ Steiner points for its quadrangulation.

On the other hand, they claimed that such a theorem does not hold for 3-colored point set as in the following theorem.

Theorem 2 (Alvarez, Sakai and Urrutia **[1]**)**.** *Let P be a 3-colored point set with* $\mathrm{Conv}(P)$ *hexagon such that the six points on* $\mathrm{Conv}(P)$ *are colored by 1, 2, 3, 1, 2, 3 in this cyclic order (shown at left in Figure 3). Then, even if we add any set Q of Steiner points in the interior of* $\mathrm{Conv}(P)$*, $P \cup Q$ is not quadrangulatable.*

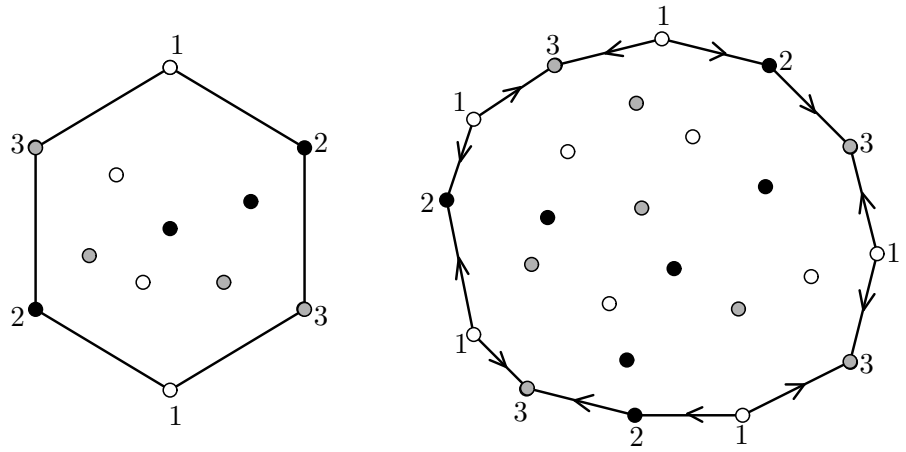


FIGURE 3. Non-quadrangulatable 3-colored point sets P, no matter how many Steiner points are added.

2 Winding number of 3-colored point sets

In this section, we define the notion of "winding number" of a k-colored point set P on the plane, which will describe a necessary condition for a 3-colored point set P to be quadrangulatable. The winding number is well known as an important tool for investigating 3-colorability of quadrangulations on surfaces from a topological point of view [**2, 8**].

Let P be a point set and a 3-coloring c of P be given by $c\colon P \to \{1, 2, 3\}$. Suppose that two consecutive vertices x and y on $\mathrm{Conv}(P)$ are such that $c(x) < c(y)$. Then we give a direction to xy from x to y. In this way, we can give a direction to all edges on $\mathrm{Conv}(P)$. The *winding number* of P, denoted $w(P)$, is defined as the subtraction of the number of directed edges along clockwise orientation of $\mathrm{Conv}(P)$ and that along counterclockwise orientation. It is easy to see that the example on the left side of Figure 3 has four directed edges along clockwise orientation and two along counterclockwise orientation, and hence this example has winding number 2.

Proposition 3. *If a 3-colored point set P with a set of Steiner points added is quadrangulatable, then $w(R) = 0$.*

Applying Proposition 3, we can see that the 3-colored point set shown at right in Figure 3 is not quadrangulatable no matter how many number of Steiner points are added. By Proposition 3, we can construct an arbitrarily large example, although Alvarez et al. gave only one example.

3 Main theorem

In this section, we describe our main theorem for the quadrangulatability of 3-colored point set possibly with Steiner points. Our main theorem is as follows.

Theorem 4. *Let P be a 3-colored point set in general position. Adding a finite set Q of Steiner points to the interior of* $\mathrm{Conv}(P)$*, we can quadrangulate $P \cup Q$ if and only if $w(P) = 0$. If $w(P) = 0$, then there is a set Q of Steiner points with $|Q| \leq \frac{7n+34m-48}{18}$ for which $P \cup Q$ is quadrangulatable, where $|P| = n$ and the number of points of P in* $\mathrm{Conv}(P)$ *is $2m$.*

Similarly to the construction of a 2-colored point set requiring many Steiner points in [**1**], we can construct a 3-colored point set P requiring at least $\frac{n}{3}$ Steiner points for its quadrangulation. Hence, if n is sufficiently larger than m in a 3-colored point set P, then the estimation for $|Q|$ in Theorem 4 is reasonably good, since we have $|Q| \approx \frac{7n}{18}$ in this case. However, if n is close to m, it is very bad.

Our proof of Theorem 4 proceeds by showing the following two facts:

(1) Let P be a point set with the points lying on $\mathrm{Conv}(P) = v_1 \cdots v_{2m}$ in this order. Suppose that one of the three colors appears only on some of $v_1, v_3, \ldots, v_{2m-1}$. Then P is quadrangulatable by adding a finite number of Steiner points.

(2) Let P be a point set with winding number 0. Then the convex set bounded by $\mathrm{Conv}(P)$ can be partitioned into several convex sets by cutting along some diagonals joining two points on $\mathrm{Conv}(P)$ so that each convex set partitioned satisfies the assumption in (1).

Note that the assumption for P in (1) guarantees $w(P) = 0$, and that (1) can be proved by a similar argument to the 2-colored case in [**1**]. So, the argument in (2) proves that, if P satisfies $w(P) = 0$, then P is partitioned into several point sets each of which has winding number 0 and is quadrangulatable with a finite number of Steiner points added. Thus, since the diagonals of $\mathrm{Conv}(P)$ are doubled in our argument, our estimation for the number of Steiner points added for P depends on the number of those convex sets partitioned. Therefore, our estimation contains a term for the number m of edges on $\mathrm{Conv}(P)$. If we have a new method for proving Theorem 4, we might have an estimation for the number of Steiner points which does not depend on m.

References

[1] V. Alvarez, T. Sakai, and J. Urrutia, Bichromatic quadrangulations with steiner points. *Graph Combin.* **23** (2007), 85–98.

[2] D. Archdeacon, J. P. Hutchinson, A. Nakamoto, S. Negami, and K. Ota, Chromatic numbers of quadrangulations on closed surfaces, *J. Graph Theory* **37** (2001), 100–114.

[3] P. Bose and G. Toussaint, Characterizing and efficiently computing quadrangulations of planar point sets, *Comput. Aided Geom. Design* **14** (1997), 763–785.

[4] D. Bremner, F. Hurtado, S. Ramaswami, and V. Sacristán, Small convex quadrangulations of point sets, *Algorithmica* **38** (2003), 317–339.

[5] C. Cortés, A. Márquez, A. Nakamoto, and J. Valenzuela, Quadrangulations and 2-colorations, 21st European Workshop on Computational Geometry, Eindhoven, March 9–11 (2005), 65–68.

[6] M. Heredia and J. Urrutia, On convex quadrangulations of point sets on the plane, to appear in LNCS, Proc. China-Japan Joint Conference on Discrete Geometry, Combinatorics and Graph Theory, 2005.

[7] M. L. Lai and L. L. Schumaker, Scattered data interpolation using piecewise polynomials of degree six, *SIAM J. Numer. Anal.* **34** (1997), 905–921.

[8] B. Mohar and P. D. Seymour, Coloring locally bipartite graphs on surfaces, *J. Combin. Theory Ser. B* **84** (2002), 301–310.

[9] S. Ramaswami, P. Ramos, and G. Toussaint, Converting triangulations to quadrangulations, *Comput. Geom.* **9** (1998), 257–276.

[10] G. Toussaint, Quadrangulations of planar sets. In Proceedings of the 4th International Workshop on Algorithms and Data Structures, 218–227. Springer-Verlag, 1995.

Simultaneously flippable edges in triangulations

Diane L. Souvaine[1,3], **Csaba D. Tóth**[2,4], **Andrew Winslow**[1,3]

[1] Tufts University, Medford, MA, USA
dls@cs.tufts.edu, awinslow@cs.tufts.edu

[2] University of Calgary, Calgary, AB, Canada
cdtoth@ucalgary.ca

Abstract. We show that every straight-line triangulation on n vertices contains at least $(n-4)/5$ simultaneously flippable edges. This bound is the best possible, and resolves an open problem by Galtier *et al.*

Introduction

A *(geometric) triangulation* of a point set P is a planar straight line graph with vertex set P such that every bounded face is a triangle and the outer face is the exterior of the convex hull of P. An edge e of a triangulation is *flippable* if it is adjacent to two triangles whose union is a *convex* quadrilateral $Q(e)$. A set E of edges in a triangulation are *simultaneously flippable* if each edge in E is flippable and the quadrilaterals $Q(e)$, $e \in E$, are pairwise interior disjoint.

For a triangulation T_P of a point set P, let $f_{\text{sim}}(T)$ denote the maximum number of simultaneously flippable edges in T_P, and let $f_{\text{sim}}(n) = \min_{(T_P\,:\,|P|=n)} f_{\text{sim}}(T_P)$ be the minimum of $f_{\text{sim}}(T)$ over all n-element point sets in general position in the plane. The value of $f_{\text{sim}}(n)$ played a key role in recent results on the number of various classes of planar straight line graph embedded on given point sets [**2, 4**]. Hurtado *et al.* [**5**] proved that every triangulation on n vertices admits at least $(n-4)/2$ flippable edges, and this bound cannot be improved in general. Galtier *et al.* [**3**] proved that $f_{\text{sim}}(n) \geq (n-4)/6$, and that there are triangulations T_P with $|P| = n$ such that $f_{\text{sim}}(T_P) \leq (n-4)/5$, which is the best possible. In this note we improve the lower bound to $f_{\text{sim}}(n) \geq (n-4)/5$. This resolves an open problem posed in [**3**] and restated in [**1**].

1 Lower bound

Fix a set P of n points in general position in the plane, h of which lie on the boundary of the convex hull, and fix a triangulation $T = T_P$. Then T has exactly $3n - h - 3$ edges and $2n - h - 2$ bounded faces.

Separable edges. Following the terminology in [**4**], we say that an edge $e = uv$ of the triangulation is *separable* at vertex u if and only if there is a line ℓ_u through u such that uv is the only edge incident to u on one side of ℓ_u. We use the following observations from [**5**]: An edge uv of T is flippable if and only if it is separable at *neither* endpoint. If u is a hull vertex, then only the two incident hull edges are separable at u. Suppose now that u is a vertex in the interior of the convex hull. If u has degree 3, then all three incident edges are separable at u. If u has degree 4 or higher, then at most two edges are separable at u and these edges must be consecutive in the rotation of u.

[3]Research supported in part by NSF grants CCF-0830734 and CBET-0941538.
[4]Research supported in part by NSERC grant RGPIN 35586.

Similarly to [**5**], we assign every non-flippable edge e to an incident vertex at which it is separable. If e lies on the boundary of the convex hull, assign e to its counterclockwise first hull vertex. If e is incident to an interior vertex of degree 3, then assign e to this vertex. Otherwise assign e to one of its endpoints at which it is separable (breaking ties arbitrarily).

Based on the above observations, we can now distinguish five types of vertices. Let h be the number of hull vertices (with $h \geq 3$) and let n_3 be the number of interior vertices of degree 3. Denote by $n_{4,0}$, $n_{4,1}$, and $n_{4,2}$ the number of interior vertices of degree 4 or higher to which 0, 1, and 2 non-flippable edges, respectively are assigned. We have

$$n = h + n_{4,2} + n_{4,1} + n_{4,0} + n_3. \tag{1}$$

Using this notation, the number of non-flippable edges is exactly $h + 3n_3 + 2n_{4,2} + n_{4,1}$. Denoting by f the total number of flippable edges in T, we use the expression (1) to infer that $f = (3n - h - 3) - (h + 3n_3 + 2n_{4,2} + n_{4,1})$, or

$$f = h + n_{4,2} + 2n_{4,1} + 3n_{4,0} - 3. \tag{2}$$

Coloring argument. Galtier *et al.* [**3**] proved that every set of flippable edges in a triangulation is 3-colorable in such a way that each color class is simultaneously flippable. This implies, in particular, that every set of k flippable edges in a triangulation contains a subset of at least $k/3$ simultaneously flippable edges. This result, combined with a lower bound of $(n-4)/2$ on the total number of flippable edges, immediately gives $f_{\text{sim}} \geq (n-4)/6$. We improve this lower bound to $f_{\text{sim}}(n) \geq (n-4)/5$.

If $f \geq 3\lceil (n-4)/5 \rceil - 2$, then the above 3-coloring argument implies that the largest color class contains at least $\lceil (n-4)/5 \rceil$ simultaneously flippable edges, as required. In the remainder of the proof, we assume that

$$f \leq 3 \left\lceil \frac{n-4}{5} \right\rceil - 3 \leq \frac{3n}{5} - 3. \tag{3}$$

Recall that we have $f \geq \frac{1}{2}(n-4)$ by the result of Hurtado *et al.* [**5**]. So the number of flippable edges must be in the range $0.5n - 2 \leq f < 0.6n - 3$. Combining (2) and (3), we have

$$\frac{3}{5}n \geq h + n_{4,2} + 2n_{4,1} + 3n_{4,0}. \tag{4}$$

We apply the 3-coloring result by Galtier *et al.* [**3**] only for a subset of the flippable edges. We call a flippable edge e *isolated* if the convex quadrilateral $Q(e)$ is bounded by 4 non-flippable edges. It is clear that an isolated flippable edge is simultaneously flippable with any other flippable edge. Let f_0 and f_1 denote the number of isolated and non-isolated flippable edges, respectively, with $f = f_0 + f_1$. Applying the 3-coloring argument for the non-isolated flippable edges only, the number of simultaneously flippable edges is bounded by

$$f_{\text{sim}} \geq f_0 + \frac{f_1}{3} = (f - f_1) + \frac{f_1}{3} = f - \frac{2}{3}f_1. \tag{5}$$

An auxiliary triangulation. Similarly to Hurtado *et al.* [**5**] and Hoffmann *et al.* [**4**], we use an auxiliary triangulation $\widehat{T}$. We construct $\widehat{T}$ from T as follows:

(1) Add an auxiliary vertex w in the exterior of the convex hull, and connect it to all hull vertices.
(2) Remove all interior vertices of degree 3 (and all incident edges).

Notice that only nonflippable edges have been deleted from T. In the triangulation $\widehat{T}$, the number of vertices is $n - n_3 + 1 = h + n_{4,0} + n_{4,1} + n_{4,2} + 1$ and all faces (including the unbounded face) are triangles. By Euler's formula, the number of faces in $\widehat{T}$ is

$$m = 2(n - n_3 + 1) - 4 = 2h + 2n_{4,2} + 2n_{4,1} + 2n_{4,0} - 2. \tag{6}$$

We 2-color the faces of $\widehat{T}$ as follows: Let all triangles incident to vertex w be white; let all triangles obtained by deleting a vertex of degree 3 be white; for each of the $n_{4,2}$ vertices (which have degree 4 or higher in T and two assigned consecutive separable edges), let the triangle adjacent to both nonflippable edges be white; finally, color all remaining triangles of $\widehat{T}$ gray. See Figure 1 for an example.

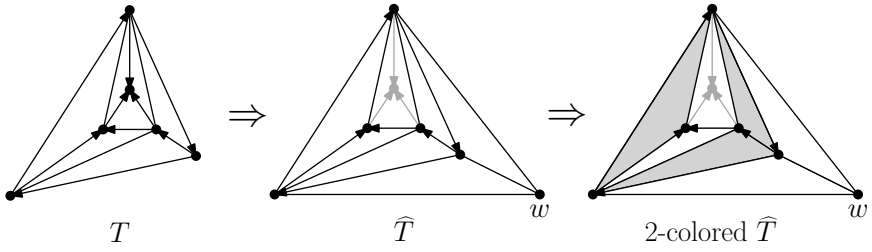


FIGURE 1. The 2-colored auxiliary triangulation $\widehat{T}$ of a triangulation T.

Under this coloring, the number of white faces is $m_{\text{white}} = h + n_{4,2} + n_3$. Using (6), the number of gray faces is

$$\begin{aligned} m_{\text{gray}} &= m - m_{\text{white}} \\ &= (2h + 2n_{4,2} + 2n_{4,1} + 2n_{4,0} - 2) - (h + n_{4,2} + n_3) \\ &= h + n_{4,2} + 2n_{4,1} + 2n_{4,0} - n_3 - 2. \end{aligned} \tag{7}$$

Putting it all together. Observe that, if a flippable edge e of T lies on the common boundary of two white triangles in the auxiliary graph $\widehat{T}$, then e is isolated. That is, if e is a nonisolated flippable edge in T, then it is on the boundary of a gray triangle in $\widehat{T}$. Since every gray triangle has three edges, the number of nonisolated flippable edges in T is at most $3m_{\text{gray}}$. Substituting this into our bound (7) on the number of simultaneously flippable edges, we have

$$\begin{aligned} f_{\text{sim}} &\geq f - \frac{2}{3} f_1 \geq f - 2m_{\text{gray}} \\ &= (h + n_{4,2} + 2n_{4,1} + 3n_{4,0} - 3) - 2(h + n_{4,2} + 2n_{4,1} + 2n_{4,0} - 2 - n_3) \\ &= 2n_3 - h - n_{4,2} - 2n_{4,1} - n_{4,0} + 1. \end{aligned} \tag{8}$$

Finally, combining twice (1) minus three times (4), we obtain

$$\begin{aligned} 2n - 3 \cdot \frac{3n}{5} &\leq 2(h + n_{4,2} + n_{4,1} + n_{4,0} + n_3) - 3(h + n_{4,2} + 2n_{4,1} + 3n_{4,0}); \\ \frac{n}{5} &\leq 2n_3 - h - n_{4,2} - 4n_{4,1} - 7n_{4,0} \\ &< 2n_3 - h - n_{4,2} - 2n_{4,1} - n_{4,0} + 1 \\ &\leq f_{\text{sim}}. \end{aligned} \tag{9}$$

Under the condition (3), we have proved a lower bound of $f_{\text{sim}} > n/5$; otherwise we have $f_{\text{sim}} \geq (n-4)/5$, as required.

2 Upper bound constructions

In this section we construct an infinite family of geometric triangulations with at most $(n-4)/5$ simultaneously flippable edges. This family includes all triangulations constructed by Galtier *et al.* [**3**]. First observe that a straight line drawing of K_4 has no flippable edge. We introduce two operations that each increase the number of vertices by 5, and the maximum number of simultaneously flippable edges by one.

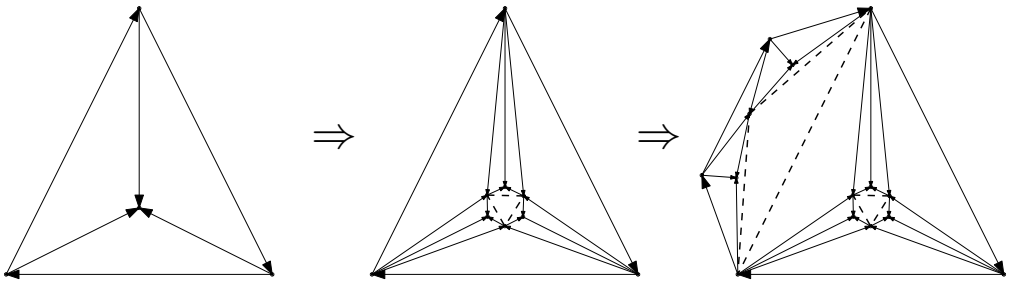


FIGURE 2. The two operations applied successively to K_4.

One operation replaces an interior vertex of degree 3 by a configuration of 6 vertices as shown at left in Fig. 2. The other operation adds 5 vertices in a close neighborhood of a hull edge as shown at right in Fig. 2. Note that both operations maintain the property that the triangles adjacent to the convex hull have no flippable edges. Each operation creates three new flippable edges, which form a triangle, so no two of them are simultaneously flippable. Each operation increases $h+n_{4,2}$ by 3 and n_3 by 2, as expected based on the previous section.

Let $\mathcal{F}_{\text{sim}}$ denote the family of all geometric triangulations obtained from K_4 via applying an arbitrary sequence of the two operations. Then every triangulation $T \in \mathcal{F}_{\text{sim}}$ on n vertices has at most $(n-4)/5$ simultaneously flippable edges, attaining our lower bound for $f_{\text{sim}}(n)$. We note that all upper bound constructions by Galtier *et al.* [**3**] can be obtained by applying our 2nd operation successively to all sides, starting from K_4.

References

[1] P. Bose and F. Hurtado, Flips in planar graphs, *Comput. Geom. Theory Appl.* **42** (1) (2009), 60–80.

[2] A. Dumitrescu, A. Schulz, A. Sheffer, and Cs. D. Tóth, Bounds on the maximum multiplicity of some common geometric graphs, in *Proc. STACS*, LIPICS, 2011, to appear.

[3] J. Galtier, F. Hurtado, M. Noy, S. Pérennes, and J. Urrutia, Simultaneous edge flipping in triangulations, *Internat. J. Comput. Geom. Appl.* **13** (2) (2003), 113–133.

[4] M. Hoffmann, M. Sharir, A. Sheffer, Cs. D. Tóth, and Emo Welzl, Counting plane graphs: Flippability and its applications, submitted manuscript, 2011. `http://arxiv.org/abs/1012.0591.`

[5] F. Hurtado, M. Noy, and J. Urrutia, Flipping edges in triangulations, *Discrete Comput. Geom.* **22** (1999), 333–346.

[6] J. Urrutia, Flipping edges in triangulations of point sets, polygons and maximal planar graphs, invited talk at the *1st Japan Conf. on Discrete and Comput. Geom. (Tokyo, November 17–20, 1997).* `http://www.matem.unam.mx/~urrutia/online_papers/TrianSurv.pdf`.

Blocking the k-holes of point sets on the plane

Javier Cano[1], Alfredo García[2], Ferran Hurtado[3], Toshinori Sakai[4], Javier Tejel[2], Jorge Urrutia[1]

[1] Instituto de Matemáticas, UNAM, Mexico
j_cano@uxmcc2.iimas.unam.mx, urrutia@matem.unam.mx

[2] Departamento de Métodos Estadísticos, IUMA, Universidad de Zaragoza, Spain
olaverri@unizar.es, jtejel@unizar.es

[3] Departament de Matemàtica Aplicada II, UPC, Spain
Ferran.Hurtado@upc.edu

[4] Research Institute of Educational Development, Tokai University, Japan
sakai@tokai-u.co.jp

Abstract. Let P be a set of n points in the plane in general position. A subset h_k of k points of P is called a k-hole if there is no element of P contained in the interior of the convex hull of h_k. A set B of points blocks the k-holes of P if any k-hole of P contains an element of B in its interior. In this paper we establish upper and lower bounds on the sizes of k-hole blocking sets.

Introduction

Let P be a set of n points on the plane in general position. We say that P is in *convex position* if the elements of P are the vertices of a convex polygon. A convex polygon $\mathcal{Q}$ with k vertices is called a *k-gon* of P if all of its vertices belong to P, and $\mathcal{Q}$ is a *k-hole* of P if it contains no element of P in its interior. A point b *blocks* a k-hole $\mathcal{Q}$ of P if it belongs to the interior of $\mathcal{Q}$. A set of points B is a *k-hole blocking set* of P ("k-blocking set of P" for short) if every k-hole of P is blocked by at least one element of B.

The problem of finding point sets that block all the 3-holes of a point set has been studied for some time now. It is known that, if a point set P with n elements has c points on its convex hull, then the 3-holes of P can be blocked with exactly $2n - c + 3$ points; see Katchalski and Meir [**4**], and Czyzowicz, Kranakis and Urrutia [**1**]. Recently, Sakai and Urrutia proved in [**6**] that there are point sets such that $2n - o(n)$ points are necessary to block all their 4-holes. Surprisingly, the problem changes substantially for k-blocking sets, $k \geq 5$. We will show that there are point sets, both in general and in convex position, for which the number of points needed to block their 5-holes is as low as a fifth of the number of triangles in a triangulation of the respective point set. In fact, the number of points needed to block the 5-holes of a point set depends on the geometry of the specific point set, unlike the case of blocking its triangles. For example, not all sets P of n points in convex position require the same number of 5-blockers. It is worth mentioning that the case $k = 2$, *i.e.*, blocking the visibility between pairs of points, has also received attention recently; see [**5**] and the references there.

[1]Partially supported by projects MTM2006-03909 (Spain) and SEP-CONACYT 80268 (Mexico).
[2]Partially suported by projects MTM2009-07242 and E58-DGA.
[3]Partially supported by projects MTM2009-07242 and Gen. Cat. DGR 2009SGR1040.

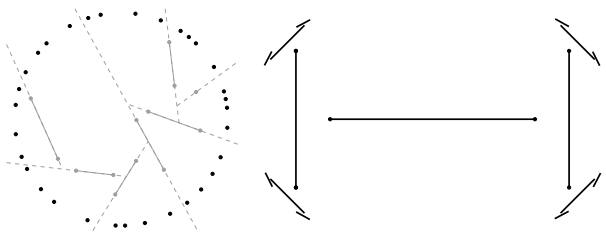


FIGURE 1. (a) Illustration of Theorem 1.1. (b) Point set $\mathcal{X}_4$.

1 Blocking the 5-holes of point sets

In this section we study the problem of blocking the 5-holes of point sets on the plane. We consider first point sets in convex position, and then point sets in general position.

1.1 Point sets in convex position

Theorem 1.1. *Let P a set of n points in convex position. Then any 5-blocking set for P has at least $2\lceil \frac{n}{4} \rceil - 3$ elements.*

Proof. Let B be a 5-blocking set of P with r elements. Let $\mathcal{M}$ be a planar geometric matching of maximum cardinality of the elements of B; that is, a set of disjoint pairs of the elements of B such that the line segments $\{\ell_1, \ldots, \ell_{\lfloor \frac{r}{2} \rfloor}\}$ joining them do not intersect. One at a time, extend them until they hit a line segment or a previously extended segment; some of them might be extended to semi-lines or lines. When r is odd, take a line segment that passes through the unmatched element of B and proceed as before; see Figure 1(a).

This will give us a decomposition of the plane into exactly $\lceil \frac{r}{2} \rceil + 1$ convex regions. Each of these regions can contain at most 4 elements of P; otherwise we would have an unblocked 5-hole. Then $|B| = r \geq 2\left\lceil \frac{n}{4} \right\rceil - 3$. □

Károlyi, Pach and Tóth **[3]** constructed families of point sets which they called *almost convex sets* as follows: Let $\mathcal{R}_1$ be a set of two points in the plane. Assume that we already defined $\mathcal{R}_1, \ldots, \mathcal{R}_j$ such that

(1) $\mathcal{X}_j := \mathcal{R}_1 \cup \cdots \cup \mathcal{R}_j$ is in general position,
(2) the vertex set of the convex hull Γ_j of $\mathcal{X}_j$ is $\mathcal{R}_j$, and
(3) any triangle determined by $\mathcal{R}_j$ contains precisely one point of $\mathcal{X}_j$ in its interior.

Let $z_1, \ldots, z_r$ denote the vertices of Γ_j in clockwise order around Γ_j, and let $\varepsilon_j, \delta_j > 0$. For any $1 \leq i \leq r$, let ℓ_i denote the line through z_i orthogonal to the bisector of the angle of Γ_j at z_i. Let z_i' and z_i'' be the two points in ℓ_i at distance ε_j from z_i. Now move z_i' and z_i'' away from Γ_j by a distance δ_j in the direction orthogonal to ℓ_i, and denote the resulting points by u_i' and u_i'', respectively.

We can choose ε_j and δ_j to be sufficiently small such that $\mathcal{R}_{j+1} := \{u_i', u_i'' | i = 1, \ldots, r\}$ also satisfies the above conditions. Conditions 1 and 2 are straightforward, so we will verify only the third.

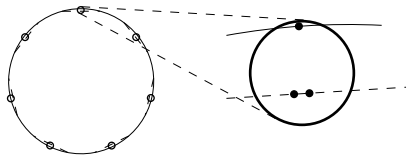
If $a \in \{u_i', u_i''\}$, $b \in \{u_m', u_m''\}$ and $c \in \{u_s', u_s'\}$ are three points of $\mathcal{R}_{j+1}$, for three distinct indices i, m, s, then any point of $\mathcal{X}_{j+1} := R_{j+1} \cup \mathcal{X}_j$ which belongs to the interior of Δabc must coincide with the point of $\mathcal{X}_j$ in the interior of $\Delta z_i z_m z_s$. If we have $a = u_i'$, $b = u_i''$ and $c \in \{u_m', u_m''\}$, with $i \neq m$, then the only point inside Δabc is z_i. Clearly $|\mathcal{X}_m| = 2^{m+1} - 2$ and $|\mathcal{R}_m| = 2^m$, for $m \geq 1$. See Figure 1(b). Now we prove:

Theorem 1.2. *There is a point set P in convex position with $n = 2^m$ that has a 5-blocking set with only $\frac{n}{2} - 2$ elements.*

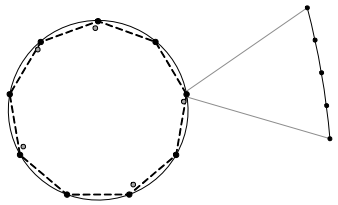
Proof. Let $P = \mathcal{R}_m$ and $B = \mathcal{X}_{m-2}$. Then $|P| = n$ and $|B| = \frac{n}{2} - 2$. We will show that B is a 5-hole blocking set for P. Suppose that B is not a 5-hole blocking set for P; then we have a 5-hole of P with no point of B in its interior. Take a triangulation of such a 5-hole —it will have 3 triangles of P. By construction, each of them contains exactly one element of $\mathcal{X}_{m-1}$, since $B = \mathcal{X}_{m-1} \setminus \mathcal{R}_{m-1}$. Then these three points have to be elements of $\mathcal{R}_{m-1}$ and they form a triangle contained in the 5-hole. By construction, such a triangle contains precisely one element of $\mathcal{X}_{m-2}$. Now, since $B = \mathcal{X}_{m-2}$, the 5-hole contains an element of B, which is a contradiction. Thus our result follows. □

1.2 Points in general position

Observe that there are point sets in general position for which roughly $\frac{2n}{3}$ points are necessary to block all their 5-holes. Take a set of points P that admits a convex pentagonization of its convex hull, and whose convex hull has five vertices. The number of pentagons in any pentagonization of the convex hull of P is $\lfloor \frac{2n-7}{3} \rfloor$; clearly any 5-blocking set of P has at least $\lfloor \frac{2n-7}{3} \rfloor$ points. We show next that there exist, surprisingly, families of point sets for which all of their 5-holes can be blocked with fewer than $\lfloor \frac{2n-7}{3} \rfloor$ points.



(a) A point set in general position in which $\frac{n}{3} - 2$ points are sufficient and necessary to block all of its convex 5-holes.



(b) The general construction when $k = 11$.

FIGURE 2

Theorem 1.3. *For any m there is a point set P in general position with $n = 3m$ points such that $m - 2$ points are sufficient and necessary to block all the 5-holes of P.*

Proof. Suppose that m is odd. Take a circle $\mathcal{C}$ and m sufficiently small disjoint chords $\{\mathcal{D}_1, \ldots, \mathcal{D}_m\}$ of $\mathcal{C}$ of equal length and evenly placed along $\mathcal{C}$. Each chord $\mathcal{D}_i$ determines a small arc $\mathcal{A}_i$ of $\mathcal{C}$, joining its endpoints. For each chord $\mathcal{D}_i$ select three points of the plane as follows: The first one is the midpoint of $\mathcal{A}_i$, and two points on $\mathcal{D}_i$ are equidistant and close enough to its mid-point so that the shaded region shown in Figure 2(a) is empty. We can think that these 3 points become one *fat point* of an m point set S_m in convex position.

Note that any convex 5-hole of P has at most two vertices in each fat point of S_m. Thus any 5-hole of P contains a point in at least three fat points of S_m. Let P' be the subset of P containing the points in the middle of $\mathcal{A}_i$, $i = 1, \ldots, m$. It is known **[1, 4]** that the set of triangles of P' can be blocked with a set Q_m of $m-2$ points. It is now easy to see that these points can be chosen in such a way that they also block any triangle

containing a point in three different fat vertices of S_m. It is not hard to see that we need at least $m-2$ points to block all the 5-holes of P. For n even, we use a similar construction. Our result follows. □

To finish this section, we prove:

Theorem 1.4. *Let P be a set of points in general position. Then any 5-blocking set of P has at least $2\lceil \frac{n}{9} \rceil - 3$ points.*

As in the proof of Theorem 1.1, we match the points of a 5-blocking set and subdivide the plane into convex regions. The main difference is that we now use a well known result of Harborth [**2**] which states that a point set with ten points always has a 5-hole.

2 Blocking *k*-holes for larger *k*

Now we consider the problem of blocking convex k-holes, $k \geq 6$. Let P be a set of n points in convex position. By a similar argument as in the proof of Theorem 1.1, it can be verified that any k-blocking set for P has at least $2\lceil \frac{n}{k-1} \rceil - 3$ elements. This bound is essentially tight.

To see the tightness for odd k, construct a point set P in the following way: First define integers m and r by $n = \frac{k-1}{2}\, m + r$, $0 < r < \frac{k-1}{2}$ (here we assume further that $r \neq 0$). We have $m = \lfloor \frac{2n}{k-1} \rfloor$. Let $Q = \{q_1, \ldots, q_{m+1}\}$ be the set of vertices of a regular $(m+1)$-gon, and let C be the circumcircle of this polygon. We replace each q_i by $\frac{k-1}{2}$ points lying on a sufficiently short arc of C (Figure 2(b)), except q_{m+1}, which we replace by r points. Denote by P_i the set of these $\frac{k-1}{2}$ or r points, and let $P = P_1 \cup \cdots \cup P_{m+1}$.

Then any k-hole with vertices in P has vertices in at least three P_i's. Thus the elements of a triangle blocking set for Q (or the points obtained by shifting them slightly if necessary) can block all convex k-holes of P. As in the proof of Theorem 1.3, take a triangle blocking set for Q with $(m+1) - 2 = \lfloor \frac{2n}{k-1} \rfloor - 1$ elements, which will also block all k-holes of P.

References

[1] J. Czyzowicz, E. Kranakis, J. Urrutia, Guarding the convex subsets of a point set, in *12th Canadian Conference on Computational Geometry*, Fredericton, New Brunswick, Canada, 2000, 47–50.

[2] H. Harborth, Konvexe, Fünfecke in ebenen Punktmengen. *Elemente Math.* **33** (1978), 116–118.

[3] G. Károlyi, J. Pach, G. Tóth, A modular version of the Erdős–Szekeres theorem, *Studia Scientiarum Mathematicarum Hungarica* **51** (2001), 245–260.

[4] M. Katchalski, A. Meir, On empty triangles determined by points in the plane, *Acta Math. Hungar.* **51** (1988), 323–328.

[5] A. Pór, D. R. Wood, On visibility and blockers, *J. Computational Geometry* **1(1)** (2010), 29–40.

[6] T. Sakai, J. Urrutia, Covering the convex quadrilaterals of point sets, *Graphs and Combinatorics* **38** (2007), 343–358.

Compatible matchings in geometric graphs

O. Aichholzer[1], A. García[2], F. Hurtado[3], J. Tejel[2]

[1] Institute for Software Technology, Graz University of Technology, Austria
oaich@ist.TUGraz.at

[2] Departamento de Métodos Estadísticos, IUMA, Universidad de Zaragoza, Spain
olaverri@unizar.es, jtejel@unizar.es

[3] Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, Spain
ferran.hurtado@upc.edu

Abstract. Two non-crossing geometric graphs on the same set of points are compatible if their union is also non-crossing. In this paper, we prove that every graph G that has an outerplanar embedding admits a non-crossing perfect matching compatible with G. Moreover, for non-crossing geometric trees and simple polygons, we study bounds on the minimum number of edges that a compatible non-crossing perfect matching must share with the tree or the polygon. We also give bounds on the maximal size of a compatible matching (not necessarily perfect) that is disjoint from the tree or the polygon.

Introduction

A *geometric graph* is a simple graph G where the vertex set $V(G)$ is a finite set of points S in the plane and each edge in $E(G)$ is a closed straight-line segment connecting two points in S. A geometric graph is *non-crossing* if no two edges cross except at a common vertex.

Throughout the paper, all the graphs considered will be geometric and non-crossing. For this reason, we will use the term "graph" ("tree", "matching"...) meaning that the graph (tree, matching...) is geometric and non-crossing. Moreover, we will assume that no three points are collinear.

Two graphs are said to be *compatible* if they have the same vertex set and their union is non-crossing. A graph that is compatible with a given graph G will be called *G-compatible*. In addition, if they have no edge in common, we call them *disjoint*.

Given a set S of n points in the plane and a graph G on S, in this paper we study the two following problems of compatibility. On one hand, to find a *perfect matching* M such that it is G-compatible and the number of common edges between M and G is minimum. On the other hand, to find a *matching* M such that it is G-compatible, disjoint from G, and the number of edges of M is maximum. Similar problems on compatible graphs have been studied in **[2, 3, 4]** and some related augmentation problems for geometric graphs appear in **[1, 5]**.

Since these numbers depend on the set of points S and on the graph G, we have focused on bounding the values defined below. Given a set S with an even number n of points, a tree $T(S)$ on S, and a $T(S)$-compatible perfect matching M, let us define $m_{(T(S),\,M)}$ to be the number of edges of M not contained in $T(S)$. Let us also define $m_{\text{Tree}}(n) = \min_{|S|=n}\{\min_{T(S)}\{\max_M m_{(T(S),\,M)}\}\}$ for n even, i.e., the worst case of the

[2]Partially supported by projects MTM2009-07242 and E58-DGA.
[3]Partially supported by projects MTM2009-07242 and Gen. Cat. DGR 2009SGR1040.

maximal number of non-shared edges. In the case of non necessarily perfect matchings, let $d_{(T(S),\,M)}$ be the number of edges of a $T(S)$-compatible matching M that is disjoint from $T(S)$ and let $d_{\text{Tree}}(n) = \min_{|S|=n}\{\min_{T(S)}\{\max_M d_{(T(S),\,M)}\}\}$. Note that the definitions of $m_{\text{Tree}}(n)$ and $d_{\text{Tree}}(n)$ are identical, except that the maximum is taken over different families of matchings.

By defining in a similar way the values $m_{\text{Polygon}}(n)$ and $d_{\text{Polygon}}(n)$ for simple polygons, the main results that we have obtained are:

Theorem 0.1. *For n arbitrarily large,*

$$\begin{aligned} 0 \leq m_{\text{Tree}}(n) &\leq 13;\\ n/10 \leq d_{\text{Tree}}(n) &\leq n/4;\\ n/20 \leq m_{\text{Polygon}}(n) &\leq n/4;\\ (n-3)/4 \leq d_{\text{Polygon}}(n) &\leq n/3. \end{aligned}$$

1 Compatible perfect matchings

As a first result, we give the following theorem characterizing a set of graphs for which a compatible perfect matching always exists.

Theorem 1.1. *Given a set S of n (even) points and a graph G on S drawn as an outerplanar geometric graph on top of S, there is always a G-compatible perfect matching.*

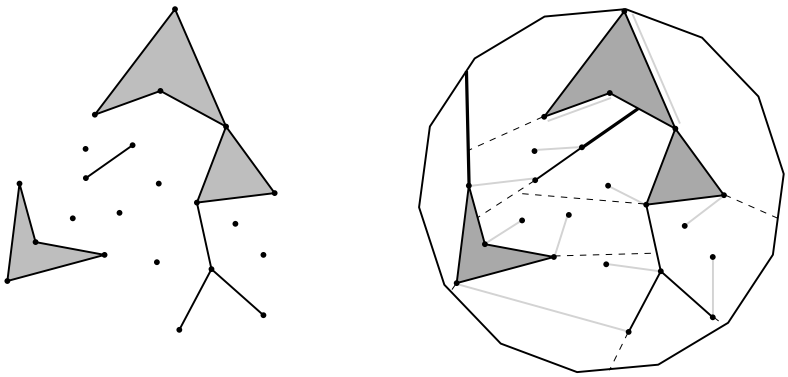


FIGURE 1. Obtaining a perfect matching for an outerplanar geometric graph.

Figure 1 shows an example of how to find this perfect matching. The method is similar to the one described in [**1**]. First, we add a big convex polygon passing through the top most point on S and containing all the points. Then, we join the non-trivial connected components of the graph by adding new edges (and consequently new vertices), until a (weakly) simple polygon is obtained. Given a component, the added edge (thick line in the figure) is part of the ray that emanates from the top most point of the component until it hits an edge. This ray bisects the reflex angle at the top most point. Note that the added vertices are convex in the polygon. Now, the (weakly) simple polygon is divided into convex regions by throwing rays (dashed lines in the figure) from the reflex vertices

of the polygon. Using the dual graph associated to this subdivision, we can guarantee an assignment of an even number of vertices to each region. Then, the perfect matching (gray edges in the figure) is obtained by matching the vertices of each convex region.

If we drop the condition of outerplanarity (all the vertices in the unbounded face), then a G-compatible perfect matching does not always exist. Figure 2(a) shows an example of a graph G formed by a tree (which is outerplanar) plus an edge. The seven points $\{a, b, c, d, e, f, g\}$ can only be linked with one of the six points $\{1, 2, 3, 4, 5, 6\}$, and hence there are no perfect matchings compatible with this graph G.

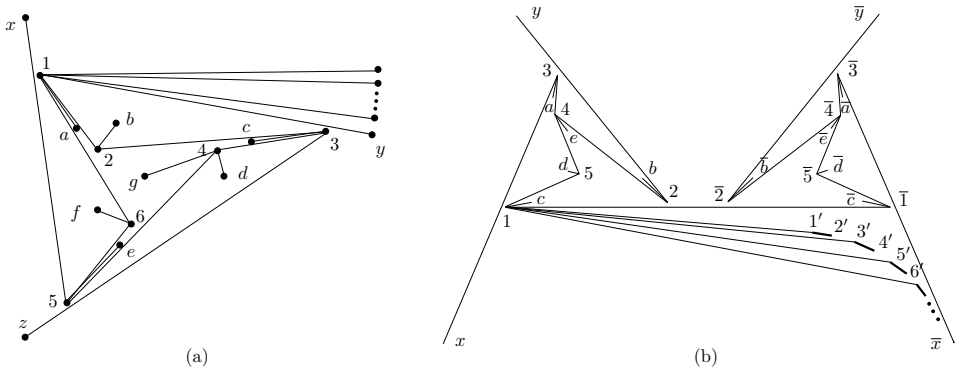


FIGURE 2. On the left, a graph G without any G-compatible perfect matching. On the right, a tree T for which any T-compatible perfect matching must share almost all its edges with the tree.

2 Compatible matchings for trees and simple polygons

In this section, we briefly explain how to obtain the bounds given in Theorem 0.1. The upper bounds are obtained by analyzing special cases of trees and simple polygons. Maybe, the most surprising bound is the upper bound for $m_{\text{Tree}}(n)$. This upper bound, which is a constant instead of a function of n, is based on the tree shown in Figure 2(b). For this tree, any compatible perfect matching must share almost all its edges with the tree (at least $n/2 - 13$ edges). The five points $\{1, 2, 3, 4, 5\}$ can only be matched with the points $\{a, b, c, d, e\}$ and the same for the points $\{\overline{1}, \overline{2}, \overline{3}, \overline{4}, \overline{5}\}$ and $\{\overline{a}, \overline{b}, \overline{c}, \overline{d}, \overline{e}\}$. Then, necessarily point $1'$ has to be matched to point $2'$, point $3'$ to point $4'$ and so on. Only the 24 points not in the convex chain $1', 2', 3'...$ and the two last points of the chain, $m' - 1$ and m', can be matched with edges not in the tree.

Figure 3 shows the graphs used to obtain the upper bounds on $d_{\text{Tree}}(n)$ (Figure 3a), $m_{\text{Polygon}}(n)$ (Figure 3b) and $d_{\text{Polygon}}(n)$ (Figure 3c). In Figure 3a, to obtain a matching without sharing edges, we can link the points on the zig-zag path among them or link a point (or more) of an arrow with some point on the zig-zag path. Since the number of points on the zig-zag path is $n/4$, the size of a matching compatible and disjoint from the tree is at most $n/4$ edges. In Figure 3b, we have a simple polygon P formed by two convex chains, C_1 and C_2, with $3k - 1$ and $k + 1$ points, respectively. Observe that, to obtain a perfect matching M, we cannot join two non-consecutive points of C_1. Hence, we need to use internal diagonals of P joining a point of C_1 to a point of C_2 (except the first point), and then the number of this type of diagonals is at most k in M. Figure 3c is analyzed in a similar way.

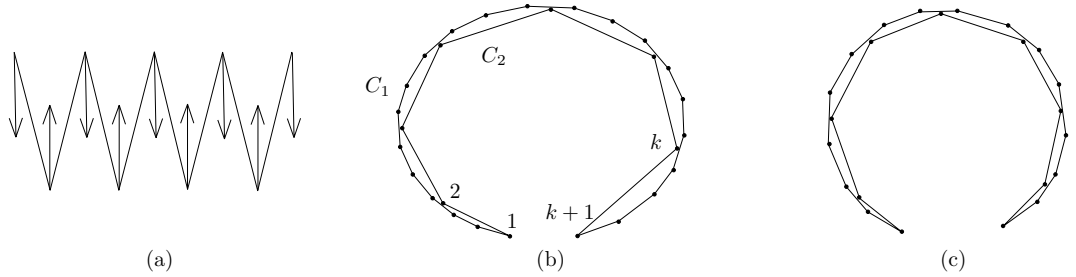


FIGURE 3. Graphs used to obtain the upper bounds on $d_{\text{Tree}}(n)$, $m_{\text{Polygon}}(n)$ and $d_{\text{Polygon}}(n)$.

Regarding the lower bounds, we can obtain them by constructing specific (perfect) matchings for any tree or polygon. For instance, given a tree T, we construct three T-compatible matchings, disjoint from T, of sizes at least $\lfloor l/4 \rfloor$, $\lfloor (n-2l)/4 \rfloor$ and $\lfloor (n-l)/6 \rfloor$, respectively, where l is the number of leaves of T. So, if the number of leaves is for example 2, then we have a matching of size $\lfloor (n-4)/4 \rfloor$ (the maximum of the three previous sizes) and, if the number of leaves is $n-1$, then we have a matching of size $\lfloor (n-1)/4 \rfloor$. With this method, the worst case appears when the number of leaves is $2n/5$. In this case, we can only guarantee a matching of size $n/10$.

The methods for obtaining the (perfect) matchings in the proof of Theorem 0.1 are based on several technical results on simple polygons. For example, let P be a simple polygon on S having r reflex vertices and c convex vertices, and let $S_0 \subseteq S$ be a subset of vertices containing all the reflex vertices and c_0 convex vertices. Then, we can show that there is a P-compatible matching among the vertices of S_0 of size at least $\lfloor (c_0-1)/2 \rfloor$ edges with all its edges being internal diagonals of P. We can also prove that if S_0 has h chains (a sequence $\{p_k, p_{k+1}, \ldots, p_{k+(l-1)}\}$ of consecutive vertices of P is a chain if all these vertices belong to S_0 and neither p_{k-1} nor p_{k+l} belong to S_0), then there is a P-compatible matching among the vertices of S_0 that is disjoint from P, its size is at least $\lfloor h/2 \rfloor$ edges and each edge of the matching is an internal diagonal of P.

Finally, let us remark that, for n points in convex position, we can prove tight bounds for trees and paths. In the case of trees, $m_{\text{Tree}}(n) = \lceil (n-2)/6 \rceil$ and $d_{\text{Tree}}(n) = \lceil n/4 \rceil$. For paths, $m_{\text{Path}}(n) = \lceil n/4 \rceil$ and $d_{\text{Path}}(n) = \lceil (2n)/5 \rceil$, where $m_{\text{Path}}(n)$ and $d_{\text{Path}}(n)$ are defined in a similar way to $m_{\text{Tree}}(n)$ and $d_{\text{Tree}}(n)$.

References

[1] M. Abellanas, A. García, F. Hurtado, J. Tejel and J. Urrutia, Augmenting the connectivity of geometric graphs, *Computational Geometry* **40** (2008), 220–230.

[2] O. Aichholzer, S. Bereg, A. Dumitrescu, A. García, C. Huemer, F. Hurtado, M. Kano, A. Márquez, D. Rappaport, S. Smorodinsky, D. Souvaine, J. Urrutia and D. Wood, Compatible geometric matchings, *Computational Geometry* **42** (2009), 617–626.

[3] A. García, C. Huemer, F. Hurtado and J. Tejel, Árboles geométricos compatibles, In *Proc. XII Encuentros de Geometría Computacional*, Valladolid, 2007, 161–167.

[4] M. Ishaque, D. L. Souvaine and C. D. Tóth, Disjoint compatible geometric matchings, In *Proc. 27th Annual Symposium on Computational Geometry* (Paris, 2011), ACM Press, to appear.

[5] C. D. Tóth, Connectivity augmentation in planar straight line graphs, *Europ. J. of Combinatorics*, to appear. (Preliminary version in: *Proc. Intl. Conf. on Topological and Geometric Graph Theory*, Paris, 2008, pp. 51–54.)

Contact numbers for congruent sphere packings

Károly Bezdek[1]

[1] Department of Mathematics and Statistics, University of Calgary, Canada
bezdek@math.ucalgary.ca

Abstract. We give estimates on the maximum number of touching pairs in a packing of n congruent spheres in Euclidean 3-space for all $n > 1$.

Introduction

Let $\mathbb{E}^d$ denote the d-dimensional Euclidean space. The *contact graph* of an arbitrary finite packing of unit balls (i.e., of an arbitrary finite family of non-overlapping unit balls) in $\mathbb{E}^d$ is the (simple) graph whose vertices correspond to the packing elements and whose two vertices are connected by an edge if and only if the corresponding two packing elements touch each other. One of the most basic questions on contact graphs is to find the maximum number of edges that a contact graph of a packing of n unit balls can have in $\mathbb{E}^d$. In 1974 Harborth **[4]** proved the following optimal result in $\mathbb{E}^2$: the maximum number $c(n)$ of touching pairs in a packing of n congruent circular disks in $\mathbb{E}^2$ is precisely $\lfloor 3n - \sqrt{12n-3} \rfloor$, implying that

$$\lim_{n \to +\infty} \frac{3n - c(n)}{\sqrt{n}} = \sqrt{12} = 3.464\ldots \ .$$

Some years later, the author **[1]** proved that the number of touching pairs in an arbitrary packing of $n > 1$ unit balls in $\mathbb{E}^3$ is less than

$$6n - \frac{1}{8}\left(\frac{\pi}{\sqrt{18}}\right)^{-\frac{2}{3}} n^{\frac{2}{3}} = 6n - 0.152\ldots n^{\frac{2}{3}}.$$

The main purpose of this extended abstract is to announce further improvements on the latter result. In order to state our theorem in a proper form, we need to introduce a bit of additional terminology. If $\mathcal{P}$ is a packing of n unit balls in $\mathbb{E}^3$, then let $C(\mathcal{P})$ stand for the number of touching pairs in $\mathcal{P}$, that is, let $C(\mathcal{P})$ denote the number of edges of the contact graph of $\mathcal{P}$ and call it the *contact number* of $\mathcal{P}$. Moreover, let $C(n)$ be the largest $C(\mathcal{P})$ for packings $\mathcal{P}$ of n unit balls in $\mathbb{E}^3$. Finally, let us imagine that we generate packings of n unit balls in $\mathbb{E}^3$ in such a special way that each and every center of the n unit balls chosen is a lattice point of the face-centered cubic lattice with minimal non-zero lattice vectors of length 2. Then let $C_{\mathrm{fcc}}(n)$ denote the largest possible contact number obtained in this way for packings of n unit balls. (Recall that, according to **[3]**, this lattice gives the largest possible density for unit ball packings in $\mathbb{E}^3$, namely $\frac{\pi}{\sqrt{18}}$ with each ball touched by 12 others.) Clearly,

$$C_{\mathrm{fcc}}(2) = C(2) = 1, \quad C_{\mathrm{fcc}}(3) = C(3) = 3, \quad C_{\mathrm{fcc}}(4) = C(4) = 6.$$

The following is our main theorem.

CRM Documents, vol. **8**, Centre de Recerca Matemàtica, Bellaterra (Barcelona), 2011

Theorem 0.1. (i) $C(n) < 6n - 0.695n^{\frac{2}{3}}$ *for all* $n \geq 2$.

(ii) $C_{\mathrm{fcc}}(n) < 6n - \frac{3\sqrt[3]{18\pi}}{\pi} n^{\frac{2}{3}} = 6n - 3.665\ldots n^{\frac{2}{3}}$ *for all* $n \geq 2$.

(iii) $6n - \sqrt[3]{486}n^{\frac{2}{3}} < C_{\mathrm{fcc}}(n) \leq C(n)$ *for all* $n = \frac{k(2k^2+1)}{3}$ *with* $k \geq 2$.

(iv) $C_{\mathrm{fcc}}(5) = C(5) = 9$, $C_{\mathrm{fcc}}(6) = C(6) = 12$, $C_{\mathrm{fcc}}(7) = C(7) = 15$, $C_{\mathrm{fcc}}(8) = C(8) = 18$, $C_{\mathrm{fcc}}(9) = C(9) = 21$, $C(10) \geq 25$, $C(11) \geq 29$, $C(12) \geq 33$, *and* $C(13) \geq 36$.

As an immediate result, we get

Corollary 0.2.

$$0.695 < \frac{6n - C(n)}{n^{\frac{2}{3}}} < \sqrt[3]{486} = 7.862\ldots$$

for all $n = \frac{k(2k^2+1)}{3}$ *with* $k \geq 2$.

The following was noted in [**1**]. Due to the Minkowski difference body method, the family $\mathcal{P}_{\mathbf{K}} = \{\mathbf{t}_1 + \mathbf{K}, \mathbf{t}_2 + \mathbf{K}, \ldots, \mathbf{t}_n + \mathbf{K}\}$ of n translates of the convex body $\mathbf{K}$ in $\mathbb{E}^d$ is a packing if and only if the family $\mathcal{P}_{\mathbf{K_o}} = \{\mathbf{t}_1 + \mathbf{K_o}, \mathbf{t}_2 + \mathbf{K_o}, \ldots, \mathbf{t}_n + \mathbf{K_o}\}$ of n translates of the symmetric difference body $\mathbf{K_o} = \frac{1}{2}(\mathbf{K} + (-\mathbf{K}))$ of $\mathbf{K}$ is a packing in $\mathbb{E}^d$. Moreover, the number of touching pairs in the packing $\mathcal{P}_{\mathbf{K}}$ is equal to the number of touching pairs in the packing $\mathcal{P}_{\mathbf{K_o}}$. Thus, for this reason and for the reason that if $\mathbf{K}$ is a convex body of constant width in $\mathbb{E}^d$, then $\mathbf{K_o}$ is a ball of $\mathbb{E}^d$, and Theorem 0.1 extends in a straightforward way to translative packings of convex bodies of constant width in $\mathbb{E}^3$.

Also, we mention that the nature of contact numbers changes dramatically for non-congruent sphere packings. For more details on that, we refer the interested reader to the elegant paper [**6**] of Kuperberg and Schramm.

In the rest of this extended abstract we prove (ii) also because it motivates the more involved proof of (i), and then we give a short proof of (iii). Our proof of (iv) is based on a combinatorial and metric case analysis.

1 Proof of (ii)

First, recall that if Λ_{fcc} denotes the face-centered cubic lattice with minimal non-zero lattice vectors of length 2 in $\mathbb{E}^3$ and we place unit balls centered at each lattice point of Λ_{fcc}, then we get the fcc lattice packing of unit balls, labelled by $\mathcal{P}_{\mathrm{fcc}}$, in which each unit ball is touched by 12 others such that their centers form the vertices of a cuboctahedron. (Recall that a cuboctahedron is a convex polyhedron with 8 triangular faces and 6 square faces having 12 identical vertices, with 2 triangles and 2 squares meeting at each vertex, and 24 identical edges, each separating a triangle from a square. As such, it is a quasiregular polyhedron, i.e., an Archimedean solid, being vertex-transitive and edge-transitive.) Second, it is well-known (see [**2**] for more details) that the Voronoi cell of each unit ball in $\mathcal{P}_{\mathrm{fcc}}$ is a rhombic dodecahedron (the dual of a cuboctahedron) of volume $\sqrt{32}$ and thus, the density of $\mathcal{P}_{\mathrm{fcc}}$ is $\frac{\pi}{\sqrt{18}}$.

Now, let $\mathbf{B}$ denote the unit ball centered at the origin $\mathbf{o} \in \Lambda_{\mathrm{fcc}}$ of $\mathbb{E}^3$ and denote by $\mathcal{P} = \{\mathbf{c}_1 + \mathbf{B}, \mathbf{c}_2 + \mathbf{B}, \ldots, \mathbf{c}_n + \mathbf{B}\}$ the packing of n unit balls with centers $\{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_n\} \subset \Lambda_{\mathrm{fcc}}$ in $\mathbb{E}^3$ having the largest number $C_{\mathrm{fcc}}(n)$ of touching pairs among all packings of n unit balls being a sub-packing of $\mathcal{P}_{\mathrm{fcc}}$. ($\mathcal{P}$ might not be uniquely determined up to congruence, in which case $\mathcal{P}$ stands for any of those extremal packings.)

The following two facts follow from the above description of $\mathcal{P}_{\text{fcc}}$ in a straightforward way. Let $\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_{13}$ be 13 different members of $\mathcal{P}_{\text{fcc}}$ such that each ball of the family $\mathbf{B}_2, \mathbf{B}_3, \ldots, \mathbf{B}_{13}$ touches $\mathbf{B}_1$. Moreover, let $\bar{\mathbf{B}}_i$ be the closed ball concentric with $\mathbf{B}_i$ having radius $\bar{r} = \sqrt{2}$, $1 \leq i \leq 13$. Then the boundary $\text{bd}(\bar{\mathbf{B}}_1)$ of $\bar{\mathbf{B}}_1$ is covered by the balls $\bar{\mathbf{B}}_2, \bar{\mathbf{B}}_3, \ldots, \bar{\mathbf{B}}_{13}$, that is,

$$\text{bd}(\bar{\mathbf{B}}_1) \subset \cup_{j=2}^{13} \bar{\mathbf{B}}_j \ . \tag{1}$$

In fact, $\bar{r}$ is the smallest radius with the above property. Moreover,

$$\frac{n\text{vol}_3(\mathbf{B})}{\text{vol}_3(\bigcup_{i=1}^{n} (\mathbf{c}_i + \bar{r}\mathbf{B}))} < \frac{\pi}{\sqrt{18}} = 0.7404\ldots \ . \tag{2}$$

As a next step, we apply the isoperimetric inequality to $\bigcup_{i=1}^{n} (\mathbf{c}_i + \bar{r}\mathbf{B})$:

$$36\pi\text{vol}_3^2\left(\bigcup_{i=1}^{n} (\mathbf{c}_i + \bar{r}\mathbf{B})\right) \leq \text{svol}_2^3\left(\text{bd}\left(\bigcup_{i=1}^{n} (\mathbf{c}_i + \bar{r}\mathbf{B})\right)\right). \tag{3}$$

Thus, (2) and (3) yield in a straightforward way that

$$15.3532\ldots n^{\frac{2}{3}} = 4\sqrt[3]{18\pi}n^{\frac{2}{3}} < \text{svol}_2\left(\text{bd}\left(\bigcup_{i=1}^{n} (\mathbf{c}_i + \bar{r}\mathbf{B})\right)\right). \tag{4}$$

Now, assume that $\mathbf{c}_i + \mathbf{B} \in \mathcal{P}$ is tangent to $\mathbf{c}_j + \mathbf{B} \in \mathcal{P}$ for all $j \in T_i$, where $T_i \subset \{1, 2, \ldots, n\}$ stands for the family of indices $1 \leq j \leq n$ for which $\text{dist}(\mathbf{c}_i, \mathbf{c}_j) = 2$. Then let $\bar{S}_i = \text{bd}(\mathbf{c}_i + \bar{r}\mathbf{B})$ and let $\bar{\mathbf{c}}_{ij}$ be the intersection of the line segment $\mathbf{c}_i\mathbf{c}_j$ with $\bar{S}_i$ for all $j \in T_i$. Moreover, let $C_{\bar{S}_i}(\bar{\mathbf{c}}_{ij}, \frac{\pi}{6})$ (resp. $C_{\bar{S}_i}(\bar{\mathbf{c}}_{ij}, \frac{\pi}{4})$) denote the open spherical cap of $\bar{S}_i$ centered at $\bar{\mathbf{c}}_{ij} \in \bar{S}_i$ having angular radius $\frac{\pi}{6}$ (resp. $\frac{\pi}{4}$). Clearly, the family $\{C_{\bar{S}_i}(\bar{\mathbf{c}}_{ij}, \frac{\pi}{6}), j \in T_i\}$ consists of pairwise disjoint open spherical caps of $\bar{S}_i$; moreover,

$$\frac{\sum_{j\in T_i} \text{svol}_2\left(C_{\bar{S}_i}(\bar{\mathbf{c}}_{ij}, \frac{\pi}{6})\right)}{\text{svol}_2\left(\cup_{j\in T_i} C_{\bar{S}_i}(\bar{\mathbf{c}}_{ij}, \frac{\pi}{4})\right)} = \frac{\sum_{j\in T_i} \text{Sarea}\left(C(\mathbf{u}_{ij}, \frac{\pi}{6})\right)}{\text{Sarea}\left(\cup_{j\in T_i} C(\mathbf{u}_{ij}, \frac{\pi}{4})\right)}, \tag{5}$$

where $\mathbf{u}_{ij} = \frac{1}{2}(\mathbf{c}_j - \mathbf{c}_i) \in \mathbb{S}^2 = \text{bd}(\mathbf{B})$ and $C(\mathbf{u}_{ij}, \frac{\pi}{6}) \subset \mathbb{S}^2$ (resp. $C(\mathbf{u}_{ij}, \frac{\pi}{4}) \subset \mathbb{S}^2$) denotes the open spherical cap of $\mathbb{S}^2$ centered at $\mathbf{u}_{ij}$ having angular radius $\frac{\pi}{6}$ (resp. $\frac{\pi}{4}$). Now, the geometry of the cuboctahedron representing the 12 touching neighbours of an arbitrary unit ball in $\mathcal{P}_{\text{fcc}}$ (see also (1)) implies in a straightforward way that

$$\frac{\sum_{j\in T_i} \text{Sarea}\left(C(\mathbf{u}_{ij}, \frac{\pi}{6})\right)}{\text{Sarea}\left(\cup_{j\in T_i} C(\mathbf{u}_{ij}, \frac{\pi}{4})\right)} \leq 6(1 - \frac{\sqrt{3}}{2}) = 0.8038\ldots, \tag{6}$$

with equality when 12 spherical caps of angular radius $\frac{\pi}{6}$ are packed on $\mathbb{S}^2$.

Finally, as $\text{Sarea}\left(C(\mathbf{u}_{ij}, \frac{\pi}{6})\right) = 2\pi(1 - \cos\frac{\pi}{6})$ and $\text{svol}_2\left(C_{\bar{S}_i}(\bar{\mathbf{c}}_{ij}, \frac{\pi}{6})\right) = 2\pi(1 - \frac{\sqrt{3}}{2})\bar{r}^2$, it follows that (5) and (6) yield that

$$\text{svol}_2\left(\text{bd}\left(\bigcup_{i=1}^{n} \mathbf{c}_i + \bar{r}\mathbf{B}\right)\right) \leq 4\pi\bar{r}^2 n - \frac{1}{6(1-\frac{\sqrt{3}}{2})} 2\left(2\pi\left(1 - \frac{\sqrt{3}}{2}\right)\bar{r}^2\right) C_{\text{fcc}}(n)$$

$$= 8\pi n - \frac{4\pi}{3} C_{\text{fcc}}(n). \tag{7}$$

Thus, (4) and (7) imply that

$$4\sqrt[3]{18\pi}n^{\frac{2}{3}} < 8\pi n - \frac{4\pi}{3}C_{\mathrm{fcc}}(n). \tag{8}$$

From (8), the inequality $C_{\mathrm{fcc}}(n) < 6n - \frac{3\sqrt[3]{18\pi}}{\pi}n^{\frac{2}{3}} = 6n - 3.665\ldots n^{\frac{2}{3}}$ follows in a straightforward way for all $n \geq 2$. This completes the proof of (ii) in Theorem 0.1.

2 Proof of (iii)

It is rather easy to show that for any positive integer $k \geq 2$ there are $n(k) = \frac{2k^3+k}{3} = \frac{k(2k^2+1)}{3}$ lattice points of the face-centered cubic lattice Λ_{fcc} such that their convex hull is a regular octahedron $\mathbf{K} \subset \mathbb{E}^3$ of edge length $2(k-1)$ having exactly k lattice points along each of its edges. Now, draw a unit ball around each lattice point of $\Lambda_{\mathrm{fcc}} \cap \mathbf{K}$ and label the packing of the $n(k)$ unit balls obtained in this way by $\mathcal{P}_{\mathrm{fcc}}(k)$. It is easy to check that if the center of a unit ball of $\mathcal{P}_{\mathrm{fcc}}(k)$ is a relative interior point of an edge (resp. of a face) of $\mathbf{K}$, then the unit ball in question has 7 (resp. 9) touching neighbours in $\mathcal{P}_{\mathrm{fcc}}(k)$. Last but not least, any unit ball of $\mathcal{P}_{\mathrm{fcc}}(k)$ whose center is an interior pont of $\mathbf{K}$ has 12 touching neighbours in $\mathcal{P}_{\mathrm{fcc}}(k)$. Thus, the contact number $C\left(\mathcal{P}_{\mathrm{fcc}}(k)\right)$ of the packing $\mathcal{P}_{\mathrm{fcc}}(k)$ is equal to

$$6\,\frac{2(k-2)^3+(k-2)}{3} + 36\,\frac{(k-3)^2+(k-3)}{2} + 42(k-2) + 12 = 4k^3 - 6k^2 + 2k.$$

As a result, we get that

$$C\left(\mathcal{P}_{\mathrm{fcc}}(k)\right) = 6n(k) - 6k^2. \tag{9}$$

Finally, $\frac{2k^3}{3} < n(k)$ implies that $6k^2 < \sqrt[3]{486}n^{\frac{2}{3}}(k)$, and so (9) implies (iii) in a straightforward way.

Acknowledgement

This work was partially supported by a Natural Sciences and Engineering Research Council of Canada Discovery Grant.

References

[1] K. Bezdek, On the maximum number of touching pairs in a finite packing of translates of a convex body, *J. Combin. Theory Ser. A* **98/1** (2002), 192–200.
[2] L. Fejes Tóth, *Regular Figures*, Pergamon Press, Oxford, 1964.
[3] T. C. Hales, A proof of the Kepler conjecture, *Ann. Math.* **162/2–3** (2005), 1065–1185.
[4] H. Harborth, Lösung zu Problem 664A, *Elem. Math.* **29** (1974), 14–15.
[5] G. A. Kabatiansky and V. I. Levenshtein, Bounds for packings on a sphere and in space, *Problemy Peredachi Informatsii* **14** (1978), 3–25.
[6] G. Kuperberg and O. Schramm, Average kissing numbers for non-congruent sphere packings, *Math. Res. Lett.* **1/3** (1994), 339—344.
[7] K. Schütte and B. L. van der Waerden, Das Problem der dreizehn Kugeln, *Math. Ann.* **125** (1953), 325–334.

Geometric graphs in the plane lattice with L-line segments

Mikio Kano[1], **Kazuhiro Suzuki**[2]

[1] Department of Computer and Information Sciences, Ibaraki University, Hitachi, Ibaraki, 316-8511, Japan
kano@mx.ibaraki.ac.jp

[2] Department of Electronics and Informatics Frontier, Kanagawa University, Yokohama, Kanagawa, 221-8686, Japan
kazuhiro@tutetuti.jp

Abstract. An L-line segment in the plane consists of a vertical line segment and a horizontal line segment having a common endpoint. In this talk, we consider some problems on geometric graphs in the plane lattice, whose vertices are points of the plane lattice and whose edges are L-line segments.

Introduction

A geometric graph in the plane is a graph drawn in the plane whose edges are straight line segments. For a point x in the plane, an L-shaped line consisting of a vertical ray and a horizontal ray emanating from x is called an *L-line*. Similarly, an L-shaped line segment consisting of a vertical line segment and a horizontal line segment with common corner is called an *L-line segment* (see Fig. 1). We consider some problems on geometric graphs in the plane lattice whose edges are L-line segments. A set X of points in the plane is *in general position* if no three points of X lie on the same line. On the other hand, a set S of points in the plane lattice is said to be *in general position* if every vertical line and horizontal line passes through at most one point of S. Some results related to this paper can be found in [**1**].

1 Geometric spanning trees on two sets of points

For a set X of points in the plane, we can draw a non-crossing geometric spanning tree on X, each of whose edges is a line segment joining two points of X. This spanning tree is denoted by $tree(X)$. When a set R of red points and a set B of blue points are given in the plane in general position, the minimum number of crossings of $tree(R)$ and $tree(B)$ is given in the next theorem, in which $conv(X)$ denotes the convex hull of X.

Theorem 1.1 (Tokunaga [**3**])**.** *Let R and B be two disjoint sets of red points and blue points in the plane, respectively, such that $R \cup B$ is in general position. Let $\tau(R, B)$ denote the number of unordered pairs $\{x, y\}$ of vertices of $conv(R \cup B)$ such that one of $\{x, y\}$ is red and the other is blue, and xy is an edge of $conv(R \cup B)$. Then $\tau(B, R)$ is an even number, and the minimum number of crossings in $tree(R) \cup tree(B)$ among all*

[2]Partially supported by research grant by Japan Society for the Promotion of Science, Grant-in-Aid for Scientific Research (C).

pairs $(tree(R), tree(B))$ *is equal to*

$$\max\left\{\frac{\tau(R,B)-2}{2},\ 0\right\}.$$

In particular, $tree(R)$ *and* $tree(B)$ *can be drawn without crossings if and only if the inequality* $\tau(B,R) \leq 2$ *holds.*

We first consider a similar problem in the plane lattice, and obtain a similar result, as shown in the following Theorem 1.2. For a set S of points in the plane lattice, the *rectangular hull* of S, denoted by $rect(S)$, is the smallest closed rectilinear rectangular enclosing S, each of whose edges is a vertical or horizontal line segment ((2) in Fig. 1). In particular, every edge of $rect(S)$ contains at least one point of S.

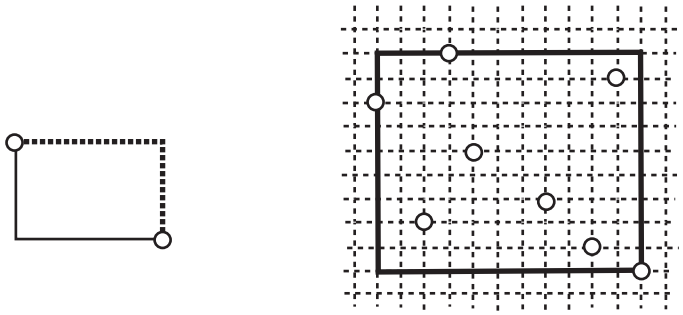


FIGURE 1. (1) Two L-line segments joining two points; and (2) A rectangular hull of a set of points in the plane lattice in general position.

For a set X of points in the plane lattice in general position, a spanning tree on X each of whose edges is an L-line segment connecting two points of X is called a *spanning tree on* X *with* L*-line segments*. A non-crossing spanning tree on X with L-line segments and with maximum degree at most 3 is denoted by $L\text{-}tree^3(X)$.

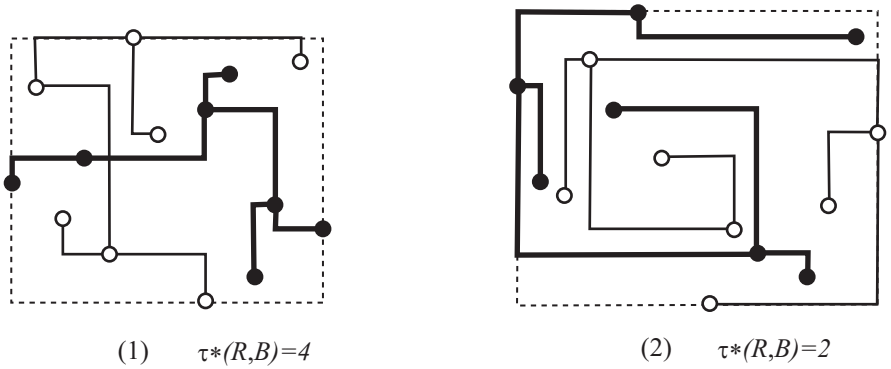


FIGURE 2. Two spanning trees $L\text{-}tree^3(R)$ and $L\text{-}tree_3(B)$ with minimum number of crossings, where R and B are disjoint sets of red and blue points in the plane lattice in general position.

Theorem 1.2. *Let R and B be two disjoint sets of red points and blue points in the plane lattice, respectively, such that $R \cup B$ is in general position. Let $\tau^*(R, B)$ denote the number of unordered pairs $\{x, y\}$ of points of $R \cup B$ such that one of $\{x, y\}$ is red and the other is blue, and x and y are on the consecutive edges of $rect(R \cup B)$. Then $\tau^*(R, B)$ is even and $0 \leq \tau^*(R, B) \leq 4$, and there exist two non-crossing spanning trees L-tree$^3(R)$ and L-tree$^3(B)$ on R and B respectively such that the crossing number in L-tree$^3(R) \cup$ L-tree$^3(B)$ is equal to 0 if $\tau(R, B) \leq 2$, and 1 otherwise (see Fig. 2).*

Theorem 1.2 can be proved by using Lemma 1.3 below and by considering the situation given in Fig. 4. However we omit their proofs.

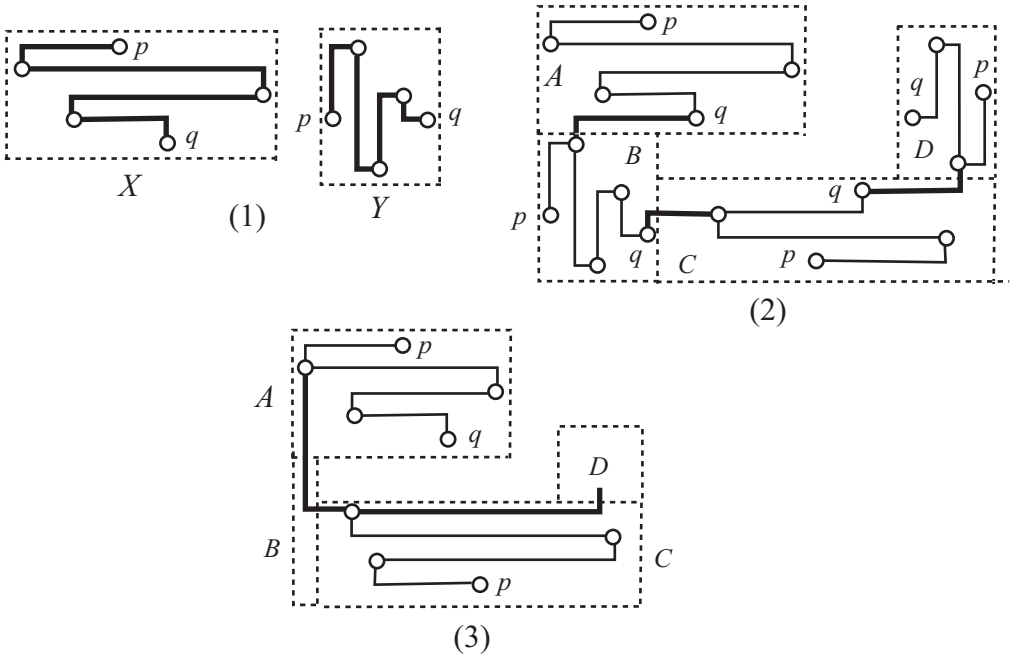


FIGURE 3. A spanning tee T on P with L-line segments and maximum degree at most three given in Lemma 1.3.

Lemma 1.3. *Let X_i, $1 \leq i \leq 4$, be disjoint rectangles in the plane such that X_i and X_{i+1} have a boundary edge in common for every $1 \leq i \leq 3$, as shown in Fig. 3, where $A = X_1$, $B = X_2$, $C = X_3$ and $D = X_4$. Let P be a set of points in the plane lattice in general position contained in $X_1 \cup X_2 \cup X_3 \cup X_4$. Then there exists a non-crossing spanning tree T on P with L-line segments and maximum degree at most three such that the leftmost point of D has degree at most two in T.*

2 Embedding trees in the plane lattice

We next consider the following conjecture and present some partial results on it. Let T be a tree and P a set of $|T|$ points in the plane lattice in general position, where $|T|$ denotes the order of T. If T can be drawn on P without crossing such that each edge of T is an L-line segment connecting two points of P, then we say that T can be drawn on P with L-line segments (see Fig. 5).

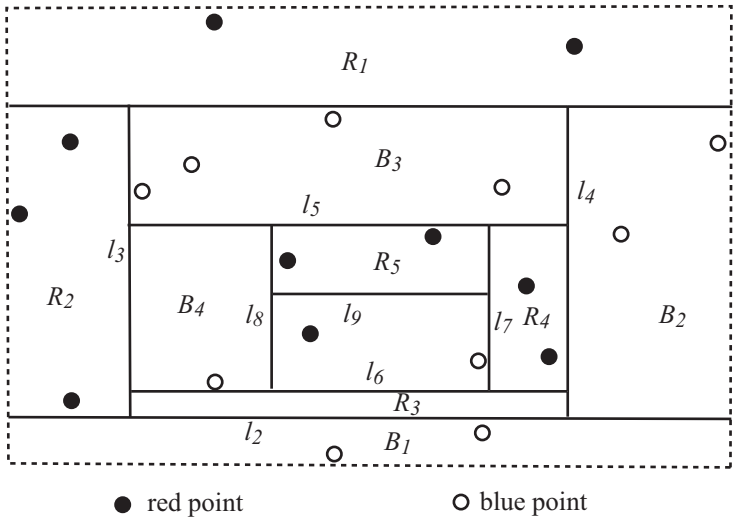


FIGURE 4. Horizontal line segments and vertical line segments.

Conjecture 2.1. *Let T be a tree with maximum degree* 3*, and let P a set of $|T|$ points in the plane lattice in general position. Then T can be drawn on P with L-line segments without crossing (see Fig.* 5*).*

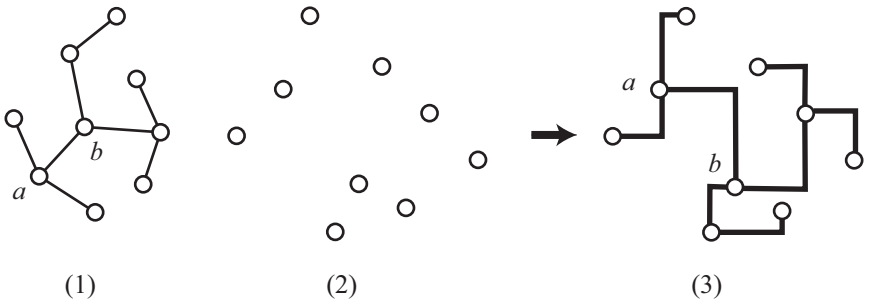


FIGURE 5. (1) A tree T with maximum degree 3; (2) A set P of $|T|$ points in the plane lattice in general position; (3) T is drawn on P with L-line segments without crossing.

References

[1] A. Kaneko and M. Kano, Discrete geometry on red and blue points in the plane – A survey, *Discrete and Computational Geometry, Algorithms and Combinatorics*, **25**, Springer (2003), 551–570.

[2] M. Kano and K. Suzuki, Discrete geometry on red and blue points in the plane lattice, preprint.

[3] S. Tokunaga, Intersection number of two connected geometric graphs. *Inform. Process. Lett.* **59** (1996), 331–333.

A note on diagonal transformations on maximal planar graphs containing perfect matchings

Ferran Hurtado[1], **Marc Noy**[1], **Eduardo Rivera-Campo**[2]

[1] Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya
ferran.hurtado@upc.edu, marc.noy@upc.edu

[2] Departamento de Matemáticas, Universidad Autónoma Metropolitana – Iztapalapa
erc@xanum.uam.mx

Abstract. Let G and F be maximal planar graphs that contain perfect matchings. If G and F have the same number of vertices, then there is a sequence $H_0, H_1, \ldots, H_t$ of maximal planar graphs, also containing perfect matchings, with the property that G is isomorphic to H_0, F is isomorphic to H_t and, for $i = 0, 1, \ldots, t-1$, H_{i+1} is obtained from H_i by a diagonal transformation.

Introduction

Let xuv and yuv be two adjacent faces of a maximal planar graph G such that xy is not an edge of G. A *diagonal transformation* or *diagonal edge flip* of G consists of deleting the edge uv and adding the edge xy. Wagner [**4**] proved that any two maximal planar graphs G and F with the same number of vertices are equivalent under diagonal transformations: There is a sequence $H_0, H_1, \ldots, H_t$ of maximal planar graphs such that G is isomorphic to H_0, F is isomorphic to H_t and, for $i = 0, 1, \ldots, t-1$, H_{i+1} is obtained from H_i by a diagonal transformation.

Let G be a maximal planar graph having a perfect matching. As shown in Fig. 1, a maximal planar graph $(G-uv)+xy$, obtained from G by a diagonal transformation, may contain no perfect matchings. Here we show that if both graphs G and F admit perfect matchings, then the graphs $H_0, H_1, \ldots, H_t$ can be chosen within the set of maximal planar graphs having perfect matchings.

In the proof of our main result, we use a geometric version of diagonal transformations. Let P be a set of points in general position in the plane. A *geometric triangulation* of P is a set of triangles with vertices in P and pairwise disjoint interiors such that their union is the convex hull $CH(P)$ of P and no triangle in T contains a point of P in its interior.

If xuv and yuv are adjacent triangles of a triangulation T of P such that $xuyv$ is a convex quadrilateral, then a new triangulation $(T-uv)+xy$ of P is obtained by deleting the edge uv and adding the edge xy. This operation is often referred to as a *geometric edge flip*.

The *graph of triangulations* $T(P)$ of P is an abstract graph whose vertices are the triangulations of P in which T_1 and T_2 are adjacent if one is obtained from the other by performing a geometric edge flip. Lawson [**3**] proved that $T(P)$ is always connected, and Houle *et al.* [**2**] proved that the subgraph $T_M(P)$ of $T(P)$, induced by the set of triangulations of P that contain perfect matchings, is also connected.

[1]Partially supported by projects MICINN MTM2009-07242 and Gen. Cat. DGR 2009SGR1040.
[2]Partially supported by Conacyt, México 83856.

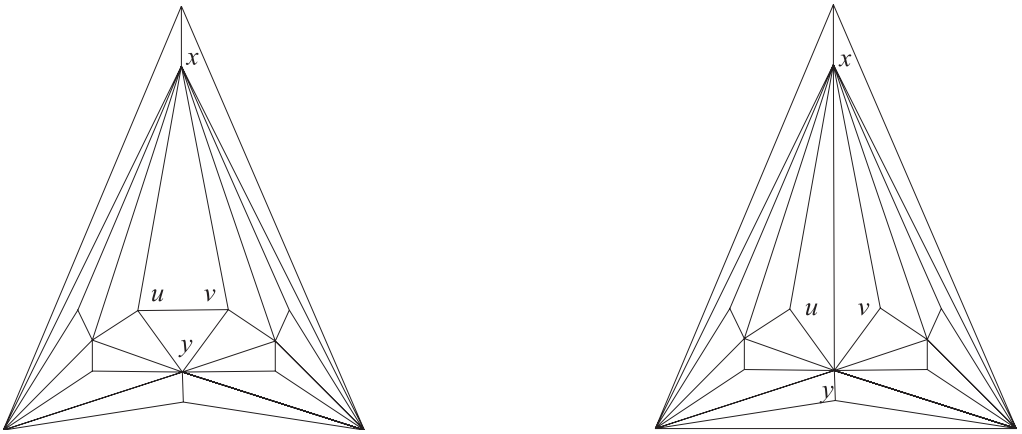


FIGURE 1. The graph G on the left contains a perfect matching, while the graph $(G - uv) + xy$ on the right does not

1 Preliminary results

In his proof, Wagner showed that any maximal planar graph G is equivalent under diagonal transformations to the graph $\triangle_n$ shown in Fig. 2. The following technical variation will be used in the proof of our main result.

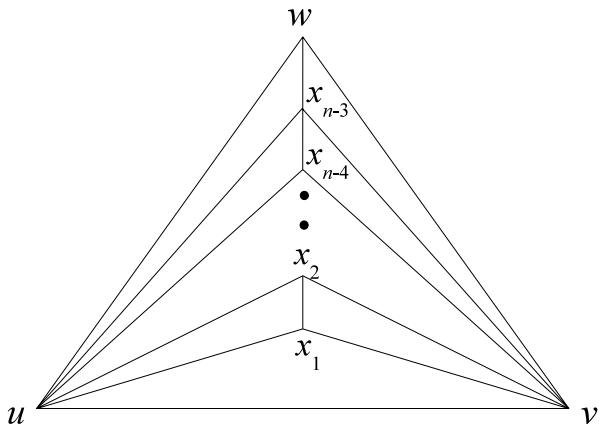


FIGURE 2. Graph $\triangle_n$

Lemma 1.1. *Let G be a maximal plane graph with n vertices. Denote by u, v and w the vertices in the outerface of G and by $x_1, x_2, \ldots, x_{n-3}$ the remaining vertices of G. If G contains the edges $ux_1, vx_1, x_1x_2, x_2x_3, \ldots, x_{n-4}x_{n-3}$ and $x_{n-3}w$, then there is a sequence $G = G_0, G_1, \ldots, G_t$ of maximal plane graphs, each with outerface uvw and containing all the edges $ux_1, vx_1, x_1x_2, x_2x_3, \ldots, x_{n-4}x_{n-3}$ and $x_{n-3}w$, such that G_t is isomorphic to $\triangle_n$ and, for $i = 0, 1, \ldots, t-1$, G_{i+1} is obtained from G_i by a diagonal transformation.*

Proof. Let $E_u = \{ux_i : i = 1, 2, \ldots, n-3\}$, $E_v = \{vx_i : i = 1, 2, \ldots, n-3\}$ and $k(G) = |(E_u \cup E_v) \backslash E(G)|$. If $k(G) = 0$, then $E_u \cup E_v \subset E(G)$ and G is isomorphic

to $\triangle_n$. We proceed by induction assuming $k(G) \geq 1$ and that the result holds for any maximal plane graph G' with $V(G') = V(G)$, with outerface uvw, containing all the edges $ux_1, vx_1, x_1x_2, x_2x_3, \ldots, x_{n-4}x_{n-3}, x_{n-3}w$ and such that $k(G') < k(G)$.

Without loss of generality, we also assume that $E_u \backslash E(G) \neq \emptyset$. Let $x_{n-2} = w$. Since ux_1 and ux_{n-2} are edges of G, there are integers k and m with $k+1 < m$ such that ux_k and ux_m are edges of G but $ux_{k+1}, ux_{k+2}, \ldots, ux_{m-1}$ are not edges of G. Because G is a maximal planar graph and $x_kx_{k+1}, x_{k+1}x_{k+2}, \ldots, x_{m-1}x_m$ are edges of G, x_k, u, x_m must be a face of G. Let $x_l \in V(G)$ be such that x_k, x_l, x_m is the other face of G incident with the edge x_kx_m (see Fig. 3). Notice that $k < l < m$ since G is planar and contains all the edges $x_kx_{k+1}, x_{k+1}x_{k+2}, \ldots, x_{m-1}x_m$. Therefore $ux_l \notin E(G)$.

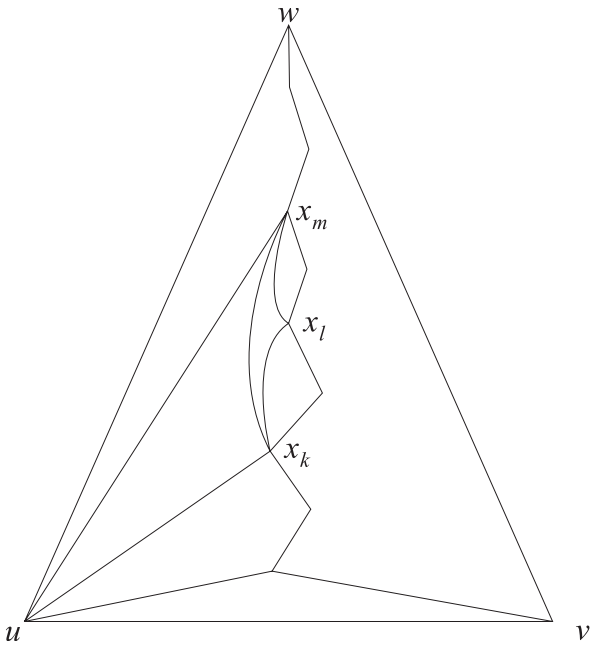


FIGURE 3. x_k, u, x_m and x_k, x_l, x_m are the faces of G incident with the edge x_kx_m

The graph $G' = (G - x_kx_m) + ux_l$ is obtained from G by a diagonal flip and is such that $k(G') = k(G) - 1$. The result follows by induction. □

For any geometric triangulation T of P, we denote by $G(T)$ the *skeleton graph* of T. That is, the abstract graph whose vertices and edges are the points in P and the edges in T, respectively.

Remark 1.2. Let T_1 and T_2 be geometric triangulations of P. If T_2 is obtained from T_1 by a geometric edge flip, then $G(T_2)$ is obtained from $G(T_1)$ by a diagonal transformation.

2 Main result

For any positive even integer n, let S_n denote the set of all maximal planar graphs with n vertices that admit a perfect matching.

Theorem 2.1. *Let $n \geq 4$ be an even integer. For every graph $G \in S_n$, there is a sequence $H_0, H_1, \ldots, H_t$ of graphs in S_n such that G is isomorphic to H_0, $\triangle_n$ is isomorphic to H_t and, for $i = 0, 1, \ldots, t-1$, H_{i+1} is obtained from H_i by a diagonal transformation.*

Proof. Let G be a maximal planar graph with a perfect matching. By a well known result of Fary [**1**], there is an embedding H of G in the plane where all edges are straight line segments. Clearly G is the skeleton graph of a geometric triangulation T of the point set $P = V(H)$; without loss of generality we assume that P is in general position.

Let u, v and w be the vertices in the outerface of H and denote by l_{uv} the line containing u and v. Let $x_1, x_2, \ldots, x_{n-3}$ be the remaining vertices of H, labelled according to their distance to the line l_{uv}.

Let T' be any geometric triangulation of P containing all the edges ux_1, vx_1, x_1x_2, $x_2x_3, \ldots, x_{n-4}x_{n-3}$ and $x_{n-3}w$. Notice that T' also contains a perfect matching. Since the graph $T_M(P)$ is connected, there is a sequence $T = T_0, T_1, \ldots, T_s = T'$ of geometric triangulations of P having perfect matchings such that for $i = 0, 1, \ldots, s-1$, T_{i+1} is obtained from T_i by performing a geometric edge flip. It follows from Remark 1.2 that, for $i = 0, 1, \ldots, s-1$, $G(T_{i+1})$ is obtained from $G(T_i)$ by a diagonal transformation.

By Lemma 1.1, there is a sequence $G(T_s) = G_s, G_{s+1}, \ldots, G_t$ of maximal plane graphs, each containing the edges $ux_1, vx_1, x_1x_2, x_2x_3, \ldots, x_{n-4}x_{n-3}, x_{n-3}w$ (and therefore a perfect matching), such that G_t is isomorphic to $\triangle_n$ and, for $i = 0, 1, \ldots, t-1$, G_{s+i+1} is obtained from G_{s+i} by a diagonal transformation. The theorem holds since G is isomorphic to $H = G(T)$. □

References

[1] I. Fary. On straight lines representation of planar graphs, *Acta Univ. Szeged. Sect. Sci. Math.* **11** (1948), 229–233.

[2] M. E. Houle, F. Hurtado, M. Noy, E. Rivera-Campo, Graphs of triangulations and perfect matchings, *Graphs Combin.* **21** (2005), 325–331.

[3] C. L. Lawson, Software for C^1-interpolation, in *Mathematical Software III* (J. Rice, ed.), Academic Press, New York, 1977, pp. 161–194.

[4] K. Wagner, Bemerkungen zum Vierfarbenproblem, *Jber. Deutsch. Math.-Verein.* **46** (1936), 26–32.

Polytopal complexes realizing products of graphs

António Guedes de Oliveira[1], Edward D. Kim[2], Marc Noy[3], Arnau Padrol[3], Julian Pfeifle[3], Vincent Pilaud[4]

[1] Universidade do Porto
agoliv@fc.up.pt

[2] Technische Universiteit Delft
e.d.h.kim@tudelft.nl

[3] Universitat Politècnica de Catalunya
marc.noy@upc.edu, arnau.padrol@upc.edu, julian.pfeifle@upc.edu

[4] Université Paris 7
vincent.pilaud@liafa.jussieu.fr

Abstract. When is the Cartesian product of two graphs the graph of a polytope, of a cellular sphere, or even of a combinatorial manifold? In this note, we determine all 3-polytopal complexes whose graph is the Cartesian product of a 3-cycle by a Petersen graph. Through this specific example, we showcase certain techniques which seem relevant to enumerate all polytopal complexes realizing a given product.

In this note, we investigate the question of finding polytopes (or more generally polytopal complexes) with a prescribed graph. This harks back to Steinitz's Theorem [**1**], which characterizes the graphs of 3-polytopes as the 3-connected planar graphs, and thus ensures that 3-polytopality is polynomially decidable. In contrast, Richter-Gebert proved that deciding 4-polytopality is NP-hard, as a consequence of his work on realization spaces of 4-polytopes [**2**]. Motivated by this computational threshold, we focus on deciding 4-polytopality for the subclass of Cartesian products of graphs.

Polytopality of Cartesian products of graphs was initially studied in [**3**]. By definition, polytopality is preserved by taking products: the graph of a product of polytopes is the product of their graphs. We are interested in the reciprocal question: can a product of non-polytopal graphs be polytopal? The answer differs significantly according to whether we require the realizing polytope to be simple or not [**3**, Theo. 2.2 and Prop. 2.7]:

(1) A Cartesian product of regular graphs is the graph of a simple polytope if and only if its factors are.
(2) There exist (non-simple) polytopal products of non-polytopal regular graphs.

This note studies the 4-polytopality of the product of a cycle by a (small) non-polytopal 3-regular graph, for which the above-mentioned results do not apply. Focusing on the product $K_3 \times \mathrm{Pet}$ of a 3-cycle by a Petersen graph, we illustrate several useful techniques to understand 4-polytopality of Cartesian products in general. Our approach consists in enumerating all 3-polytopal complexes whose graph is $K_3 \times \mathrm{Pet}$, and requires two steps: we first compute all possible facets (3-dimensional faces) of all possible 3-polytopal complexes realizing $K_3 \times \mathrm{Pet}$, and we then study all possible ways to glue these facets along ridges (2-dimensional faces) to form a complex with the desired graph.

[2]Supported by Netherlands Organization for Scientific Research (NWO) Vidi grant 639.032.917.
[3]Supported by MEC grants MTM2008-03020, MTM2009-07242, and AGAUR grant 2009 SGR 1040.
[4]Supported by MEC grant MTM2008-04699-C03-02.

Definitions

Cartesian products of graphs. The *(Cartesian) product* of two graphs G and H is the graph $G \times H$ whose vertex set is $V(G \times H) := V(G) \times V(H)$ and with edge set $E(G \times H) := (V(G) \times E(H)) \cup (E(G) \times V(H))$. We call *$G$-edges* (or *blue edges*) the edges of $G \times H$ that are products of an edge of G by a vertex of H, and *H-edges* (or *red edges*) the ones obtained as the product of a vertex of G by an edge of H. See Fig. 1(b).

Polytopal complexes. We consider a *d-polytopal complex* to be a closed regular combinatorial d-dimensional manifold, each of whose *faces* is (combinatorially) a convex polytope. In other words, a d-polytopal complex can be seen as a set of d-polytopes together with a combinatorial (*i.e.*, not geometric) gluing rule of their facets such that:

(1) the link of any k-face is a combinatorial $(d-k-1)$-sphere, and
(2) any pair of faces meets in exactly one lower-dimensional (possibly empty) face.

We say that a d-polytopal complex realizes a graph G if its 1-skeleton is isomorphic to G.

Example 1 (A 4-polytopal complex realizing $K_3 \times \mathrm{Pet}$)**.** Let $\triangle$ denote a 3-cycle, seen as a cellular decomposition of the circle $\mathbb{S}^1$. Let $\mathcal{C}$ denote the cellular decomposition induced by the embedding of the Petersen graph Pet in the projective plane $\mathbb{P}$ (see Figure 1(c)). Then the Cartesian product $\triangle \times \mathcal{C}$ induces a cellular decomposition of $\mathbb{S}^1 \times \mathbb{P}$ consisting of six cycles of three pentagonal prisms, whose graph is $K_3 \times \mathrm{Pet}$. In this note, we enumerate all other 3-polytopal complexes realizing $K_3 \times \mathrm{Pet}$.

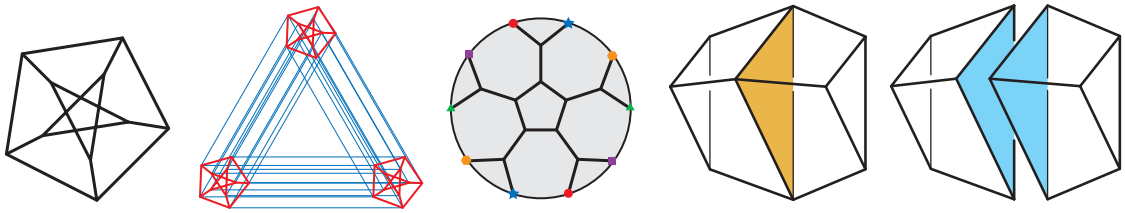


FIGURE 1. The Petersen graph (a), its product with a 3-cycle (b), and its embedding into the projective plane (c). The refinement of a missing triangle (d).

Refinement. When looking for a 3-polytopal complex $\mathcal{K}$ realizing a graph, we can always assume that no facet of $\mathcal{K}$ contains a *missing triangle*, that is, a 3-cycle that does not bound a triangular 2-face. Otherwise, we can always refine any facet with a missing triangle T into two polytopal cells that share the triangle T. See Figure 1(d). Note that the complex $\mathcal{K}$ could still miss a triangle whose edges belong to three distinct facets of $\mathcal{K}$.

Finding all possible facets

Our first step to enumerate all possible 3-polytopal complexes $\mathcal{K}$ realizing $K_3 \times \mathrm{Pet}$ is to understand the possible candidates for the facets of $\mathcal{K}$. As mentioned earlier, we can assume that the complex $\mathcal{K}$ is refinement minimal: no facet contains a missing triangle. Thus, the facet candidates are given by the induced 3-connected planar subgraphs of $K_3 \times \mathrm{Pet}$ with no separating 3-cycle. Assisted by a computer enumeration program, we obtain the graphs of the triangular, pentagonal and hexagonal prisms (as expected), plus the graphs of the four polytopes presented in Figure 2.

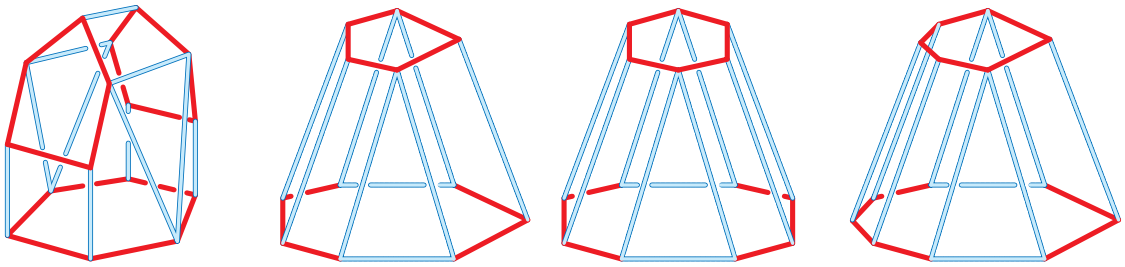


FIGURE 2. The four non-simple facet candidates for a realization of $K_3 \times \text{Pet}$.

It turns out that these seven polytopes all have red degree at most 2. We use this property to discard the four candidates of Figure 2, by considering the vertex figures of $\mathcal{K}$. Remember that the *vertex figure* of a vertex v in a d-polytopal complex $\mathcal{C}$ is the $(d-1)$-sphere bounding the faces of $\mathcal{C}$ incident to v: it has a k-cell for each $(k+1)$-face of $\mathcal{C}$ incident to v. In our situation, each vertex figure bounds a 3-polytope with 5 vertices, which can only be an Egyptian pyramid or a bipyramid over a triangle. Furthermore, the blue and red colors on the edges of the product $K_3 \times \text{Pet}$ induce a coloring of the vertices of our vertex figures. Figure 3 depicts all possible configurations.

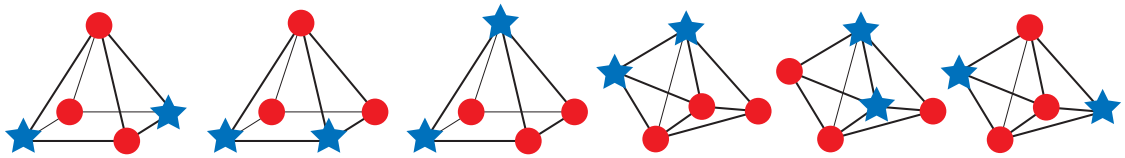


FIGURE 3. All 3-polytopes with 2 blue vertices (stars) and 3 red vertices (circles).

In fact, not all these configurations can appear as vertex figures of $\mathcal{K}$:

- The first one has two opposite blue vertices on its square base. This would imply that the facet of $\mathcal{K}$ corresponding to this square has a missing 2-face delimited by two blue edges, which is forbidden by refinement minimality of $\mathcal{K}$.
- The next three have 2-faces with three red vertices, which is impossible since all facet candidates for $\mathcal{K}$ have red degree at most 2.

Thus, only the last two configurations of Figure 3 can appear as vertex figures of $\mathcal{K}$, and all the facets of $\mathcal{K}$ are simple. This discards the candidates of Figure 2:

Proposition 2. *The only facet candidates for a refinement minimal* 3*-polytopal complex realizing* $K_3 \times \text{Pet}$ *are the triangular, pentagonal and hexagonal prisms.*

Gluing facets together

In this second step, we study the possible ways of gluing the facet candidates for $\mathcal{K}$. Since only triangular (resp. pentagonal, resp. hexagonal) prisms contain triangles (resp. pentagons, resp. hexagons), we get the following structural result:

Lemma 3. *The triangular prisms in* $\mathcal{K}$ *form disjoint cycles of triangular prisms, obtained as products of the* 3*-cycle* K_3 *by disjoint cycles of* Pet*. Any pentagonal (resp. hexagonal) prism of* $\mathcal{K}$ *belongs to a cycle of three pentagonal (resp. hexagonal) prisms, obtained as the product of the* 3*-cycle* K_3 *by a* 5*-cycle (resp.* 6*-cycle) of* Pet*.*

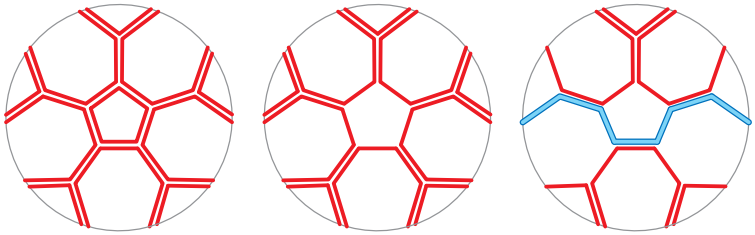


FIGURE 4. The projection representation of three possible 3-dimensional refinement minimal polytopal complexes whose graph is $K_3 \times \mathrm{Pet}$.

Projecting the facets of $\mathcal{K}$ on the Petersen graph, we obtain a family of 5-cycles (pentagonal prisms of $\mathcal{K}$), a family of 6-cycles (hexagonal prisms of $\mathcal{K}$), and a family of isolated edges (triangular prisms of $\mathcal{K}$) such that:

(1) each edge of Pet is covered precisely twice by the elements of these families;
(2) any two elements of these families intersect in at most one edge;
(3) the family of edges form disjoint cycles in Pet.

We have represented three possibilities in Figure 4. The leftmost one has six 5-cycles and represents the 3-polytopal complex constructed in Example 1. The middle one has five 5-cycles and five isolated edges (omitted in the picture). The rightmost one has three 5-cycles, one 6-cycle (blue in the picture) and nine isolated edges (omitted again).

Proposition 4. *These three complexes are the only refinement minimal* 3*-polytopal complexes realizing* $K_3 \times \mathrm{Pet}$*. None of them is a combinatorial sphere (even less the boundary complex of a* 4*-polytope).*

Remark 5. From our description of refinement minimal complexes realizing $K_3 \times \mathrm{Pet}$, we can now obtain all complexes realizing $K_3 \times \mathrm{Pet}$ by coarsening along triangular 2-faces. The realizations of Figure 4 give rise to one, four, and seven non-isomorphic realizations, respectively. Thus, we obtain in total twelve non-isomorphic realizations.

Conclusion

The approach discussed in this note enables us to enumerate all 3-polytopal complexes realizing certain Cartesian products of graphs. As another example, we can prove in a similar way that there is a unique 3-polytopal complex realizing $K_3 \times K_{3,3}$. Nevertheless, our method only applies to small products (less than 50 vertices, say) for obvious complexity reasons. It leaves open Ziegler's question [**4**] regarding the polytopality of the Cartesian product of two Petersen graphs, which initially motivated this research.

References

[1] E. Steinitz. Polyeder und Raumenteilungen. In *Encyclopädie der mathematischen Wissenschaften, Band 3 (Geometrie), Teil 3AB12*, 1922, 1–139.

[2] J. Richter-Gebert. *Realization spaces of polytopes*, Volume 1643 of *Lecture Notes in Mathematics*, Springer-Verlag, Berlin, 1996.

[3] J. Pfeifle, V. Pilaud & F. Santos. Polytopality and Cartesian products of graphs. Accepted in *Israel Journal of Mathematics*, 2011, `arXiv:1009.1499`.

[4] G. M. Ziegler. Convex polytopes: Examples and conjectures. In Marc Noy and Julian Pfeifle, editors, *DocCourse Combinatorics and Geometry 2009, Part I: Intensive Courses*, volume 5.1 of *CRM Documents*, Centre de Recerca Matemàtica, Barcelona, 2010, pages 9–49.

Object recognition using Delaunay triangulation

Russel Apu, Marina Gavrilova[1]

[1] CPSC Department, University of Calgary
marina@cpsc.ucalgary.ca

Abstract. We present a novel method based on Delaunay triangulation for shape recognition. The method is able to represent invariant arcs in a topological context and use that representation for general sign recognition. We utilize Delaunay triangulation for optimal weight assignment and apply maximum bipartite algorithm for shape matching. Extensive runtime analysis indicates that the proposed Delaunay triangulation based method had a significantly better detection rate for the test images than other current methods.

Introduction

Image understanding is a classical problem which originated several decades ago [**5**]. Application areas that have been extensively studied are military target recognition, image recognition, object class recognition, image processing and enhancement, face recognition, biometric recognition, Augmented Reality (AR), and motion capture. One defining characteristic of the Visual Sign Recognition problem is the presence of features that are virtually identical if scale and/or rotation are ignored. This property makes the task of shape matching difficult. Such inherent indistinctiveness cannot be properly addressed by a local matching algorithm that considers individual features. The relative locations of these features are important to perform recognition. Furthermore, the distinctiveness of features does not depend on the feature itself but rather on the class of objects being examined. Therefore, a group of training objects must be observed and learned to understand which features are more reliable. Thus, algorithms with simple correspondence techniques that solely rely on features are not sufficient to properly identify geometrically similar objects from a pool of similar shapes (such as traffic signs).

Literature review

Considerable progress has been made in object recognition domain using feature based classification [**6, 7**]. However, these methods often fail to detect objects manifested by geometric shapes that are often found in signs and landmarks [**7**]. Recent studies confirm that most state-of-the-art computer vision techniques struggle to detect geometric properties in objects, resulting in confusion and misclassification [**7, 8**]. Some authors proposed to discard rotation invariance and consider feature directionality to match the object. In such approaches, the template is rotated at regular intervals and inserted into the database [**4, 7, 8**]. The matching is performed against a single rotated (and/or scaled) instance of a template. The problem, however, is that this method only works when the shape contains enough distinctive features relative to other objects in the database. Also, the accuracy of the method is dependent on the frequency of the rotation

[2]Partially supported by NSERC Discovery grant.

interval. To address the above problems, we propose an original method based on Delaunay triangulation (DT), which considers topological relationships among objects. DT has been proven highly beneficial in many applied problems pertaining to space decomposition, biometric processing, information systems, navigation, clustering, and pattern analysis [**3, 10**].

Methodology based on Delaunay triangulation

To establish the uniqueness of an object class, we present an original method based on topological relationships to incorporate the relative positions of the geometric features. At first, we considered to establish a topology through a boundary-arc relational graph or clustering from the transitive closure of the relations graph and applying original Maximum Flow Bipartite Matching algorithm [**10**]. However, there were many issues with methods being computationally expensive and not practical. To provide an optimal topological representation of features, we propose training and matching algorithm to identify objects based on Delaunay triangulation. Each vertex representing a matched 2D object feature, called CART feature (introduced by the authors in [**1**]), has three directional vectors that define the orientation and characteristics of the edge, with DT constructed on these feature vectors as sites. Next, each graph-edge in the DT represents an augmented feature vector which is constructed using the edge and the CART feature vectors (Fig. 1). The entries are the two simpletones of the CART feature vectors that are joined by the DT edge, the angles of the directional vectors (three for each CART feature), and the length of the DT edge. Then, the characteristics vector of each CART feature is added, containing information such as sharpness, skew, scale, region influence and rotation histogram. The weights are assigned in a consistent manner which is dependent on the type of feature being compared. For Simpletone matching, three weights are needed. Two weights are for the matching priority of the RGB colors, and the third weight is for the dot product of the 2-Simpletone (rod-like patterns in color subspace) directions. In order to compute the distance between two feature vectors, we use the sum of weighted distances of all sub-components. Most distance computations are fairly simple, except the Simpletone distances. These are the "S-tone 1" and "S-tone 2" components of the feature vector. The distance between two 2-simpletones (or 2.5-simpletones) is computed as the area of the minimal surface (or the maximum tension surface) that contains both Simpletones. A good way to approximate this surface is by triangulating the four endpoints of the Simpletone [**2**]. There are four such triangulation possible. The triangulation with the minimum total surface area is chosen as the distance. The area of a triangle in three dimensional space is computed using the cross product of the two edge vectors. When computing distances between clusters in 2.5-simpletones, the closest distance between P_i and Q_j is chosen as the edge length.

For training, we assume that we have a number of instances of the object that we call "the templates". Each of these templates contains a single instance of the object which occupies most of the area. A large number of templates should be collected for training in such a way that the background is randomized and the instances cover all aspects of small variations in perspective, lighting, etc. We devised a method based on Delaunay triangulation to adjust the weight matrix in such a way that the impact of the background and the unstable clutter of feature on the matching cost is attenuated. The weight is then assigned to be the inverse of the computed cumulative cost. After the training process is completed, we have a set of templates each consisting of a Delaunay

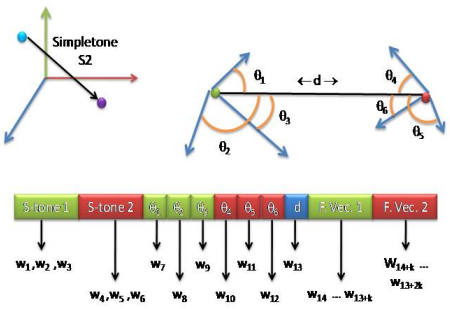


FIGURE 1. The feature vector for a Delaunay edge consists of elements from the vertices (green and red represents the two endpoints) and the edge (blue).

triangulation and feature vectors for all vertices and edges. In addition, each DT consists of a weight matrix that maps each edge to an array of weights $w_k(e, k)$, where e belongs to $DT(t)$ for the given template t and the subcomponent index k for subcomponent S_k. For matching, we consider a test image against a given template. The test image is processed using the Simpletone analysis, followed by CART and the feature vector extraction. The template is rescaled in the range [0..1] with respect to the test image at regulag interval. For higher efficiency, we consider a correspondence matrix and examine scale at several levels. Initially, we consider a few evenly distributed scales and pick the best match. In the next level we reduce the range and further refine the scale (i.e., in our case 32 different scases, in 3 levels). For each scale, we find a matching cost is computed using the Maximum Bipartite Matching Algorithm [**2**] (Figure 3). One crucial optimization is to prune Delaunay edges using the connectivity graph and randomization.

Experimentation. We implemented the proposed method and performed exhaustive experimental studies on performance of the object recognition system.

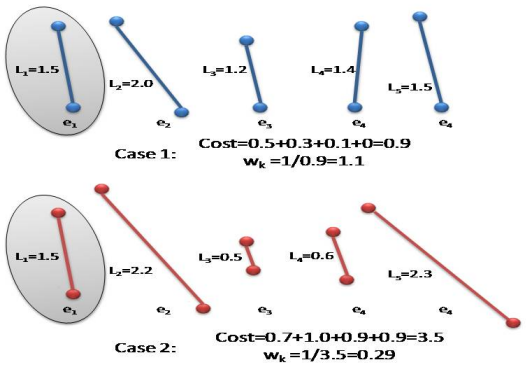


FIGURE 2. The two cases describe the response of the training algorithm for two sets of training templates for a feature L.

First, training is carried out by providing positive and negative examples of the object class we are attempting to detect. After this, object recognition system takes as

input a test image and decides whether the image contains an instance of the object class. We analyzed the detection characteristic of the proposed DT based method and compare it with the result obtained from Haar Cascade [**9**] by creating a training database of 128 stop signs cropped from natural images. In trial with 20 signs and non signs, the detection rate at 80% had a 65% FAR, while our DT CART method showed much better performance. The runtime profile indicates that our method outperforms the Haar Cascade for the given set of test images. At 90% detection rate, the Haar method had a 28.67% error rate. In comparison, the false alarm rate for our CART based DT method was only 5.33% at 90% detection rate. The training time for the Haar method was very high (five day training). The training time for our method was 2 hours. The method achieves near real-time performance with an average 6.73 seconds per frame speed or 9 frames per minute.

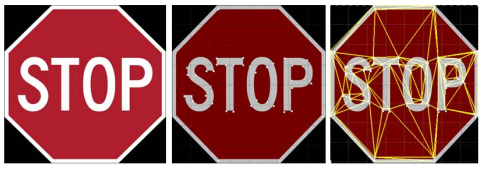


FIGURE 3. Example of a template and corresponding DT.

Conclusion

We presented a novel method based on DT for shape recognition. The method is able to represent invariant arcs in a topological context and use that representation for general sign recognition. Delaunay triangulation was utilized for optimal weight assignment and then maximum bipartite algorithm ws applied for shape matching. Extensive runtime analysis indicates that the proposed DT-based method has a significantly better detection rate for the test images than popular methods such as Haar transform.

References

[1] R. A. Apu and M. L. Gavrilova, Shape matching through contour extraction using Circular Augmented Rotational Trajectory (CART) algorithm, *IJBIDM Journal*, **4, 2** (2010), 192–210.

[2] R. A. Apu and M. L. Gavrilova, Flexible tracking of object contours using LR-traversing algorithm, IGIV 2006, IEEE-CS (2006), 503–513.

[3] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, A. Lounsbery, and W. Stuetzle, Multiresolution analysis of arbitrary meshes, ACM SIGGRAPH (1995), 173–182.

[4] R. Fergus, P. Perona, and A. Zisserman, Object class recognition by unsupervised scale-invariant learning, IEEE-CS, CVPR (2003), II-264 – II-271.

[5] M. T. Fischler and R. A. Elschlager, The representation and matching of pictorial structures, *IEEE Transactions on Computers*, **C-22, 1** (1973), 67–92.

[6] P. E. Forssen and D. G. Lowe, Shape descriptors for maximally stable extremal regions, ICCV'07 (2007), 1–8.

[7] D. G. Lowe. Distinctive image features from scale-invariant keypoints, *International Journal of Computer Vision*, **60, 2** (2004), 91–110.

[8] A. Vedaldi, G. Guidi, and S. Soatto, Relaxed matching kernels for object recognition, IEEE Conference on Computer Vision and Pattern Recognition (2008), 1–8.

[9] P. A. Viola and M. D. Jones, Rapid object detection using a boosted cascade of simple features, CVPR **1** (2001), 511–518.

[10] C. Wang and M. L. Gavrilova, Delaunay triangulation algorithm for fingerprint matching, 3rd International Symposium on Voronoi Diagrams in Science and Engineering (2006), 208–216.

Parallel Delaunay triangulation based on Lawson's incremental insertion

Narcís Coll[1], **Marité Guerrieri**[1]

[1] Departament d'Informàtica i Matemàtica Aplicada, Universitat de Girona
coll@ima.udg.edu, mariteg@ima.udg.edu

Abstract. In this paper we present an ongoing research work on Delaunay triangulations using GPU-based algorithms. The proposed algorithm is based on Lawson's incremental insertion, taking special care to avoid concurrent insertion of points in triangles and conflicts between edge swaps.

Introduction

Graphics Processing Units (GPUs) are specialized processors which use a highly parallel structure that makes them perfect for solving problems that can be partitioned into independent and smaller parts. The development of CUDA (Compute Unified Device Architecture) and some programming languages such as OpenCL (Open Computing Language), makes GPUs attractive to solve problems in a parallel way.

Delaunay triangulation is one of the fundamental topics in Computational Geometry and it is used in many areas such as terrain modelling, finite element methods, pattern recognition, computer graphics, data interpolation, robotics, etc. Although the 2D Delaunay triangulation can be computed in $O(n \log n)$ time, where n is the number of input points, it still consumes a lot of time, especially for current applications that often need to work with millions of points. In such cases, a parallel algorithm is necessary to accelerate its computation.

In Lawson's incremental algorithm, when a point is inserted, the triangle that contains it is found, and three new edges are inserted to attach the new vertex to the vertices of the containing triangle. Next, a recursive procedure tests whether the new vertex lies within the circumcircles of any neighbouring triangles. Each affirmative test triggers an edge flip that removes a locally not Delaunay edge. Each edge flip reveals two additional edges that must be tested. Thus, two points lying in the same triangle cannot be inserted at the same time, and two points lying in different triangles can not be processed independently if they share neighbours. In this paper we propose a GPU-based algorithm in OpenCL language that avoids concurrent insertion of points in triangles and conflicts between edge swaps.

1 Related work

Several methods **[3, 4, 7, 9]** for parallel Delaunay triangulation use the divide and conquer strategy to construct partial Delaunay triangulations in subregions, and finally merge these partial triangulations to get the result. However, this kind of algorithms have the following drawbacks. First, the merge phase is quite complex because it involves not only building the faces connecting the subregions but also correcting of existing faces in the

[1]Partially supported by research grant TIN2010-20590-C02-02.

subregions to satisfy the Delaunay criterion. In the worst case, these corrections affect the whole triangulation. Second, the merge phase is done by just one processor and thus it limits the efficiency of the algorithm.

Parallel 2D Delaunay triangulation algorithms [**1, 2, 5, 6**] avoiding the divide and conquer strategy are inspired on Bowyer–Watson's algorithm that is based on incremental insertion with cavity retriangulation. Firstly, these algorithms create a coarse triangulation of a subset of points. Secondly, the triangles are partitioned into areas and assigned to processors and the parallel insertion begins. Synchronizations across area boundaries are necessary.

In [**8**], a randomized insertion is used as a base for the parallel algorithm. The triangulation is structured in a directed acyclic graph (DAG) that stores the history of its changes. This algorithm consists of three phases: location, subdivision and legalization where the circumcircle criterion is applied and, if necessary, the local part of the triangulation is modified. All three phases of the algorithm need to access the DAG structure. A node of the DAG can be accessed simultaneously by several threads. When any of these threads needs to modify it, synchronizations among the threads are implemented to avoid artifacts in the resulting triangulation.

Other methods [**10, 11**] are based on computing by GPU a discrete Voronoi diagram of the points, store it in a texture, and use it to derive the Delaunay triangulation. The drawback of these methods is that the accuracy of the triangulation strongly depends on the resolution of the texture.

2 Our approach

In CUDA, the parallelizable parts of an algorithm are executed by a collection of threads grouped into blocks of user defined size running in parallel. The code to be executed by each thread is written in a kernel where different types of memory can be used: registers (local memory of a thread), global (accessible to thread) and shared (accessible by every thread of a block). Atomic operations are used to operate on a memory position without allowing any other access to that memory position during the process.

After the initialization phase, we follow an iterative process that finishes when all input points are inserted into the Delaunay triangulation. In each iteration we insert as many points as possible with the condition that only one point can be inserted into one single triangle. Each iteration is divided into three steps: location, where the triangle containing every non inserted point is determined; insertion, where at most one point is inserted in a triangle; and swapping, where non Delaunay edges are swapped avoiding conflicts between them.

2.1 Initialization phase

In order to use as efficiently as possible the GPU's resources the following data structures are used. Let n be the number of vertices. `Vertices` is an array of size $n+3$ where each element contains a position (x, y) in 2D. Its first three positions corresponds to the three vertices of an auxiliary large triangle that contains all points. `Triangles` is a $2n+1$ sized array of indices to `Vertices` where each three consecutive indices correspond to a triangle. Position zero of this array corresponds to the auxiliary triangle. The array `Neighbours` contains indices to neighbors and future neighbors triangles of each triangle. Each six consecutive indices are related to a triangle. The first three correspond to

neighbors of the triangle and the last three to future neighbors of the triangle before executing a swapping. Other arrays storing results of intermediate steps are needed to facilitate the general process. For each vertex, the `Inserted` array contains a flag to know whether the vertex has been inserted or not, and the `ContainingTriangle` array indicates which triangle contains the vertex. Initially, all the vertices are contained in the auxiliary triangle. For each triangle, the `VertexToInsert` and the `EdgeToSwap` arrays respectively record the vertex to be insert in the triangle and the edge of the triangle to be swapped. All arrays are allocated in the global memory.

2.2 Location step

This step updates `ContainingTriangle` and `VertexToInsert` by the use of a kernel. Each thread operates on a vertex of index i. If vertex $v = \texttt{Vertices}[i]$ is not inserted yet into the triangulation ($\texttt{Inserted}[i] = 0$) and it is not contained into its assigned triangle $t = \texttt{ContainingTriangle}[i]$, a walking process is launched along direction cv, where c is the centroid of t, until the triangle t' containing v is reached. If v lies on an edge, it is assigned to the triangle with less index incident to the edge. Then, $\texttt{ContainingTriangle}[i]$ is updated with t' and $\texttt{VertexToInsert}[t']$ is updated with v by the use of an atomic operation. In this manner, $\texttt{VertexToInsert}[t']$ contains the first vertex arriving to t'.

2.3 Insertion step

This step inserts the vertices stored in `VertexToInsert` into the triangulation by the use of three kernels. Each thread of the kernels operates on a vertex of index i. Let triangle $t = \texttt{ContainingTriangle}[i]$. If $i = \texttt{VertexToInsert}[t]$, triangle t will be subdivided in three triangles of indices t, $2i + 1$ and $2i + 2$. The first kernel checks, for each neighbour t' of t, if neigbour t of t' will be $2i + 1$ or $2i + 2$ after the subdivision of t. In that case, this information is stored in the future neighbours part of $\texttt{Neighbours}[t']$. The second kernel effectively inserts i in t by using its future neighbours.

2.4 Swapping step

This step swaps edges and is separated in three kernels that operate on a triangle of index t. The first kernel selects, if there exists, an edge of t to be swapped and stores it in `EdgeToSwap`. An edge is candidate to be swapped if it does not fulfill the Delaunay criterion and it is the only one edge of the adjacent neighbour triangle candidate to be swapped. Then three cases arise: (1) Only one edge can be swapped —then it is selected to be swapped if t is lower than the adjacent neighbour; (2) two edges are candidates to be swapped —then one of them is chosen for swapping; and (3) three edges are candidates to be swapped —then again one of them is chosen for swapping. If an edge has been selected, let t' be the neighbouring triangle of t sharing the edge. Then, all triangles t'' adjacent to the quadrilateral determined by t and t' are updated in the following way. If after the edge swapping, neighbour t of t'' will be changed by t' or viceversa, this information is stored in the future neighbours part of $\texttt{Neighbours}[t'']$. The second kernel effectively swaps the selected edge, while the third kernel updates the neighbours with the information stored in future neighbours. These three kernels are executed sequentially until no edge can be selected.

3 Results

The algorithm has been executed ten times on each of five different sets of random points. Table 1 shows the mean running times (the input set of vertices is previously loaded in memory) and mean number of iterations for computing the Delaunay triangulations. These results have been carried out on a computer equipped with an Intel(R) Pentium(R) D CPU 3.00GHz, 3.5GB RAM and a GPU NVidia GeForce GTX 580/PCI/SSE2 which has a cached global memory, reducing the access to global memory problems and time.

n	25600	52100	256000	521000	1000000
Mean time (s)	0.088	0.146	0.665	1.283	2.373
Mean iterations	18	19	21	23	24

TABLE 1. Behaviour of the proposed algorithm.

As it has been pointed out in the abstract, in this paper we present an ongoing research. Future work will consist in studying the behaviour of our approach on different point distributions and comparing its performance with the current parallel implementations.

References

[1] C. D. Antonopoulos, F. Blagojevic, A. N. Chernikov, N. P. Chrisochoides and D. S. Nikolopoulos, Algorithm, software, and hardware optimizations for Delaunay mesh generation on simultaneous multithreaded architectures. *J. Parallel Distrib. Comput.* **69** (2009), 601–612.

[2] C. D. Antonopoulos, F. Blagojevic, A. N. Chernikov, N. P. Chrisochoides and D. S. Nikolopoulos, A multigrain Delaunay mesh generation method for multicore smt-based architectures. *J. Parallel Distrib. Comput.* **69** (2009), 589–600.

[3] G. E. Blelloch, G. L. Miller, J. C. Hardwick and D. Talmor, Design and implementation of a practical parallel Delaunay algorithm. *Algorithmica* **24** (1999), 243–269.

[4] M.-B. Chen, T.-R. Chuang and J.-J. Wu, Parallel divide-and-conquer scheme for 2D Delaunay triangulation: Research articles. *Concurr. Comput. Pract. Exper.* **18** (2006), 1595–1612.

[5] N. Chrisochoides and D. Nave, Parallel Delaunay mesh generation kernel. *International Journal for Numerical Methods in Engineering* **58** (2003), 161–176.

[6] N. Chrisochoides and F. Sukup, Task parallel implementation of the Bowyer–Watson algorithm, in: *Proceedings of Fifth International Conference on Numerical Grid Generation in Computational Fluid Dynamics and Related Fields* (1996), 773–782.

[7] P. Cignoni, C. Montani, R. Perego and R. Scopigno, Parallel 3D Delaunay triangulation. *Computer Graphics Forum* **12** (1993), 129–142.

[8] J. Kohout, I. Kolingerová and J. Zára, Parallel Delaunay triangulation in E^2 and E^3 for computers with shared memory. *Parallel Computing* **31** (2005), 491–522.

[9] S. Lee, C.-I. Park and C.-M. Park, An improved parallel algorithm for Delaunay triangulation on distributed memory parallel computers, in: *Proceedings of the 1997 Advances in Parallel and Distributed Computing Conference (APDC'97)*, APDC'97, IEEE Computer Society, 131–138.

[10] G. Rong, T.-S. Tan, T.-T. Cao and Stephanus, Computing two-dimensional Delaunay triangulation using graphics hardware, in: *Proceedings of the 2008 symposium on Interactive 3D graphics and games*, I3D'08, ACM, 89–97.

[11] J. Valdés, Cálculo de la triangulación de Delaunay en la GPU, in: *Actas Encuentros de Geometría Computacional* (2009), 125–129.

Connecting red cells in a bichromatic Voronoi diagram

Manuel Abellanas[1,2], Antonio L. Bajuelos[3,4], Santiago Canales[5,2], Mercè Claverol[6,7], Gregorio Hernández[1,2], Inês Matos[3,6,8]

[1] Departamento de Matemática Aplicada, Universidad Politécnica de Madrid, Spain
{mabellanas,gregorio}@fi.upm.es

[3] Departamento de Matemática & CIDMA, Universidade de Aveiro, Portugal
{leslie,ipmatos}@ua.pt

[5] Escuela Técnica Superior de Ingeniería, Universidad Pontificia Comillas de Madrid, Spain
scanales@dmc.icai.upcomillas.es

[6] Departaments de Matemàtica Aplicada II & IV, Universitat Politècnica de Catalunya, Spain
merce@ma4.upc.edu

Abstract. Let S be a set of $n+m$ sites, of which n are red and have weight w_R, and m are blue and weigh w_B. The objective of this paper is to calculate the minimum value of w_R such that the union of the red Voronoi cells in the weighted Voronoi diagram of S is a connected set. The problem is solved for the multiplicatively-weighted Voronoi diagram in $\mathcal{O}((n+m)^2 \log(nm))$ time and for the additively-weighted Voronoi diagram in $\mathcal{O}(nm \log(nm))$ time.

Introduction

Suppose that $S = R \cup B$ is a set of $n+m$ sites, n of which are red and m of which are blue; and let $\mathrm{VD}(S)$ denote the ordinary Voronoi diagram of S. A Voronoi cell of $\mathrm{VD}(S)$ is said to be red (resp. blue) if the corresponding generator site is red (blue). The goal of this paper is to connect the red cells in order to allow one to travel within red regions following paths that do not cross blue regions. If this is not possible for $\mathrm{VD}(S)$, then there are several options to make this happen; for instance, one can add red sites or delete some blue sites, or even move sites. The approach chosen in the following is to assign different weights to red and blue sites and therefore consider their weighted Voronoi diagram (using multiplicatively- and additively-weighted distances) [**5**]. All red sites are assigned the same weight w_R, as all blue sites are assigned w_B. Consequently, the main goal is to calculate the values of w_R and w_B for which the union of red cells is connected. As it is easy to realise, in these conditions the only relevant data is the relationship between w_R and w_B. Therefore, and assuming w_B constant, the problem can be restated as calculating the minimum weight w_R^* of the red sites that connects all red cells under the appropriate diagram.

Let $\mathrm{VD}^w(S)$ denote the weighted Voronoi diagram of S. Bear in mind that $\mathrm{VD}^w(S) = \mathrm{VD}(R)$ when the weight of R tends to infinity, which assures the existence of a solution to our problem. It is clear that the structure of $\mathrm{VD}^w(S)$ depends on the weight of R

[2]Partially supported by Projects MTM2008-05043 and HP2008-0060.
[4]Supported by FCT through CIDMA of University of Aveiro.
[7]Partially supported by Projects MTM2009-07242 and Gen. Cat. DGR 2009SGR1040.
[8]Supported by FCT through CIDMA of University of Aveiro and grant SFRH/BPD/66572/2009.

and if w_R is small enough then $\mathrm{VD}^w(S) = \mathrm{VD}(B)$. Starting from this case, increasing w_R will expand the red cells and eventually two of them will meet and form one red connected component of $\mathrm{VD}^w(S)$. The point where these two cells meet is called a *critical point*. As red cells keep expanding, more will connect at different weights and each of these weights will be defined by a sole critical point. The sought weight is the one that finally unites the last two disconnected red components of $\mathrm{VD}^w(S)$. Consequently, finding these critical points is the key to solve the problem. The remainder of this paper is divided in two sections that correspond to the two types of weighted distances in question: multiplicatively- and additively-weighted distance, respectively.

1 Multiplicatively-weighted distance

The weighted distance used in this section is called the *multiplicatively-weighted distance*. Given a point p on the plane and $s_i \in S$, its distance is defined by $d_M(p, s_i) = \frac{1}{w} d_E(p, s_i)$, where d_E is the Euclidean distance and w should be replaced by the current weight of R if $s_i \in R$ or w_B if $s_i \in B$. The multiplicatively-weighted distance characterises the multiplicatively-weighted Voronoi diagram of S [**2**]. As previously mentioned, critical points are the key to calculate w_R^*, and in order to find them we need to understand how red cells form clusters on this diagram. To this end, the following definition characterises the events where the red cells of $\mathrm{VD}^w(S)$ meet. Let $b(s_i, s_j) = \{p \in \mathbb{R}^2 : d_M(p, s_i) = d_M(p, s_j)\}$ denote the bisector between sites s_i and s_j.

Definition 1.1. If w_R is the exact weight of R when two red cells of $\mathrm{VD}^w(S)$ meet for the first time at point c, then c is a *critical point of type I* if there exist red sites r_i and r_j and blue sites b_k and b_l such that $\{c\} = b(r_i, b_k) \cap b(r_i, b_l) \cap b(r_j, b_k) \cap b(r_j, b_l)$. Otherwise c is a *type II critical point:* if $w_R < w_B$ and c belongs to the blue cell of $\mathrm{VD}^w(S)$ defined by b_k then $\{c\} = b(r_i, b_k) \cap b(r_j, b_k)$; if $w_R > w_B$ and c belongs to the red cell of $\mathrm{VD}^w(S)$ defined by r_i then $\{c\} = b(r_i, b_k) \cap b(r_i, b_l)$.

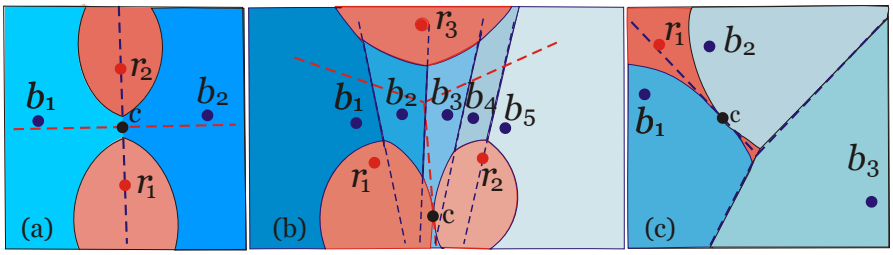


FIGURE 1. $\mathrm{VD}(R)$ is shown in a dashed red trace and $\mathrm{VD}(B)$ in dark blue. (a) Point c is a type I critical point. (b) Point c is a type II critical point for $w_R < w_B$. (c) Point c is a type II critical point for $w_R > w_B$.

Figure 1(a) shows clearly that a type I critical can also be found as an intersection point between $\mathrm{VD}(R)$ and $\mathrm{VD}(B)$. As the weight of R increases, the red cells defined by r_1 and r_2 will meet at c and form one red connected component of $\mathrm{VD}^w(S)$. Figures 1(b) and 1(c) illustrate two examples of critical points of type II. Figure 1(c) also shows that the regions of the multiplicatively-weighted Voronoi diagram may be disconnected. Therefore, a critical point is also created when a region meets itself (in this case, point c at the intersection of the two red cells defined by r_1).

Proposition 1.2. *Red connected components of* $\mathrm{VD}^w(S)$ *can only meet at critical points of type I or II.*

Proposition 1.3. *Red connected components of* $\mathrm{VD}^w(S)$ *can only meet* $\mathcal{O}(n+m)$ *times.*

A method to find candidates to critical points follows directly from Definition 1.1. As previously noted, candidates to type I critical points are easily found since they exist on the intersections of the edges of $\mathrm{VD}(R)$ with $\mathrm{VD}(B)$. If $\{c\} = b(r_i, r_j) \cap b(b_k, b_l)$, then the weight w_R needed to reach c with red sites is given by

$$d_M(c, r_i) = w_B \frac{d_E(c, r_i)}{d_E(c, b_k)}.$$

Candidates to type II critical points depend on the relationship between w_R and w_B to be found on edges of $\mathrm{VD}(R)$ or on edges of $\mathrm{VD}(B)$. However, the procedure to find them is similar and so only the first case will be described: candidates to critical points on edges of $\mathrm{VD}(R)$ that fall in the interior of cells of $\mathrm{VD}(B)$. Since an edge of $\mathrm{VD}(R)$ may cross several cells of $\mathrm{VD}(B)$ (see Figure 1(b)), one decides which is the blue site responsible for c by computing all the intersection points between such edge of $\mathrm{VD}(R)$, $b(r_i, r_j)$, and $\mathrm{VD}(B)$. For each intersection point p_k that corresponds to a blue site b_k, calculate $d_M(p_k, r_i) - d_M(p_k, b_k)$. This distance will be zero at c and therefore studying how this value alternates along $b(r_i, r_j)$ is the solution to find the cell of $\mathrm{VD}(B)$ where c is. Once blue site b_k is found, one can work out the functions that describe $b(r_i, b_k)$ and $b(r_j, b_k)$, both depending on the unknown weight of R. Finally, c is found at the minimum weight of R for which these bisectors are tangent.

Proposition 1.4. *There are* $\mathcal{O}(nm)$ *candidates to critical points.*

This proposition shows that there is a gap between the actual number of critical points and the number of candidates to critical points. To conclude, a binary search will then locate w_R^* amongst these candidates, as stated in the following theorem.

Theorem 1.5. *Concerning the multiplicatively-weighted distance, weight* w_R^* *can be found in* $\mathcal{O}((n+m)^2 \log(nm))$ *time.*

Proof. The first task to find candidates to critical points is to construct and intersect $\mathrm{VD}(B)$ and $\mathrm{VD}(R)$, which takes $\mathcal{O}(nm \log(n+m))$ time [**3**]. According to Proposition 1.4, there are $\mathcal{O}(nm)$ of these candidates that correspond to $\mathcal{O}(nm)$ different weights. Afterwards, sort these weights into a list in ascending order, which takes $\mathcal{O}(nm \log(nm))$ time. A binary search will locate w_R^* on this list: for each listed weight, construct $\mathrm{VD}^w(S)$ in $\mathcal{O}((n+m)^2)$ time [**2**]. Using this diagram, build a graph G that has a node for each red cell and two nodes are connected if the respective cells are neighbours. To verify if G is connected, traverse it using the Depth-First Search algorithm that runs in $\mathcal{O}((n+m)^2)$ time [**4**]. If G is indeed connected, then the search proceeds to lower weights, otherwise it proceeds to higher weights. Finally, this step takes $\mathcal{O}((n+m)^2)$ time for each weight and so it is concluded in $\mathcal{O}((n+m)^2 \log(nm))$ time. □

2 Additively-weighted distance

The weighted distance studied in this section is called the *additively-weighted distance* and is defined by $d_A(p, s_i) = d_E(p, s_i) - w$. Such distance characterises the additively-weighted Voronoi diagram of S [**1**]. The method to find critical points on this diagram is

similar to the one used in the last section —the main difference here is that the regions of the additively-weighted Voronoi diagram are always connected, if they exist. Let $b(s_i, s_j) = \{p \in \mathbb{R}^2 : d_A(p, s_i) = d_A(p, s_j)\}$ represent the bisector between sites s_i and s_j.

Definition 2.1. If w_R is the exact weight of R when two red cells of $\mathrm{VD}^w(S)$ meet for the first time at point c, then c is a *critical point of type I* if there exist red sites r_i and r_j and blue sites b_k and b_l such that $\{c\} = b(r_i, b_k) \cap b(r_i, b_l) \cap b(r_j, b_k) \cap b(r_j, b_l)$. Otherwise c is a *type II critical point* if $\{c\} = \overrightarrow{b_i r_i} \cap b(r_i, r_j)$, where $\overrightarrow{b_i r_i}$ is a ray from b_i to r_i.

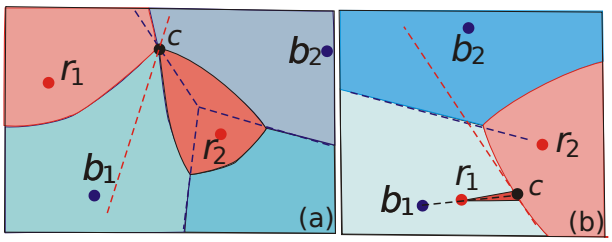


FIGURE 2. $\mathrm{VD}(R)$ is shown in a dashed red trace and $\mathrm{VD}(B)$ in dark blue. Point c is a critical point of type I in (a) and of type II in (b).

As before, type I critical points are found on the intersections of $\mathrm{VD}(R)$ and $\mathrm{VD}(B)$ (see Figure 2(a)). A type II critical point $\{c\} = \overrightarrow{b_1 r_1} \cap b(r_1, r_2)$ is shown in Figure 2(b).

Proposition 2.2. *Red connected components of* $\mathrm{VD}^w(S)$ *can only meet at critical points of type I or II, and they will meet* $\mathcal{O}(n)$ *times at most.*

Proposition 2.3. *There are* $\mathcal{O}(nm)$ *candidates to critical points.*

Once again, there is a gap between the actual number of critical points and the number of candidates to critical points. As soon as the list of these candidates is found, w_R^* can be located by means of a binary search. Since this is the same method as used in Theorem 1.5, the proof of the ensuing result is omitted.

Theorem 2.4. *Concerning the additively-weighted distance, the weight* w_R^* *can be found in* $\mathcal{O}(nm \log(nm))$ *time.*

Observe that it is possible for a type II critical point to exist "in the infinity" and this is true for both weighted distances. Nonetheless, this solution is not interesting to our problem as a path visiting all red cells would be infinitely long. Therefore, we assume the existence of a bounding box or of a polygon containing all the sites of S. In both cases, this type of critical point can be found as the intersection between the respective bisector (between the red sites) and such bounding box.

References

[1] F. Aurenhammer, Voronoi diagrams – a survey, *Institute for Information Processing Technical University of Graz* Report **263** (1988).

[2] F. Aurenhammer and H. Edelsbrunner, An optimal algorithm for constructing the weighted Voronoi diagram in the plane, *Pattern Recognition* **17** (1984), 251–257.

[3] F. Aurenhammer and R. Klein, Voronoi diagrams, in *Handbook of Computational Geometry*, J.-R. Sack and J. Urrutia, eds., North-Holland, Amsterdam, 2000, 201–290.

[4] S. W. Golomb and L. D. Baumert, Backtrack programming, *Journal of ACM* **12,4** (1965), 516–524.

[5] A. Okabe, B. Boots, K. Sugihara and S. N. Chiu, *Spatial Tessellations – Concepts and Applications of Voronoi Diagrams*, 2nd ed., John Wiley, 2000.

Sensor recalibration with Voronoi diagrams

Belén Palop[1]

[1] Departamento de Informática, Universidad de Valladolid, Spain
b.palop@infor.uva.es

Abstract. In this paper we propose the use of Voronoi diagrams to automatically recalibrate sensors. Every sensor will compute some estimate of its own value that depends on the information measured by its Delaunay neighbors using Sibson's interpolation. Whenever this value is too different from its own measures, the sensor should recalibrate and assume the estimate to be its real measurement. We have implemented a software simulating the behaviour of this recalibration process. Our experiments show that, for perturbations higher than a 30% of their original values, a 99.62% of the sensors are recalibrated even for very degradated sensor networks. The total number of comunication rounds averages 6.33, having 98.87% of the experiments required 9 rounds or less. On the other hand, for perturbations ranging from 10 to 100%, we wrongly recalibrate on average a 20.0% of the sensors that had correct values, but the average translation of the wrong recalibrations is nearly one half of the correct ones.

Introduction

Wireless sensor networks can provide dense monitoring of environments. Whenever factors like humidity, temperature, light or others have to be strictly controlled, sensor networks happen to be one of the best solutions [**1**]. In these situations, redundant information is usually captured in order to successfully detect misfunctions in the sensors. A dense deployment of the sensors is therefore needed due to the error-prone nature of these little devices. In fact, environments such as nuclear plants or concrete factories among others have strict regulations on how often sensors have to be recalibrated in order to ensure the accuracy of the measured values. This calibration process is usually performed manually with a high cost on human resources. Moreover, the need to certify the correct behaviour of the sensors in the meantime increases their manufacturing cost.

In this paper we present a second step (see the first in [**3**]) towards the study of the usefulness of Sibson's interpolation [**4**] in sensor networks, and evaluate its power in order to automatically detect and correct outliers in a densely deployed wireless sensor network (see [**2**] for a survey on other outlier detection methodologies). For each sensor, we compare its measured value with a weighted mean of the values measured by its Delaunay neighbors. Whenever these two values differ significantly, the sensor is recalibrated and the interpolated value is assumed to be the real measure.

1 Sibson's interpolation for a sensor network

Given a point p and a set S of sites s_i in the plane with associated values z_i, the value at point p is computed as a weighted mean as follows. Let $s_1, \ldots, s_k$ be the Delaunay neighbors of p when we insert it in the Voronoi $VD(S)$ diagram of S. The weight of neighbor s_j for $j \in \{1, k\}$ is the proportion of the area of the Voronoi region of p in $VD(S \cup p)$ that intersects the Voronoi region of s_j in $VD(S)$; see Figure 1.

[1]Partially supported by VA094A08, HP-2008-0060 and MTM2008-05043/MTM.

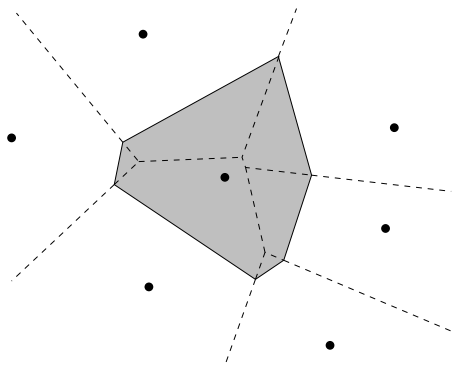


FIGURE 1. Sibson's interpolation assigns weights to the Delaunay neighbors according to the proportion of their area in the shaded Voronoi region.

In **[3]** we explored the effect of different synchronization schemes for this technique since circular references arise in the process. We simulated both synchronous and asynchronous processes. In the first, recalibrations happen after communication rounds, while in the latter, values are updated independently. We concluded that the synchronous process needs less rounds to reach stability and shows better correction levels. Another interesting result therein is that whether the simulated input shape was a circle or a square did not significantly affect the measured values. The results in the present work are thus performed with a synchronous scheme for simulated disc-shaped sensor networks.

2 Experimental results

The sensor network has been simulated following a uniform distribution on a disc. We have set the values of the sensors according to a normal distribution centered at the origin (see Figure 2). We have changed the values of some of the sensors by some proportion and marked which of the sensors were recalibrated afterwards. We call the proportion of corrected values the *success ratio*. On the other hand, the *non-correct ratio* is the proportion of the sensors that have been wrongly recalibrated.

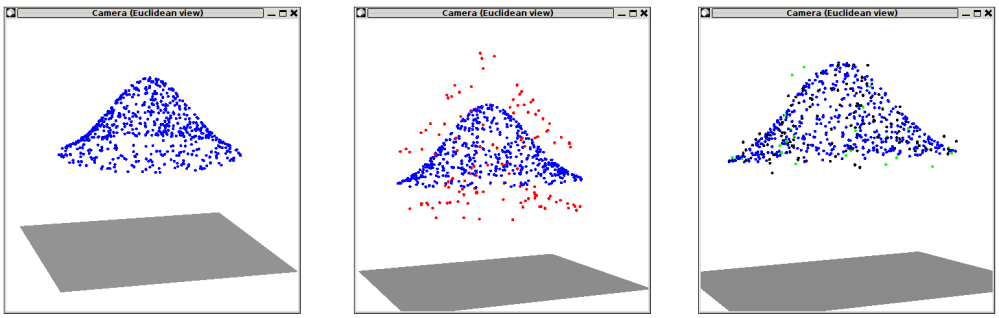


FIGURE 2. Input sensor network and assigned data (left). 3D view of a degradated network with a 20% of faulty sensors that need recalibration (center). 3D view of a recalibrated network (right).

We have performed 1500 simulations using a latin hypercube scheme, where the following four parameters have been studied:

- Size: The number of sensors is a value between 100 and 1000.
- Resistance: A sensor accepts its neighbour's value when it differs from its measured value from 0.005 to 0.05.
- Errors: The proportion of sensors with wrong values varies between 10% and 50%.
- Strength: The measures of the sensors with errors are between 10% and 100% of their original values.

Since wrongly calibrated sensors are chosen randomly, for each combination of these four parameters, 5 runs have been performed.

Our experiments show that the most significant parameter is strength, being the other three of little to none effect. Figure 3 shows the performance ratios and the translation for the measured values with respect to the strength. We can see that, for small perturbations (a strength value of less than 30%), the success ratio sometimes does not even reach 50%, but the miscalculated recalibrations are also very low. On the other hand, for values bigger than 30%, 99.62% of the wrongly working sensors are correctly detected and recalibrated.

With respect to the average translation of the measured values, we can say that, as expected, the higher the value of the strength parameter, the bigger the translation performed. Nevertheless, the translation of the successful recalibrations for those high values nearly doubles the one of the wrongly recalibrated sensors.

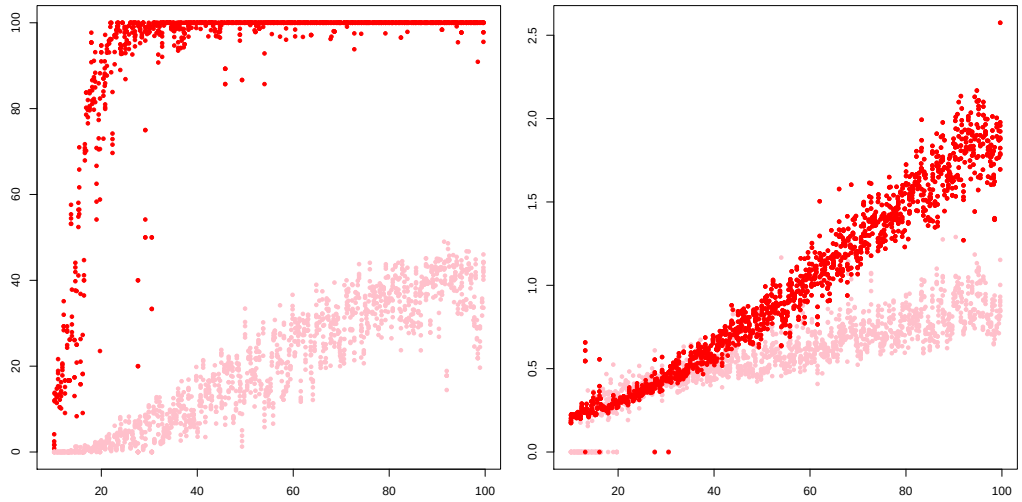


FIGURE 3. On the left, success ratio (dark) and non-correct calibrations ratio (light). On the right, average translation of correct calibrations (dark) and non-correct ones (light).

The number of communication rounds is less sensitive to the strength but tends to increase both with high strength values and bigger input sizes. Anyway, the average number of rounds is as low as 6.33, and 98.87% of the experiments have required 9 rounds or less. Figure 4 shows the histogram of the number of comunication rounds for all experiments performed.

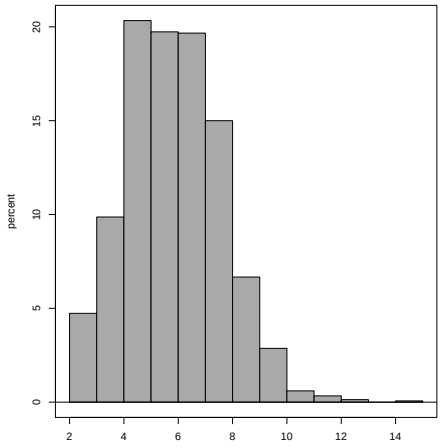


FIGURE 4. Histogram of the number of comunication rounds.

3 Conclusions

We have implemented a software that simulates a recalibration process based on Sibson's interpolation. Among the four parameters used in order to launch the experiments using Latin hypercube sampling, the variation on the original values (strength) is the most important one. In fact, all measured values have a high dependence on this parameter and it affects from the success ratio to the number of comunication rounds. On the other hand, the threshold value that marks when a sensor has to be recalibrated (resistance) seemed to be of importance a priori, but the experimental results show that its effect is negligible. Similarly, the amount of wrong data (errors) was not important, at least with the reasonable assumption that it had to be less than 50%.

This is an ongoing work and we still seem to have more questions than answers. In particular, it would be very useful to address the same problem in a non-random setting, where sensors do not simply misfunction, but are attacked in a strategical way. Two interesting approaches appear: how to defend a sensor network from such an attack, and how to attack such a sensor network.

Acknowledgements

The author wants to acknowledge Raúl Santos for the implementation of the first version of the simulator, and Joaquín Aguilar, Diego R. Llanos and David Orden for the fruitful discussions.

References

[1] I. F. Akyildiz, Weilian Su, Y. Sankarasubramaniam, E. Cayirci, A survey on sensor networks, *IEEE Communications Magazine* **40(8)** (2002), 102–114.

[2] V. J. Hodge, J. Austin, A survey of outlier detection methodologies, *Artificial Intelligence Review* **22 (2)** (2004), 85–126.

[3] R. Santos, Estrategias de contagio de información en una red de sensores, Master's Thesis, Universidad de Valladolid, 2010.

[4] R. Sibson, A vector identity for the Dirichlet tessellation, *Mathematical Proceedings of the Cambridge Philosophical Society* **87** (1980), 151–155.

Spiral serpentine polygonization of a planar point set

Justin Iwerks[1], **Joseph S. B. Mitchell**[1]

[1] Department of Applied Mathematics and Statistics, Stony Brook University, Stony Brook, NY 11794-3600, USA
jiwerks@ams.sunysb.edu, jsbm@ams.sunysb.edu

Abstract. We introduce a simple algorithm for constructing a spiral serpentine polygonization of a set S of $n \geq 3$ points in the plane. Our algorithm simultaneously gives a triangulation of the constructed polygon at no extra cost, runs in $O(n \log n)$ time and uses $O(n)$ space.

Introduction

A polygonization of a planar point set S is a simple polygon having S as the set of its vertices. Different types of polygonizations have been investigated in settings where objects are being constructed from limited data, such as pattern recognition and image reconstruction **[3, 5, 6]**. The number of polygonizations for a given point set can be exponential in n, even when restricted to monotone or star-shaped polygonizations **[7]**.

Agarwal et al. have discussed the attractiveness of the subset of polygonizations that admit *thin* triangulations, which minimize the number of nodes of degree three in the dual, and in particular, *serpentine* triangulations, whose dual graph is a path. In **[2]**, the authors gave an $O(n \log n)$ algorithm for computing a serpentine polygonization of a point set S.

We show that any point set S has a *spiral* serpentine polygonization. A spiral serpentine polygon is a simple polygon possessing at most one chain of reflex vertices and exactly one chain of convex vertices (see Figure 1) and admitting a serpentine triangulation. We present a simple algorithm in Section 1 for constructing such a polygonization in $O(n \log n)$ time, requiring $O(n)$ space and explicitly giving a serpentine triangulation at no extra cost. In Section 2, a series of claims are presented (many without proof due to space constraints) establishing the correctness of the algorithm.

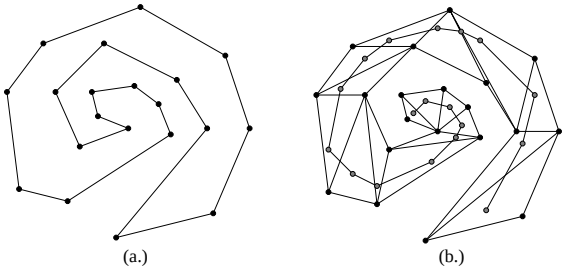


FIGURE 1. A spiral polygon is shown in (a.) and a serpentine triangulation of this polygon is shown in (b.), where the dual of the triangulation is the path depicted.

[1]This work is partially supported by the National Science Foundation (CCF-1018388).

1 The algorithm

Here we introduce the algorithm `SpiralSerpentinePolygonize`, which produces a spiral serpentine polygonization of a planar set S of $n \geq 3$ points. During the first step of the algorithm's execution, we compute the *convex layers* of all points in S. The points of a convex layer L have *depth* d_L, the number of times one would have to iteratively remove the convex hull of S until the points of L are removed [**1, 4**]. After identifying the rightmost point with minimum y-coordinate, the algorithm processes one unvisited point of S at a time. Each point becomes a vertex of a new triangle that is attached to a visible edge of the triangulation constructed so far. The spiral and serpentine invariants are maintained throughout as we discuss in Section 2.

Algorithm: `SpiralSerpentinePolygonize`(S)

1.) Compute the convex layers of S.
2.) Determine $p_1 \in S$, the point with smallest y-coordinate (breaking ties by maximizing the x-coordinate). Mark p_1 as visited.
3.) Set $p^* = p_1$ as the *pivot* point. Let p_2 be the first point encountered by rotating counterclockwise the rightwards ray emanating from p_1 (breaking ties by picking the point closest to p^*); mark p_2 as visited. Set $p^{**} = p_2$; we call p^{**} the *pass through* point. Set $\widehat{p^*}$ = null; $\widehat{p^*}$ is the *previous pivot point*. Draw the segment $\overline{p^*p^{**}}$.
4.) **while** all points have not been visited

Find the next unvisited point q encountered by rotating ccw the ray $\overrightarrow{p^*p^{**}}$ about the pivot point p^*. (q is found in time $O(\log n)$ using binary search on points of depth within 1 of the depth of p^*. Ties are broken by picking the point closest to p^*.)
if $p^{**}p^*q$ forms a right turn
Mark q as visited
Draw segments $\overline{p^*q}$ and $\overline{p^{**}q}$
Set $\widehat{p^*} = p^*$
Set $p^* = q$
else
Set $p^{**} = p^*$
Set $p^* = \widehat{p^*}$
Set $\widehat{p^*}$ = null
end while

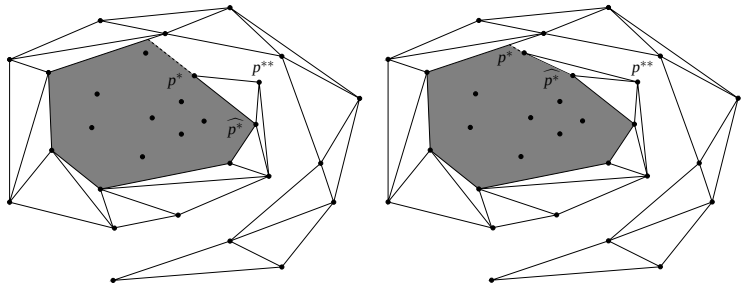


FIGURE 2. The states of the algorithm at iterations i and $i+1$, respectively, during the execution of `SpiralSerpentinePolygonize` on an example point set. The gray region indicates where unvisited points may lie. No point will be marked as visited during the $(i+1)^{th}$ iteration in this example.

In Figure 2 we illustrate a typical state of the algorithm at iteration i. The region of points yet to be visited is depicted in gray. In this example, the point with the maximum y-coordinate among the unvisited points in the gray region will be marked as visited during the i^{th} iteration. After updating the labels of the points according to the **if** clause, we obtain the state shown on the right. Clearly, the next point q encountered by rotating $\overrightarrow{p^*p^{**}}$ will not be such that $p^{**}p^*q$ forms a right turn. Thus, the **else** clause is executed, causing a "back-up" relabeling of the pivot point and reassignment of the pass through point.

2 Correctness

Lemma 2.1. *The while loop of* `SpiralSerpentinePolygonize` *terminates within* $2n$ *iterations.*

The reasoning behind this claim is that execution of the **else** clause of the **while** loop, during which no point is marked as visited, can only happen at most on every other iteration of the loop. We now show that the algorithm's output is a spiral polygon:

Lemma 2.2. `SpiralSerpentinePolygonize` *constructs a spiral polygon* P.

Proof. The proof is by induction on the iteration count. After the first iteration, we have just one triangle, which is trivially spiral. Assume the claim holds after $k < n$ iterations and consider the state after the $(k+1)^{th}$ iteration. We remove the point q_{k+1} that was most recently appended along with the two edges incident to this point (if no point was appended on the $(k+1)^{th}$ iteration, then the triangulation is unchanged and we are done). Use labels $\widehat{p^*}$, p^*, p^{**} as assigned at the end of the k^{th} iteration. The resulting polygon is spiral by the induction hypothesis. We need to show that, when q_{k+1} is processed, (a) p^{**} remains a convex vertex, and (b) p^* becomes a reflex vertex (unless $p^* = p_1$).

We consider p^{**} first. When this vertex was originally marked as visited during iteration $j < k$, it took on the label p^* during iteration j. On the subsequent iteration, no unvisited point q was discovered such that $p^{**}p^*q$ formed a right turn; otherwise, p^* would have been given the label $\widehat{p^*}$ and could never become a pass through point during some future iteration. It follows that any unvisited point q (including q_{k+1}) encountered in a subsequent iteration must be to the left of the oriented line $p^{**}p^*$. Hence, p^{**} remains a convex vertex.

We now look at p^*. If $p^* = p_1$, then it is on the convex hull and will necessarily be a convex vertex.

Now, let i be the first iteration for which $p^* \neq p_1$ and $p^{**}p^*q$ forms a right turn. If no such i exists, we argue as before. Since p^* necessarily has a depth of 2 at iteration i, there must be a point to the left of the oriented line p_1p^* with an edge to p^* in the polygonization, ensuring that p^* is reflex. This establishes the base case.

Suppose the claim holds after $k > i$ iterations. Then we wish to show that the current pivot point p^* becomes a reflex vertex upon the appendage of q_{k+1}. If no point is marked as visited on the $(k+1)^{th}$ iteration, then we simply relabel the pivot and pass through points, and the polygonization remains the same. Otherwise, we observe that q_{k+1} cannot be to the right of the oriented lined vp^*, where v is the reflex vertex previous to p^* along the reflex polygonal chain ($v = \widehat{p^*}$ if $\widehat{p^*} \neq$ null); otherwise, q_{k+1} would have been discovered previously according to the behavior of the algorithm. In other words, as in the left image of Figure 2 ($v = \widehat{p^*}$ in this example), the remaining unvisited points

are to the left of the oriented line vp^*. It follows that vp^*q_{k+1} forms a left turn, making p^* a reflex vertex. We conclude that Q is spiral. □

The subsequent two lemmas establish that, at the completion of the algorithm, a serpentine triangulation of the polygon is constructed. Although the proofs are omitted here, they are based on the observation that the algorithm constructs the polygon by simply attaching triangles iteratively. The serpentine property follows from each triangle being attached to a visible edge of the most recently formed triangle.

Lemma 2.3. `SpiralSerpentinePolygonize` *constructs a triangulation T of P.*

Lemma 2.4. *The triangulation T constructed by* `SpiralSerpentinePolygonize` *is serpentine.*

The previous four lemmas yield the desired result, stated in the following theorem:

Theorem 2.5. `SpiralSerpentinePolygonize` *constructs a spiral serpentine polygon.*

We see below that we need only consider a subset of all unvisited points during an iteration of the algorithm's **while** loop. Finally, we examine the algorithm's runtime and space usage:

Lemma 2.6. *When searching for the next unvisited point during the execution of* `SpiralSerpentinePolygonize`*, we need only consider points having depth within one of the depth of the current pivot point.*

Theorem 2.7. `SpiralSerpentinePolygonize` *runs in $O(n \log n)$ time and requires $O(n)$ space.*

Proof. The convex layers of S can be computed in $O(n \log n)$ time using $O(n)$ space [**4**]. Each subsequent point marked as visited during the execution of the while loop has depth within one of the depth of the current pivot point by Lemma 2.6. Hence, we may perform binary type searches on unvisited points of candidate convex layers in $O(\log n)$ time per iteration. There are at most $2n$ iterations by Lemma 2.1 arriving us at the desired $O(n \log n)$ run time. Since the convex layers, input points, and outputted segments can be stored in arrays of linear size, we require just $O(n)$ space. □

References

[1] M. Abellanas, J. García, G. Hernández-Peñalver, F. Hurtado, O. Serra, and J. Urrutia, Onion polygonizations, *Information Processing Letters* **57** (1996), 165–173.

[2] P. K. Agarwal, F. Hurtado, G. T. Toussaint, and J. Trias, On polyhedra induced by point sets in space, *Discrete Applied Mathematics* **156** (2008), 42–54.

[3] E. Arkin, S. P. Fekete, F. Hurtado, J. S. B. Mitchell, M. Noy, V. Sacristán, and S. Sethia, On the reflexivity of point sets, *Discrete and Computational Geometry: The Goodman–Pollack Festschrift, Algorithms and Combinatorics* **25** (2003), 139–156.

[4] B. Chazelle, On the convex layers of a planar set, *IEEE Transactions on Information Theory* **31** (1985), 509–517.

[5] L. Deneen and G. Shute, Polygonizations of point sets in the plane, *Discrete and Computational Geometry* **3** (1988), 77–87.

[6] S. P. Fekete, On simple polygonizations with optimal area, *Discrete and Computational Geometry* **23** (2000), 73–110.

[7] M. Newborn and W. O. J. Moser, Optimal crossing-free hamiltonian circuit drawings of K_n, *Journal of Combinatorial Theory, Series B* **29** (1980), 13–26.

Rectilinear convex hull with minimum area

Carlos Alegría-Galicia[3], Tzolkin Garduño[3], Areli Rosas-Navarrete[3], Carlos Seara[2], Jorge Urrutia[1]

[1] Instituto de Matemáticas, Universidad Nacional Autónoma de México (UNAM)
urrutia@matem.unam.mx

[2] Universitat Politècnica de Catalunya (UPC)
carlos.seara@upc.edu

[3] Posgrado Ciencia e Ingeniería de la Computación, Universidad Nacional Autónoma de México (UNAM)
{alegria_c,garduno_t,areli}@uxmcc2.iimas.unam.mx

Abstract. Let P be a set of n points in the plane. We solve the problem of computing the orientations for which the rectilinear convex hull of P has minimum area in optimal $\Theta(n \log n)$ time and $O(n)$ space.

Introduction

The interest in the rectilinear convex hull of planar point sets arises from the study of ortho-convexity [**10**], a relaxation of traditional convexity. Unlike convex regions, an ortho-convex region might be disconnected, which makes the study of the ortho-convex closure for a point set [**5, 8**] harder. Several definitions have been presented by different authors. We will use a definition stated by Ottman et al. [**8**] as the *mr-convex hull*, see also Matoušek et al. [**5, 7**]. The study of rectilinear convex hulls has gained attention partly because of some applications in digital image processing [**3**] and VLSI circuit layout design [**11**].

The rectilinear convex hull of point sets is an orientation-dependent region, i.e., it changes as the orientation of the plane changes. In this paper we are interested in computing an orientation for which the rectilinear convex hull of P has minimum area. We show that the set of orientations $\theta \in [0, 2\pi)$ can be divided into a set of linear intervals such that, within each interval I, the angle $\theta \in I$ which minimizes the area of the rectilinear convex hull of a point set (save the first one we process) can be calculated in constant time. These intervals can be computed in $O(n \log n)$ time and $O(n)$ space. Using this result and based on techniques from Avis et al. [**1**], Bae et al. [**2**], and Díaz-Báñez et al. [**4**], we present an optimal $\Theta(n \log n)$ time and $O(n)$ space algorithm for this problem. Our result improves the $O(n^2)$ time complexity presented by Bae et al. [**2**].

1 Terminology and notation

An *orthogonal wedge* is the intersection of two open half-planes whose supporting lines are orthogonal. The *apex* of the wedge is the intersection point of these supporting lines. An orthogonal wedge is *P-free* if it does not contain points of P in its interior. An orthogonal wedge is called a *θ-wedge* if its supporting lines can be obtained by first rotating the X-

[1]Coauthors from UNAM are partially supported by projects MTM2006-03909 of Spain and SEP-CONACYT 80268 of México. The coauthor from UPC is partially supported by projects MTM2009-07242 and Gen. Cat. DGR 2009GR1040.

and Y-axis θ degrees, and then translating the origin to the apex of our wedge. The *rectilinear convex hull of P with orientation θ* is the region

$$\mathcal{RH}_\theta(P) = \mathbb{R}^2 - \bigcup_{w \in \mathcal{W}_\theta} w,$$

where $\mathcal{W}_\theta$ is the set of all P-free orthogonal θ-wedges [**2, 4, 8**].

As θ changes, the set of orthogonal P-free θ-wedges change, and, thus, $\mathcal{RH}_\theta(P)$ changes (see Figure 1). A θ-orientation of the plane, $\theta \in [0, 2\pi)$, is the coordinate system obtained by rotating the axes of $\mathbb{R}^2$ by θ degrees with respect to the origin. For a fixed θ, $\mathcal{RH}_\theta(P)$ has a close relation to the maxima problem [**6, 9**]. A *vertex* of $\mathcal{RH}_\theta(P)$ is a point in P that lies on the boundary of $\mathcal{RH}_\theta(P)$. Let $X_\theta(P)$ be the set of maximal points of P with respect to vector dominance in the θ-orientation of the plane. The set of vertices of $\mathcal{RH}_\theta(P)$ is equal to the set $X_\theta(P) \cup X_{\theta+\frac{\pi}{2}}(P) \cup X_{\theta+\pi}(P) \cup X_{\theta+\frac{3}{2}\pi}(P)$ [**2, 8**]. Given a fixed θ, $\mathcal{RH}_\theta(P)$ can be computed in optimal $\Theta(n \log n)$ time and $O(n)$ space [**6, 9**].

We say that a point $p \in P$ is *θ-maximal with respect to P* if there is an orthogonal P-free wedge with apex at p in a θ-orientation of the plane. The set of orientations for which p is θ-maximal forms at most three intervals. The endpoints of each interval mark the *in-* and an *out-* events of p, i.e., the θ-orientations when p becomes and stops being θ-maximal. The set of intervals corresponding to the elements of P and the set of angles at which these points of P start and stop being θ-maximal can be computed in optimal $\Theta(n \log n)$ time and $O(n)$ space [**4**].

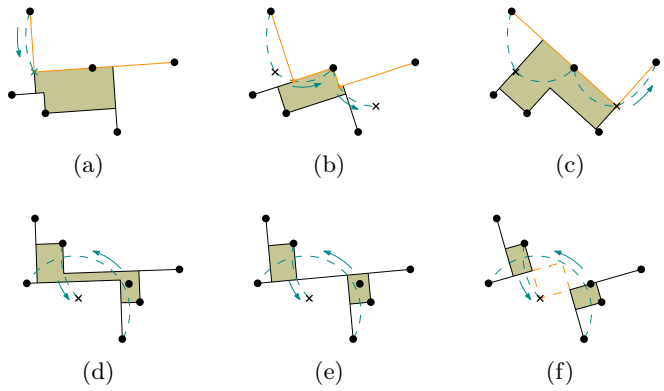


FIGURE 1. The rectilinear convex hull of P changes with the orientation.

Let *X_θ-axis* and *Y_θ-axis* denote the coordinate axes rotated θ degrees. For a θ orientation, consider the coordinates of the points of P in terms of the X_θ- and Y_θ-axes. Since $\mathcal{RH}_\theta(P)$ is monotone with respect to the X_θ-axis [**8**], the points of P can be re-labelled as $v_1, \ldots, v_m$ in increasing order according to X_θ. Two consecutive points $v_i, v_{i+1} \in P$ with respect to X_θ define the *step* $s_\theta(v_i, v_{i+1})$. Given two orientations α and β, we say that two steps $s_\alpha(v_i, v_{i+1})$ and $s_\beta(v_j, v_{j+1})$ are *opposite* to each other if $|\alpha - \beta| = \pi$; see Figure 1(b). Every step $s_\theta(v_i, v_{i+1})$ supports a P-free θ-wedge. Let W_1 and W_2 be the wedges supported by two opposite steps $s_\theta(v_i, v_{i+1})$ and $s_{\theta+\pi}(v_j, v_{j+1})$, respectively. If W_1 and W_2 intersect, $\mathcal{RH}_\theta(P)$ is disconnected. In such case, we say that $s_\theta(v_i, v_{i+1})$ and $s_{\theta+\pi}(v_j, v_{j+1})$ *overlap*, and denote $W_1 \cap W_2 = t_\theta(i, j)$; see Figure 1(f).

Consider four points v_i, v_{i+1} and v_j, v_{j+1} and let I be the the interval of orientations θ for which $s_\theta(v_i, v_{i+1})$ and $s_{\theta+\pi}(v_j, v_{j+1})$ overlap. As before, we call the ends of I the *start-* and *stop*-events of $t_\theta(i, j)$; see Figures 1(e) and 1(f). We wish to compute a counterclockwise ordered start- and stop-event list that resembles the one we computed for in- and out-events of the elements of P. Overlap events are not necessarily vertex events and thus, they have to be computed independently.

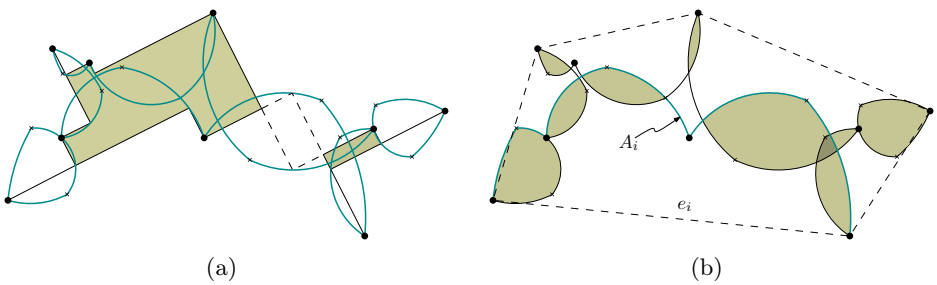
2 Computing the start- and stop-overlap events list

The *apex* of a step is the apex of the wedge that it supports. As θ changes from 0 to 2π, the θ-orientation of the plane *rotates* counterclockwise, and the apex of every step traces a circular arc. We orient the arcs traced by the elements of P as shown in Figures 1(a)–1(c). The *arc-chain* $\mathcal{A}(P)$ of P is the closed curve formed by the union of the set of arcs traced by the elements of P which, at some point in time are θ-maximal for some $\theta \in [0, 2\pi)$, let $\mathcal{A}(P) = \langle a_1, \ldots, a_l \rangle$ (Figure 2(a)). Since there is a linear number of steps in a complete rotation, $l = O(n)$. Observe that the endpoints of the arcs in $\mathcal{A}(P)$ include the points in P that are θ-maximal for some $\theta \in [0, 2\pi)$.

Let $\{e_1, \ldots, e_h\}$, be the set of edges of the convex hull $\mathcal{CH}(P)$ of P in counterclockwise order. A *sub-chain* $A_i(P)$ of $\mathcal{A}(P)$ is a subsequence of consecutive elements of $\mathcal{A}(P)$, whose endpoints are the endpoints of e_i. It is easy to see that $A_i(P)$ is monotone in the direction determined by e_i. Thus the orthogonal projection of $A_i(P)$ on e_i defines a total order ($\prec_i$) on the set of endpoints of its arcs. Moreover, using the fact that every point in $A_i(P)$ is an apex of a P-free wedge, the next lemma follows easily.

Lemma 2.1. *Let a, b, c be three points in $A_i(P)$ such that $a \prec_i b \prec_i c$. Then, the angle $\angle abc$ is such that $\frac{\pi}{2} \leq \angle abc < \pi$.*

Suppose that we relabel the endpoints of the arcs in $A_i(P)$ as $p_1, \ldots, p_m$ so that, if $r < s$, then $p_r \prec_i p_s$. Let $\ell_{r,s}$ be a subsequence $p_r, \ldots, p_s$ of $A_i(P)$ such that for $r < t < s$, $p_t \notin P$ and $p_r, p_s \in P$. We call any such $\ell_{r,s}$ a *link*. Observe that, if two opposite steps overlap, then the arcs traced by their apices belong to links that intersect; see Figure 2(a). The open area bounded by $A_i(P)$ and e_i is P-free, since it is covered by P-free wedges. Thus, two intersecting links have at least two intersection points. By Lemma 2.1, this number is tight, as none of the intersecting links can cross a line segment joining its intersection points; see Figure 2(b). Thus we have the following result, that is a central tool for computing the start- and stop-overlap event list in $O(n \log n)$ time.



(a) (b)

FIGURE 2. The *arc-chain* $\mathcal{A}(P)$ of P.

Theorem 2.2. *There are $O(n)$ intersections between links in $\mathcal{A}(P)$.*

3 Computing the orientation of $\mathcal{RH}_\theta(P)$ with minimum area

The event points obtained in the previous section generate a set of intervals of orientations, in which the set of vertices of $\mathcal{RH}_\theta(P)$ remain unchanged, and the set of overlaps among the steps of $\mathcal{RH}_\theta(P)$ does not change. Let $I_{(\theta_1,\theta_2)}$ be one such interval. Then, for any $\theta \in (\theta_1, \theta_2)$, the area of $\mathcal{RH}_\theta(P)$ is given by the following formula [**2**]:

$$\text{area}(\mathcal{RH}_\theta(P)) = \text{area}(\mathcal{P}) - \sum \text{area}(s_\theta(v_i, v_{i+1})) + \sum \text{area}(t_\theta(i,j)).$$

It is easy to see that the areas of the steps $s_\theta(v_i, v_{i+1})$ and overlaps $t_\theta(i,j)$ of $\mathcal{RH}_\theta(P)$ can be expressed as a function of $\sin 2\theta$ and $\cos 2\theta$. Doing the derivative, we obtain:

$$\sum \text{area}'(s_\theta(v_i, v_{i+1})) = -\big[\sum A_i\big] \sin 2\theta + \big[\sum B_i\big] \cos 2\theta, \tag{1}$$

$$\sum \text{area}'(t_\theta(i,j)) = \big[\sum C_i\big] \cos 2\theta - \big[\sum D_i\big] \sin 2\theta, \tag{2}$$

and thus the value $\theta \in (\theta_1, \theta_2)$ for which the area of $\mathcal{RH}_\theta(P)$ is minimized can be computed in linear time. For each new event interval, we update these values in constant time by subtracting or adding new constant values. There can be more than one θ-orientation in which $\mathcal{RH}_\theta(P)$ has minimum area, but our algorithm is able to report all of them. From the discussion above and from the fact that the convex hull of P can be computed from the rectilinear convex hull of P in $O(n)$ time, we obtain the following:

Theorem 3.1. *Computing the set of orientations for which the rectilinear convex hull of P has minimum area can be done in optimal $\Theta(n \log n)$ time and $O(n)$ space.*

References

[1] D. Avis, B. Beresford-Smith, L. Devroye, H. Elgindy, E. Guévremont, F. Hurtado, and B. Zhu, Unoriented Θ-maxima in the plane: complexity and algorithms, *SIAM J. Comput.* **28:1** (1999), 278–296.

[2] S. W. Bae, C. Lee, H.-K. Ahn, S. Choi, and K.-Y. Chwa, Computing minimum-area rectilinear convex hull and L-shape, *Computational Geometry: Theory and Applications* **42:3** (2009), 903–912.

[3] A. Biswas, P. Bhowmick, M. Sarkar, and B. B. Bhattacharya, Finding the orthogonal hull of a digital object: a combinatorial approach, *Combinatorial Image Analysis*, LNCS 4958, 2008, 124–135.

[4] J. M. Díaz-Báñez, M. A. López, M. Mora, C. Seara, and I. Ventura, Fitting a two-joint orthogonal chain to a point set, *Computational Geometry: Theory and Applications* **44:3** (2011), 135–147.

[5] V. Franěk and J. Matouseˇk, Computing D-convex hulls in the plane, *Computational Geometry: Theory and Applications* **42** (2009), 81–89.

[6] H. T. Kung, F. Luccio, and F. P. Preparata, On finding the maxima of a set of vectors, *Journal of the ACM* **22** (1975), 469–476.

[7] J. Matoušek and P. Plecháč, On functional separately convex hulls, *Discrete and Computational Geometry* **19** (1998), 105–130.

[8] T. Ottman, E. Soisalon-Soisinen, and D. Wood, On the definition and computation of rectilinear convex hulls, *Information Sciences* **33** (1984), 157–171.

[9] F. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, 1985.

[10] G. J. E. Rawlins and D. Wood, Ortho-convexity and its generalizations, in: *Computational Morphology: A Computational Geometric Approach to the Analysis of Form*, Machine Intelligence and Pattern Recognition, 6, Elsevier Science Publishers B.V., North-Holland, 1988, 137–152.

[11] E. Uchoa, M. P. de Aragão, and C. C. Ribeiro, Preprocessing Steiner problems from VLSI layout, *Networks* **40:1** (2002), 38–50.

Locating a service facility and a rapid transit line

J. M. Díaz-Báñez[1], M. Korman[2], P. Pérez-Lantero[3], I. Ventura[1]

[1] Departamento de Matemática Aplicada II, Universidad de Sevilla, Spain
{dbanez,iventura}@us.es

[2] Université Libre de Bruxelles (ULB)
mkormanc@ulb.ac.be

[3] Departamento de Computación, Universidad de Valparaíso, Chile
pablo.perez@uv.cl

Abstract. In this paper we study a facility location problem in the plane in which a single point (facility) and a rapid transit line (highway) are simultaneously located in order to minimize the total travel time from the clients to the facility, using the L_1 or Manhattan metric. The rapid transit line is given by a segment with fixed length and it is an alternative transportation line that can be used by the clients to reduce their travel time to the facility. This problem was introduced by Espejo and Chía in [**4**]. They gave both a characterization of the optimal solutions and an algorithm running in $O(n^3 \log n)$ time, where n represents the number of clients. In this paper it is shown that the Espejo and Chía's algorithm does not always work correctly. At the same time, we provide a proper characterization of the solutions with a simpler proof and give an algorithm solving the problem in $O(n^3)$ time.

1 Introduction

In computational geometry, geometric problems related to transportation networks have been recently considered. Abellanas *et al.* introduced the *time metric* model in [**1**]: Given an underlying metric, the user can travel at speed $\nu(h)$ when moving along a highway h or unit speed elsewhere. The particular case in which the underlying metric is the L_1 metric and all highways are axis-parallel segments of the same speed, is called the *city metric* [**2**]. The optimal positioning of transportation devices that minimize the maximum travel time among a set of points has been deeply treated recently [**5**]. In the variant introduced by Espejo and Chía [**4**], the objective is to minimize the sum of the travel times from the demand points to the new facility service that has to be located simultaneously with a highway. The highway is used by a demand point whenever it saves time to reach the facility. Díaz-Báñez *et al.* [**3**] study a variation of this problem in which they want to minimize the largest travel time between the clients and the facility.

We introduce some notation. Let S be the set of n client points, f be the service facility point, h be the highway, ℓ be the length of h, t and t' be the endpoints of h, and $v > 1$ be the speed in which the points move along h. Let $w_p > 0$ be the weight (or demand) of a client point p. Given a point u of the plane, let $x(u)$ and $y(u)$ denote the x- and y-coordinates of u respectively. The travel time between a point p and the service facility f is denoted by

$$d_{t,t'}(p, f) = \min\{\|p - f\|_1,\ \|p - t\|_1 + \tfrac{\ell}{v} + \|t' - f\|_1,\ \|p - t'\|_1 + \tfrac{\ell}{v} + \|t - f\|_1\},$$

where $\|\cdot\|_1$ denotes the L_1 norm. The problem is formulated as follows:

[1]Partially supported by MEC project MTM2009-08652.
[3]Partially supported by MEC project MTM2009-08652.

Facility and Highway Location Problem (FHL-problem)**.** Given a set S of n points, a weight $w_p > 0$ associated with each point p of S, a fixed highway length $\ell > 0$, and a fixed speed $v > 1$, locate a point (facility) f and a line segment (highway) h of length ℓ with endpoints t and t' such that the function $\sum_{p \in S} w_p \cdot d_{t,t'}(p, f)$ is minimized.

In Section 2 we first provide a proper characterization of the solutions. Our proof uses geometric observations and is simpler than the proof given in [**4**]. After that, we show that Espejo and Chía's characterization is not true in general. In Section 3 we present an improved algorithm running in $O(n^3)$ time that correctly solves the FHL-problem. Due to space constraints, some of the proofs are omitted.

2 Properties of an optimal solution

A straightforward observation (also stated in [**4**]) is that the service facility can be located at one of the endpoints of the rapid transit line. Therefore, we assume throughout the paper that $f = t'$, thus the distance from a point $p \in S$ to the facility f is now $d_t(p, f) = \min\{\|p - f\|_1, \|p - t\|_1 + \frac{\ell}{v}\}$.

We say that a point p uses the highway if $\|p-t\|_1+\frac{\ell}{v} < \|p-f\|_1$, and that p does not use it (or goes directly to the facility) otherwise. Given f and t, we say that the bisector of f and t is the set of points z such that $\|z - f\|_1 = \|z - t\|_1 + \frac{\ell}{v}$; see Figure 1.

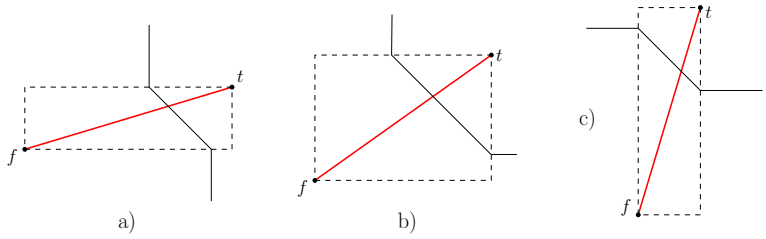


FIGURE 1. The bisector of f and t.

Consider the grid G defined by the set of all axis-parallel lines passing through the elements of S.

Lemma 2.1. *There exists an optimal solution to the FHL-problem satisfying one of the next conditions: (a) One of the endpoints of the highway is a vertex of G; (b) one endpoint of the highway is on a horizontal line of G, and the other endpoint is on a vertical line of G.*

Proof. Let f and t be the endpoints of an optimal highway h and assume that none of the two conditions is satisfied. Using local perturbation, we will transform this solution into one that satisfies one of the two conditions. Assume that neither f nor t is on vertical lines of G. Let $\delta_1 > 0$ (resp. $\delta_2 > 0$) be the smallest value such that if we translate h with vector $(-\delta_1, 0)$ (resp. $(\delta_2, 0)$) then one endpoint of h touches a vertical line of G. Given $\varepsilon \in [-\delta_1, \delta_2]$, let f_ε, t_ε, and h_ε be f, t, and h translated with vector $(\varepsilon, 0)$, respectively. We partition S into three sets B, L and R as follows:

- $s \in B$ if s is in the bisector between f and t;
- $s \in L$ if s walks to f and $x(s) > x(f)$ or if s uses the highway and $x(s) > x(t)$;
- $s \in R$ if s walks to f and $x(s) < x(f)$ or if s uses the highway and $x(s) < x(t)$.

That is, the set L contains the demand points that travel leftwards to reach f (analogously R contains the points that travel rightwards). Let $w_B = \sum_{p \in B} w_p$ be the sum of weights of the points in set B (we define w_L and w_R analogously). By linearity of the L_1 metric, for any $\varepsilon \in [-\delta_1, \delta_2]$ the change in the objective function is as follows:

$$\sum_{p \in S} w_p d_{t_\varepsilon}(p, f_\varepsilon) = \sum_{p \in S} w_p \cdot d_t(p, f) + \varepsilon(w_R - w_L) - |\varepsilon| w_B.$$

By optimality of f and t, we must have $w_R - w_L = 0$ and $w_B = 0$. In particular, this implies that $B = \emptyset$ and that we can translate h either rightwards or leftwards until one of the highway endpoints reaches a vertical line of G. We repeat the same operation on the y coordinates and also obtain that one of the two endpoints must be on a horizontal line of G, hence satisfing one of the two conditions of the Lemma. □

In [**4**] the authors stated that there always exists an optimal solution satisfying Lemma 2.1 (a). Unfortunately, the above claim is false and their algorithm may miss some highway locations; indeed, it may miss the optimal location and thus fail. We provide here one counterexample —see Figure 2 and the following result.

Lemma 2.2. *There exists a set of unweighted points in which no optimal solution to the FHL-problem satisfies Lemma* 2.1 (a).

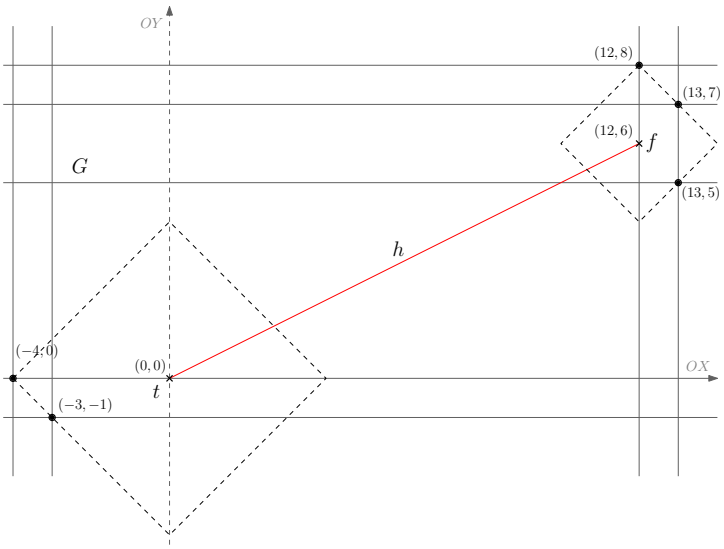


FIGURE 2. A counterexample to the algorithm of Espejo and Chía.

3 The algorithm

We will discuss the solution of the FHL-problem for the case in which Lemma 2.1 (a) holds. The case in which Lemma 2.1 (b) holds can be treated similarly. For each vertex u of the grid G we can solve the problem subject to $f = u$ or $t = u$. We show how to obtain a solution if $f = u$. The case where $t = u$ can be solved analogously.

We will assume without loss of generality that $\theta \in [0, \frac{\pi}{4}]$. Given a point u and an angle θ, let $u(\theta)$ be the point with coordinates $(x(u) + \cos\theta, y(u) + \sin\theta)$. There exists an angle $\phi \in [0, \frac{\pi}{4}]$ such that the bisector of the endpoints u and $u(\theta)$ has the shape in Figure 1 (a) for all $\theta \in [0, \phi)$, and has the shape in Figure 1 (b) for all $\theta \in (\phi, \frac{\pi}{4}]$.

The distance between a point $p \in S$ and the facility u has the expression $c_1 + c_2 \cos\theta + c_3 \sin\theta$, where $c_1 > 0$ and either $c_2, c_3 = \pm w_p$ (p uses the highway) or $c_2 = c_3 = 0$ (p does not use the highway). When θ goes from 0 to $\frac{\pi}{4}$ this expression changes at the values of θ such that one of the following conditions is satisfied: *a*) The point p switches from using the highway to going directly to the facility (or vice versa). We call these changes *bisector events*; *b*) the highway endpoint $u(\theta)$ crosses the vertical or horizontal line passing through p —we call this event a *grid event*; *c*) $\theta = \phi$. This event is called a *ϕ-event.*

Lemma 3.1. *After an $O(n \log n)$-time preprocessing, the angular order of all the events associated with a given vertex of G can be obtained in linear time.*

Proof (sketch). Let Π_x, Π_y, and Π_{x+y} denote the point set S sorted according to the x-, y-, and $(x + y)$-order, respectively, and let u be a vertex of G. It is straightforward to see that they are $O(n)$ grid events and that we can obtain their angular order in linear time by using both Π_x and Π_y. The calculation of the bisector events in linear time needs more elaboration. The bisector of u and $u(\theta)$ consists of two axis-aligned half-lines and a line segment with slope -1 connecting their endpoints. Given a point $p \in S$, if p belongs to the line segment of the bisector then the event is denoted by α_p. If p belongs to the leftmost half-line of the bisector, which is always vertical, we denote that event by β_p. Otherwise, if p belongs to the rightmost half-line, which can be either vertical or horizontal, we denote that event by γ_p.

Let Π_1 be the subsequence of Π_{x+y} containing all elements p such that $\alpha_p \in [0, \frac{\pi}{4}]$, Π_2 be the subsequence of Π_x containing all elements p such that $\beta_p \in [0, \frac{\pi}{4}]$, and Π_3 be the subsequence of Π_x that contains all elements p such that $y(p) < y(u)$ and $\gamma_p \in [0, \frac{\pi}{4}]$, concatenated with the subsequence of Π_y that contains all elements p such that $x(p) > x(u)$ and $\gamma_p \in [0, \frac{\pi}{4}]$. Given a point $p \in S$, the corresponding events of p in $[0, \frac{\pi}{4}]$ can be found in constant time, thus Π_1, Π_2, and Π_3 can be built in linear time.

Let Γ_1 (resp. Γ_2, Γ_3) be the sequence obtained by replacing each element p in Π_1 (resp. Π_2, Π_3) by α_p (resp. β_p, γ_p). Then Γ_1, Γ_2, and Γ_3 are sorted sequences. We now merge in linear time Γ_1, Γ_2, Γ_3, the grid events, and the ϕ-event. Thus, the angular order of all events associated with a vertex u can be obtained in $O(n)$ time. □

Lemma 3.2. *After an $O(n \log n)$-time preprocessing, the angular order of all the events associated with a pair of perpendicular lines of G (case Lemma 2.1 (b)) can be obtained in linear time.*

Theorem 3.3. *The FHL-problem can be solved in $O(n^3)$ time.*

References

[1] M. Abellanas, F. Hurtado, V. Sacristán, C. Icking, L. Ma, R. Klein, E. Langetepe, and B. Palop, Voronoi diagram for services neighboring a highway. *Inf. Process. Lett.* **86**, 5 (2003), 283–288.

[2] O. Aichholzer, F. Aurenhammer, and B. Palop, Quickest paths, straight skeletons, and the city Voronoi diagram. in: *Proc. of SCG'02*, ACM, New York, NY, USA, 2002, 151–159.

[3] J. M. Díaz-Báñez, M. Korman, P. Pérez-Lantero, and I. Ventura, The 1-center and 1-highway problem, in: *XIV Spanish Meeting on Computational Geometry*, P. Ramos and V. Sacristán (eds.). CRM Documents, vol. 8, Centre de Recerca Matemàtica, Bellaterra (Barcelona), 2011, 193–196.

[4] I. Espejo and A. M. Rodríguez-Chía, Simultaneous location of a service facility and a rapid transit line. *Comput. Oper. Res.* **38** (2011), 525–538.

[5] M. Korman and T. Tokuyama, Optimal insertion of a segment highway in a city metric, in: *Proc. of COCOON'08*, X. Hu and J. Wang (eds.). Springer-Verlag, Berlin, Heidelberg, 2008, 611–620.

The 1-center and 1-highway problem

J. M. Díaz-Báñez[1], M. Korman[2], P. Pérez-Lantero[3], I. Ventura[1]

[1] Departamento de Matemática Aplicada II, Universidad de Sevilla, Spain
{dbanez,iventura}@us.es

[2] Université Libre de Bruxelles (ULB)
mkormanc@ulb.ac.be

[3] Departamento de Computación, Universidad de Valparaíso, Chile
pablo.perez@uv.cl

Abstract. We study a variation of the 1-center problem, in which, in addition to a single supply facility, we are allowed to locate a highway. This highway increases the transportation speed between any demand point and the facility. That is, given a set S of points and $v > 1$, we are interested in locating the facility point f and the highway h that minimize the expression $\max_{p \in S} d_h(p, f)$, where d_h is the time distance between p and f. We show that we can find the optimal location of both the facility and the highway in $O(n^2)$ or $O(n \log n)$ time, depending on whether or not the highway's length is fixed.

1 Introduction

Geometric optimization related to urban transportation systems is an important topic in computational geometry. Although the metric given by a real urban transportation system is often quite complicated, simplified mathematical models have been widely studied in order to investigate basic geometric properties of urban transportation systems. Abellanas *et al.* [**1**] considered a geometric modeling of this environment: represent highways as polygonal chains consisting of line segments in the plane, giving each line segment an associated speed. Then, the travel time between two points gives a metric called the *time distance*.

Recently, there has been an interest in problems derived from urban modeling. In many cases we are interested in locating a highway that optimizes some given function that depends on the distance between elements of a given point set (see for example [**2, 4, 7**]). Espejo and Chía [**6**] introduced a variant of the problem in which we are given a set of clients (represented by a set of points S) located in a city. Then, one is interested in locating a service facility and a highway simultaneously in a way that the average supply time between the clients and the supply point is minimized. Unfortunately, it was shown that their algorithm could give an incorrect solution in some cases [**5**]. In this paper we study a variation of this problem in which we want to minimize the largest travel time between the clients and the facility.

2 Definitions and notation

Let S be the set of n client points, f be the service facility point, h be the highway, ℓ be the length of h, t and t' be the endpoints of h, and $v > 1$ be the speed. We assume that

[1]Partially supported by MEC project MTM2009-08652.
[3]Partially supported by MEC project MTM2009-08652.

the highway to locate can have any orientation. Given a point u of the plane, let $x(u)$ and $y(u)$ denote respectively the x and y coordinates of u.

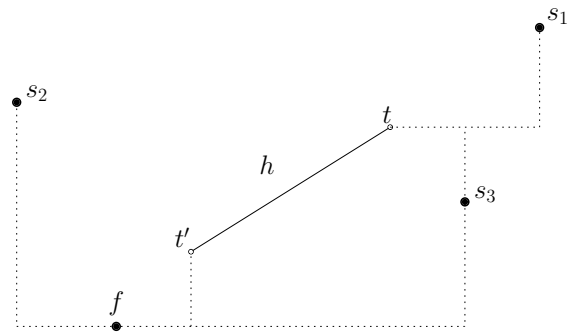


FIGURE 1. The distance model; in the example, s_1 uses the highway from t to t' in order to reach f faster. The highway does not speed up transportation between s_2 and f, hence is not used by s_2. Demand point s_3 however, can either walk or use the highway to reach f, and will need the same time in both cases. Observe that, since we are interested in paths that reach f, the highway will only be used in one direction.

Fixed the location of the facility and the endpoints of the highway, the distance from a demand point $p \in S$ to f is defined as

$$d_h(p,f) = \min\left\{\|p-f\|_1,\ \|p-t\|_1 + \frac{\ell}{v} + \|t'-f\|_1,\ \|p-t'\|_1 + \frac{\ell}{v} + \|t-f\|_1\right\},$$

where $||\cdot||_1$ represents the L_1 distance between two points; see Figure 1. Whenever $d_h(p,f) < \|f-p\|_1$, we say that p uses the highway to reach f. Otherwise, we say that p walks (or does not use h) to reach the facility.

The problem that we study can be formulated as follows:

The 1-center and 1-highway problem (1C1H-problem**).** Given a set S of n points and a fixed speed $v > 1$, locate a point (facility) f and a line segment (highway) h with endpoints t and t' such that the function $\max_{p\in S} d_h(p,f)$ is minimized. The case in which the highway's length $\ell = ||t-t'||_2$ is fixed is called the *fixed length* 1-Center and 1-Highway problem (FL-1C1H for short). The case in which the highway can have any length is called the *variable length* (or VL-1C1H) problem.

It is easy to see that, in either variant of the 1C1H problem, the highway will only be used in one direction. In particular, there always exists an optimal location in which one of the rapid transit line endpoints coincides with the facility. This result was also observed in [**6**, Lemma 2.1]. Therefore, we assume throughout the paper that $f = t'$; thus the distance from a demand point $p \in S$ to f is now $d_h(p,f) = \min\{\|p-f\|_1,\ \|p-t\|_1 + \frac{\ell}{v}\}$.

3 Solving the 1C1H Problem

In this section we give a general algorithm for solving the 1C1H problem. Unless otherwise stated, all results of this section hold for both variants of the problem. Due to space constraints, some of the proofs of this paper have been omitted. Using the standard transformation from L_1 to L_∞, we solve the problem using L_∞ instead. Let f^* and h^*

be an optimal solution of a given problem instance. Let t^* be the endpoint of h^* other than f^*, and let $R^* = \max_{p \in S} d_{h^*}(p, f^*)$.

Let $B(u, r)$ denote the axis-parallel square of radius r centered at u, and consider the balls $B(f^*, R^*)$ and $B(t^*, R^* - \ell/v)$. By definition of R^*, all points of S must be included in the union of both balls. We partition the pointset S into two sets W^* and H^* as follows: the set W^* contains the points whose L_∞ distance to f^* is at most R^*, while the set $H^* = S \setminus W^*$ contains the points that must use the highway to reach f^* in R^* or less units of time.

Observe that we cannot have $W^* = \emptyset$, since by reversing the positions of f^* and t^* we would obtain a better solution. By definition, the set H^* is empty if and only if all points of S can walk to f^* in R^* or less units of time. This case can be easily handled, since f^* is the solution of the rectilinear 1-center problem (which can be computed in linear time). Hence, from now on we assume that neither W^* nor H^* is empty.

We consider the next problem, called the *basic problem*: Given a partition $\{W, H\}$ of S, find the smallest value R (called the *radius* of the partition) and the coordinates of f and t such that $W \subseteq B(f, R)$ and $H \subseteq B(t, R - \ell/v)$. When we consider the fixed-length variation of the problem, we also add the constraint that f and t must satisfy $\|f - t\|_2 = \ell$. Since f^* and t^* are optimal, it is easy to see that they are the solution of the basic problem for the partition $\{W^*, R^*\}$. Moreover, the radius of any other partition of S will have equal or higher radius than R^*.

Our algorithm works as follows: we consider different partitions of S and solve the basic problem associated to each partition. We identify $\{W^*, R^*\}$ as the partition whose radius is smallest. A naive method would be to guess the partition $\{W^*, R^*\}$ among the $O(2^n)$ candidates. In the following we reduce the search space to one of polynomial size:

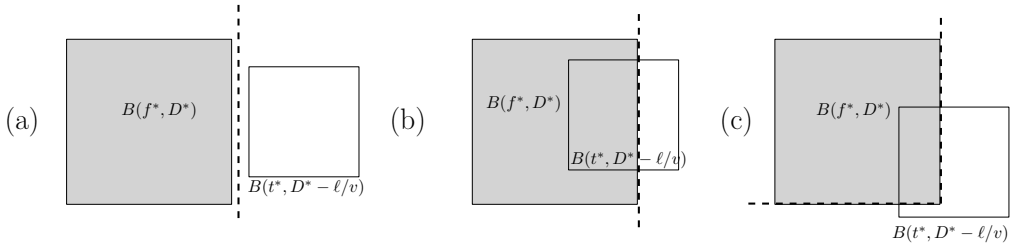


FIGURE 2. Relative positions of the balls $B(f^*, R^*)$ and $B(t^*, R^* - \ell/v)$. For each of the cases, the sets W^* (marked in grey) can be split from H^* with either an axis-aligned line or an upper-left quadrant.

Lemma 3.1. *For any set S, the partition $\{W^*, R^*\}$ can be found among $O(n^2)$ candidates.*

Proof. Without loss of generality we can assume that f^* is above and to the left of t^*. It is then easy to see that there are three possible relative positions of the two balls (see Figure 2). In each of these cases the two sets can be split by either an axis-aligned line or an upper left quadrant (see dashed lines in Figure 2). Each possible partition is uniquely determined by the number of points above and/or to the left of the splitting line/quadrant. In particular, there are $O(n^2)$ different cases, hence the Lemma is shown. □

Given a set T of points, let $X(T) \subseteq T$ be the set containing the points with highest and lowest x and y coordinate of T (this set is called the set of *extreme points* of T).

Lemma 3.2. *Let $\{W, H\}$ be a partition of S. If we are given the extreme sets $X(W)$ and $X(H)$, then the basic problem can be solved in constant time (for both fixed-length and variable-length cases).*

Theorem 3.3. *Both variants of the 1C1H problem can be solved in $O(n^2)$ time and $O(n)$ space.*

The bottleneck of the algorithm is case (c) of Lemma 3.1. In the following we show how to treat this case more efficiently for the variable-length case.

Lemma 3.4. *If in every optimal solution of the VL-1C1H problem each ball contains a corner of the other one, then there exists an optimal solution (f^*, t^*) of radius R^* of the VL-1C1H problem in which the extreme points $X(S)$ are in the boundary of $B(f^*, R^*) \cup B(t^*, R^* - \ell/v)$.*

Theorem 3.5. *The VL-1C1H problem can be solved in $O(n \log n)$ time.*

4 Concluding remarks

In our model we only allow entering and leaving the highway at its endpoints (in other literature, this kind of highway is called *walkway* [**4**] or *turnpike* [**3**]). We note that there exists another model of highway (called *freeway* [**3**] or simply *highway* [**1**]) in which one is allowed to enter and leave at any point. A natural extension of the problems studied in this paper is considering the location of a freeway instead.

The faster algorithm for the variable length variant of the problem is based on Lemma 3.4. Unfortunately, we have examples in which this result does not hold whenever the highway's length is fixed. Thus it remains open to show whether or not the fixed-length problem can also be solved in $O(n \log n)$ time (or showing that it is 3-SUM hard).

References

[1] M. Abellanas, F. Hurtado, C. Icking, R. Klein, E. Langetepe, L. Ma, B. Palop, and V. Sacristán. Voronoi diagram for services neighboring a highway. *Information Processing Letters*, 86:283–288, 2003.

[2] Hee-Kap Ahn, Helmut Alt, Tetsuo Asano, Sang Won Bae, Peter Brass, Otfried Cheong, Christian Knauer, Hyeon-Suk Na, Chan-Su Shin, and Alexander Wolff. Constructing optimal highways. *Int. J. Found. Comput. Sci.*, 20(1):3–23, 2009.

[3] S. W. Bae, M. Korman, and T. Tokuyama. All farthest neighbors in the presence of highways and obstacles. In *Proc. of the 3rd International Workshop on Algorithms and Computation*, WALCOM'09, pages 71–82, Berlin, Heidelberg, 2009. Springer-Verlag.

[4] J. Cardinal, S. Collette, F. Hurtado, S. Langerman, and B. Palop. Optimal location of transportation devices. *Comput. Geom. Theory Appl.*, 41:219–229, 2008.

[5] J. M. Díaz-Báñez, M. Korman, P. Pérez-Lantero, and I. Ventura. Locating a service facility and a rapid transit line. In *XIV Spanish Meeting on Computational Geometry*, CRM Documents, vol. 8, pages 189–192, Centre de Recerca Matemàtica, Bellaterra (Barcelona), 2011.

[6] I. Espejo and A. M. Rodríguez-Chía. Simultaneous location of a service facility and a rapid transit line. *Comput. Oper. Res.*, 38:525–538, 2011.

[7] M. Korman and T. Tokuyama. Optimal insertion of a segment highway in a city metric. In *Proceedings of the 14th International Conference on Computing and Combinatorics (COCOON'08)*, LNCS, pages 611–620, 2008.

Longest-edge refinement schemes for real-time terrain triangulations

Lucana Santos[1], **José Pablo Suárez**[2], **Ángel Plaza**[1]

[1] Institute for Applied Microelectronics (IUMA), University of Las Palmas de Gran Canaria
lsfalcon@iuma.ulpgc.es, aplaza@dmat.ulpgc.es

[2] Department of Cartography and Graphic Engineering, University of Las Palmas de Gran Canaria
jsuarez@dcegi.ulpgc.es

Abstract. In this paper we present an implementation of longest edge refinement algorithms for application in real-time terrain operations. The proposed refinement schemes are suitable tools for the subdivision of underlying mesh triangles. They pose quite acceptable properties as quality ratio, linear time operation and algorithm simplicity. We provide a comparison of the performance of the algorithms when applied to virtual fracture terrains in Gran Canaria island.

Introduction

Local refinement can be useful when modeling hierarchical terrain meshes, in order to have a better representation of problems where the initial discretization is not the best, especially when more resolution is needed or there are non linear regions in the terrain [**1, 2, 3**]. The refinement problem can be defined as any technique that involves the insertion of at least one additional vertex in a mesh in order to produce more accurated meshes. Frequently, representations of the real world are not static because they change over time or depend on a prescribed observed terrain distance or a numerical solution within the mesh. This can be viewed as real-time terrain triangulations. If there is an alteration of the terrain, the mesh which represents it must consequently change and local refinement will be needed in order to have a resulting mesh which represents the area in a precise way.

A well-known triangle partition which has received much attention is the four triangles longest-edge subdivision (4T-LE); see [**1**] and the references therein. This partition scheme bisects a triangle into four subtriangles: the original triangle is first subdivided by its longest edge and then the two resulting triangles are bisected by joining the new midpoint of the longest edge to the midpoints of the remaining two edges of the original triangle. In this work, for the sake of comparison, we provided an implementation of 4T-LE following [**1**].

The longest-edge trisection algorithm (3T-LE), recently presentend in [**4**], offers a valid strategy to refine triangular meshes, but it has not been applied for the refinement of real terrain representations before. In this work, we provide a MATLAB implementation of the algorithm, in order to find out how it behaves when applied to real-time terrain triangulations in comparison to other refinement algorithms like 4T-LE and algorithms based on Delaunay triangulations. This will confirm that longest edge algorithms provide suitable representation tools for real-time terrain triangulations.

This work is organized as follows: first, the longest-edge trisection (3T-LE) refinement algorithm is described and implemented in MATLAB. Afterwards 3T-LE and 4T-LE algorithms are applied to a real-time terrain triangulation in order to compare their

performance. A further comparison of their performance and accuracy is made with a Delaunay triangulation.

1 3T-LE refinement algorithm

The longest-edge trisection of a triangle T (3T-LE) is obtained by connecting two equally spaced points, which trisect the longest edge of T, with the opposite vertex [**4**]. The trisection of a triangle has been used recently for the introduction of a new partition called the seven triangle longest-edge partition (7T-LE); see [**5**]. To assure that any adjacent elements share an entire edge or a common vertex (mesh conformity), the refinement is propagated after the subdivision of the target triangles. Once the longest edge of the triangle T is divided, the refinement is extended to the adjacent non conforming triangle, which is the neighbor of T by its longest edge. The process is repeated for all non conforming triangles. We refer to the set of these additional triangles as the propagation zone. The algorithm finishes once all triangles in the mesh are conforming.

The longest edge propagation path of a triangle is the ordered finite list of all adjacent triangles such that T_n is the longest edge neighbor triangle of T_{n+1}. Although it is possible to implement the algorithm by first dividing the target triangles and then propagating the refinement, in this work we will first find the longest edge propagation path and, after that, the triangles in this path will be trisected. Given a mesh of triangles $M(T)$, which contains a list of triangles to be refined, named S, the algorithm preforms as follows:

Require: a mesh of triangle $M(T)$, a list S of triangles of $M(T)$ to be refined
  for each triangle T_n of the list S **do**
    P = propagation path
    store T_n in P
    while P is not empty **do**
      extract T_n from P
      find longest edge of T_n
      find T_{n+1} = neighbor of T_n by LE
      if LE of T_n is LE of T_{n+1} or T_n is a boundary **then**
        equally divide LE in three by inserting points v_1 and v_2
        if LE of T_n is a boundary edge **then**
          join v_1 and v_2 with the opposite vertex of T_n
        else
          join v_1 and v_2 with the opposite vertex of T_n and T_{n+1}
        end if
      else
        store T_n in P
        store T_{n+1} in P
      end if
    end while
  end for

The algorithm starts by analyzing the first triangle T_n of the list S and finding its neighbor T_{n+1} by its longest edge. If the shared edge of both triangles is also the longest edge of T_{n+1}, both triangles are trisected by inserting points in the longest edge and joining them with the opposite vertex. Otherwise, T_n is stored in P, which is a LIFO

(last in, first out) queue, and the process is repeated for T_{n+1}. Only in case a couple of triangles which share their longest edge is found, or a boundary edge is reached, triangles are trisected. After that, the next triangle is extracted from P and the process is repeated for the extracted triangle. Once the LIFO stack P is empty, the refinement of T_n and its propagation path is finished, so that all triangles are conforming, thus we can repeat the refinement algorithm for the next triangle of the list S. Notice that, if a triangle is in the list S and it is also part of the propagation path of a triangle which occupies a previous position on that list, it will not be necessary to refine it, because its propagation path is part of the path of the previous triangle to be refined.

2 Real-time terrain application and conclusions

MATLAB implementations of 3T-LE and 4T-LE algorithms were applied to an example real-time triangulation of Gran Canaria island. Given a right tringulated irregular network (RTIN) terrain mesh sample, the refinement experiment is simulated by supposing there is an unexpected linear fracture in the terrain of Gran Canaria island. The algorithms were applied for local refinement on the triangles which represented areas of the terrain affected as the fracture was moving forward. Those triangles conform the list S of triangles to be refined. Results of the local refinement performed by 3T-LE can be observed in Figure 1.

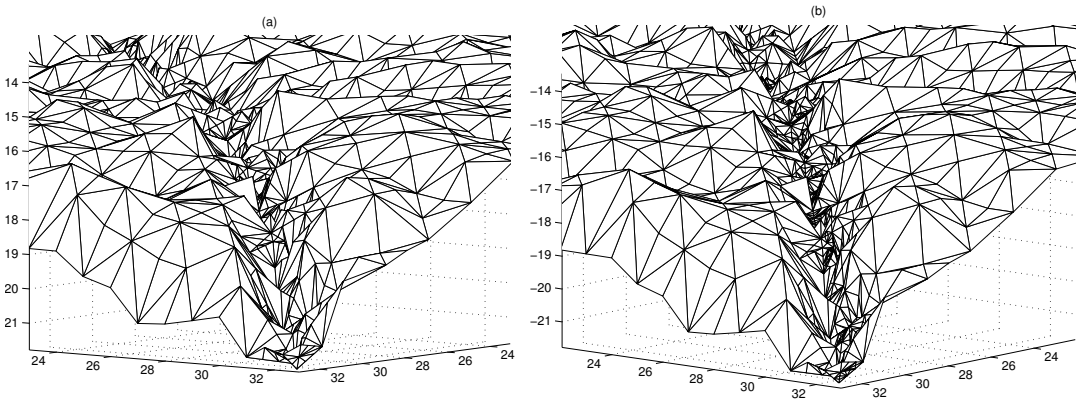


FIGURE 1. Refinement of a fracture with the 3T-LE algorithm. (a) 3D view of the refined fracture with one level of refinement. (b) 3D view with two levels of refinement.

A comparison of the performance of the algorithms is shown in Table 1. We applied the algorithms to the sample mesh and measured: the number of vertex of the resulting mesh (V), the execution time ($T(s)$) in an Intel Core 2 Duo 2.53 GHz processor with 4 GB RAM and the quality of the generated triangles (q), evaluated with the MATLAB function *pdetriq*( ), which states that a triangle is of acceptable quality if $q > 0.6$. Furthermore, we present the results observed when we simply insert points ir order to section the longest edge as prescribed by 3T-LE and 4T-LE and then make a Delaunay triangulation of the original and inserted points. In Tables 2 and 3 we show results of the consecutive application of the algorithms for two and three levels of refinement. It can be observed

Algorithm	V	$T(s)$	$mean(q)$
3T-LE	1466	1.4060	0.7728
3T-LE with Delaunay	1506	2.8120	0.8085
4T-LE	1209	2.4690	0.8591
4T-LE with Delaunay	1775	2.9690	0,8158

TABLE 1. Performance comparison with 1 level of refinement

Algorithm	V	$T(s)$	$mean(q)$
3T-LE	2300	5.0930	0.6876
3T-LE with Delaunay	2112	7.6720	0.7654
4T-LE	1563	6.3910	0.8382
4T-LE with Delaunay	2573	9.8130	0,7906

TABLE 2. Performance comparison with 2 levels of refinement

Algorithm	V	$T(s)$	$mean(q)$
3T-LE	3169	10.5480	0.6159
3T-LE with Delaunay	2744	14.7500	0.7380
4T-LE	1927	12.5630	0.8183
4T-LE with Delaunay	3389	13.2650	0,7659

TABLE 3. Performance comparison with 3 levels of refinement

that the application of 3T-LE and 4T-LE algorithms implemented in this work provides good results in terms of execution time and quality of the mesh when compared with 3T-LE and 4T-LE performed with a Delaunay triangulation. Other tests were executed on different terrain meshes leading to similar results as those presented in Tables 1–3.

Acknowledgements

This work has been supported in part by CICYT Project MTM2008-05866-C03-02/MTM from Ministerio de Educación y Ciencia of Spain.

References

[1] J. P. Suárez, A. Plaza, Four-triangles adaptive algorithms for RTIN terrain meshes, *Mathematical and Computer Modelling* **49(5)** (2009), 1012–1020.

[2] W. Evans, D. Kirkpatrick, D. Townsend, Right-triangulated irregular networks *Algorithmica: Special Issue on Algorithms for Geographical Information Systems* **30** (2001), 264–286.

[3] P. Lindstrom, D. Koller, W. Ribarsky, L. Hodeges, N. Faust, G. Turner, Real-time continuous level of detail rendering of height fields, *SIGGRAPH'96* (1996).

[4] A. Plaza, S. Falcón, J. P. Suárez, A local refinement algorithm for the longest-edge trisection of triangle meshes, Mathematics and Computers in Simulation, preprint, 2010.

[5] A. Márquez, A. Moreno-González, A. Plaza, J. P. Suárez, The 7-triangle longest-side partition of triangles and mesh quality improvement, *Finite Elements in Analysis and Design* **44(12-13)** (2008), 748–758.

A lower bound on the angles of triangles constructed by LE-trisection

Francisco Perdomo[1], **Ángel Plaza**[1], **Eduardo Quevedo**[2], **José P. Suárez**[3]

[1] Department of Mathematics, University of Las Palmas de Gran Canaria, Campus de Tafira, 35017 Las Palmas de Gran Canaria, Spain
aplaza@dmat.ulpgc.es

[2] Department of Electronic and Automatic Engineering, University of Las Palmas de Gran Canaria, Spain

[3] Department of Cartography and Graphic Engineering, University of Las Palmas de Gran Canaria, Spain

Abstract. The longest-edge (LE) trisection of a triangle Δ is obtained by joining the two equally spaced points of its longest-edge with the opposite vertex. Let $\alpha > 0$ be the smallest interior angle of Δ and α' the smallest angle of any triangle obtained after iteration of the LE-trisection. In this paper we prove that $\alpha' \geq \alpha/c$, where $c = (\pi/3)/(\arctan\left(\frac{\sqrt{3}}{11}\right))$.

Introduction

Mesh refinement has been an important research area in applied mathematics and engineering applications. For example, longest-edge bisection guarantees the construction of good-quality irregular and nested triangulations. The main reason is the lower bound condition on the small angles of the triangles so generated: It has proven to be critical to numerical convergence, for example in finite element method. So non-degeneracy of the involved mesh partitions has received much interest: from early works in the seventies [**1**] to last studies [**2, 3, 4**].

The longest-edge (LE) trisection of a triangle $t = t(A, B, C)$ is obtained by joining the two equally spaced points of the longest-edge of ABC with the opposite vertex. Figure 1(a) shows the LE-trisection of triangle t. The three new triangles generated will be named $t_L = t_L(A, D_1, C)$, $t_M = t_M(C, D_1, D_2)$ and $t_R = t_R(D_2, B, C)$, where the subscripts L, M, R stand for *left*, *medium* and *right* respectively. Repeated application of the partition generates a triangular mesh (Figure 1(b)).

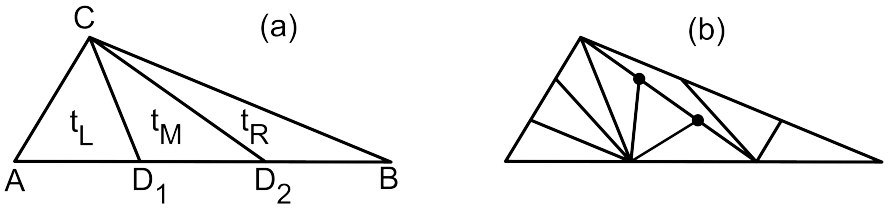


FIGURE 1. (a) LE-trisection of the triangle ABC. (b) 2nd iteration with hanging nodes.

We have proved in [**2**] that, for a given initial triangle with smallest interior angle $\alpha > 0$, the LE-trisection produces three new triangles such that any of their interior angles

[2]Partially supported by research grant SMCG-2730062011.

α_1 satisfies $\alpha_1 \geq \alpha/c_1$, where $c_1 = \frac{\pi/3}{\arctan(\sqrt{3}/5)} \approx 3.1403$. Besides, empirical evidence has been given of the non-degeneracy of the meshes or partitions obtained by iteration of the LE-trisection. In fact, if α is the minimum interior angle of the initial triangle, and α' is the minimum interior angle of any triangle in a mesh after iterated applications of LE-trisection, then $\alpha' \geq \alpha/6.7052025350$.

We have proved more recently in **[5]** a sharp upper bound of the diameters of triangles generated in LE-trisection method. It follows implicitly the non-degeneracy property of LE-trisection from our proof of this result. In this work we prove mathematically that $\alpha' \geq \alpha/c$, where $c = \frac{\pi/3}{\arctan\left(\frac{\sqrt{3}}{11}\right)}$. It confirms our previous numerical research in **[2]**.

1 LE-trisection and hyperbolic geometry

A method used in the literature of triangular mesh refinement is to normalize triangles **[6, 7]**. The normalization process consists in applying, possibly, several isometries and dilations to a triangle, matching its longest edge with the segment whose endpoints are $(0, 0)$ and $(1, 0)$, leaving its shortest edge to the left. In this way, all the similar triangles are represented by a complex number z in the *normalized region* $\{0 \leq \text{Re } z \leq 1/2, \text{Im } z \geq 0, |z - 1| \leq 1\}$, being z the opposite vertex to the longest edge of the triangle, once it has been normalized.

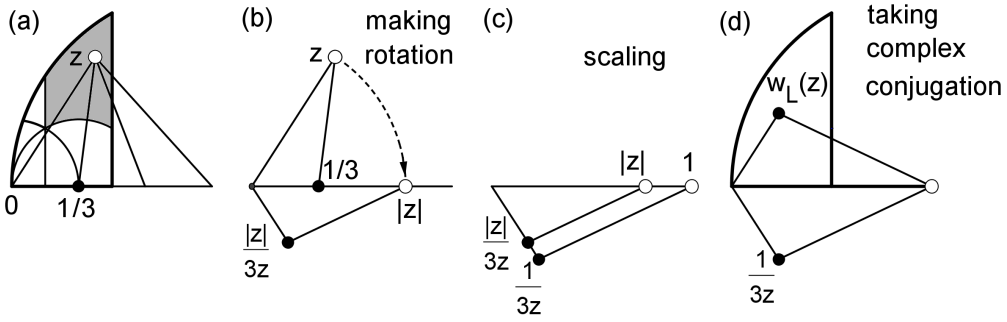


FIGURE 2. Transformations to obtain the expression of $w_L(z)$ for z in the gray zone.

Let z be a complex number in the normalized region which satisfies Re $z \geq 1/6$ and $|z - 1/3| \geq 1/3$; see Figure 2(a). The LE-trisection is applied to the triangle 0, 1 and z (we name a triangle with its vertices). Three triangles are performed joining z with $1/3$ and $2/3$. The normalization of the triangle 0, $1/3$ and z (resp. $1/3$, $2/3$ and z or $2/3$, 1 and z) defines the point $w_L(z)$ (resp. $w_M(z)$ or $w_R(z)$) in the normalized region. The geometric transformations used in the normalization of the triangle 0, $1/3$ and z are showed in Figure 2. For this case it is obtained $w_L(z) = \frac{1}{3\bar{z}}$.

In the same way we obtain the expressions which are given in Figure 3. Therefore, the normalization reduces the LE-trisection method to the complex dynamic in the normalized region associated to three complex functions w_L, w_M and w_R.

We use results of hyperbolic geometry and particularly the Poincaré half plane model **[8]**. The circumferences and the straight line which appear in the description of w_L, w_M and w_R are geodesics in the Poincaré half plane. The expressions which appear in w_L, w_M and w_R are isometries in the half plane hyperbolic model too. These functions have the additional following property.

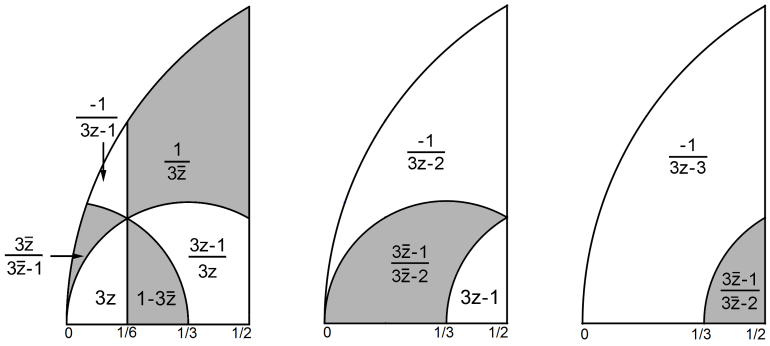


FIGURE 3. Left: function w_L. Middle: function w_M. Right: function w_R.

Lemma 1.1. *Let W be any of the functions w_L, w_M and w_R. If z_1 and z_2 are in the normalized region, then $d(W(z_1), W(z_2)) \leq d(z_1, z_2)$, where $d(\,\cdot\,,\,\cdot\,)$ denotes the hyperbolic distance in the half plane.*

Proof. First we mention that W is invariant under inversion with respect to the circumferences (or symmetry with respect the straight line) which appear in its description. For example, let $W = w_L$. The symmetry with respect to Re $z = 1/6$ is given by $z \mapsto \frac{1}{3} - \bar{z}$. This symmetry composed with $\frac{-1}{3z-1}$ (resp. $\frac{3\bar{z}}{3\bar{z}-1}$, $3z$) is $\frac{1}{3\bar{z}}$ (resp. $\frac{3z-1}{3z}$, $1-3\bar{z}$).

Now, if z_1 and z_2 are in a zone with same expression of W, then $d(W(z_1), W(z_2)) = d(z_1, z_2)$, because W is isometric in the half plane hyperbolic model. In any case, by the symmetries of W, there exist z_1' and z_2' in the normalized region with $W(z_1) = W(z_1')$ and $W(z_2) = W(z_2')$, z_1' and z_2' in a zone with same expression of W and, finally, with $d(z_1', z_2') \leq d(z_1, z_2)$. Thus the lemma follows. □

2 Closed for trisection and non-degeneracy property

A set Ω in the normalized region is *closed for trisection* (or *closed*) if, for all $z \in \Omega$, $w_L(z)$, $w_M(z)$ and $w_R(z)$ are in Ω. The union of the three hyperbolic circles C_1, C_2 and C_3 with radius $\ln\sqrt{2}$ and centers $\frac{1}{3}+\frac{\sqrt{2}}{3}i$, $\frac{1}{3}+\frac{\sqrt{2}}{6}i$ and $\frac{4}{9}+\frac{\sqrt{2}}{9}i$, respectively, form a closed region; see Figure 4(a). In fact, the set $\left\{\frac{1}{3}+\frac{\sqrt{2}}{3}i, \frac{1}{3}+\frac{\sqrt{2}}{6}i, \frac{4}{9}+\frac{\sqrt{2}}{9}i\right\}$ is closed; therefore the union of C_1, C_2 and C_3 is closed too by Lemma 1.1.

Let z be a complex number in the normalized region. We define Γ_z as the set of successive images of z by the functions w_L, w_M and w_R. If z is the complex associated to a triangle in the normalization process, the minimum angle α of the triangle is equal to the argument of $1-\bar{z}$. We denote by α' the argument of $1-\bar{z'}$ with $z' \in \Gamma_z$. We have to prove that α'/α has a lower bound. In our proof we split the normalized region in other closed regions where it is possible obtain a lower bound.

Let $z_{\text{eq}} = \frac{1}{2}+\frac{\sqrt{3}}{2}i$. Let Γ be the complex numbers in $\Gamma_{z_{\text{eq}}}$ exterior to C_1, C_2 and C_3. We consider the union of the circles C_1, C_2, C_3 and the circles with centers in Γ and radius $\ln\sqrt{3}$. The intersection of this union with the normalized region is a closed region Σ (see Figure 4(a)) with the following nice property.

Proposition 2.1. *If $z \in \Sigma$ and $z' \in \Gamma_z$, then $\alpha' \geq \alpha/c$, where $c = \dfrac{\pi/3}{\arctan\left(\frac{\sqrt{3}}{11}\right)} \approx 6.7052$.*

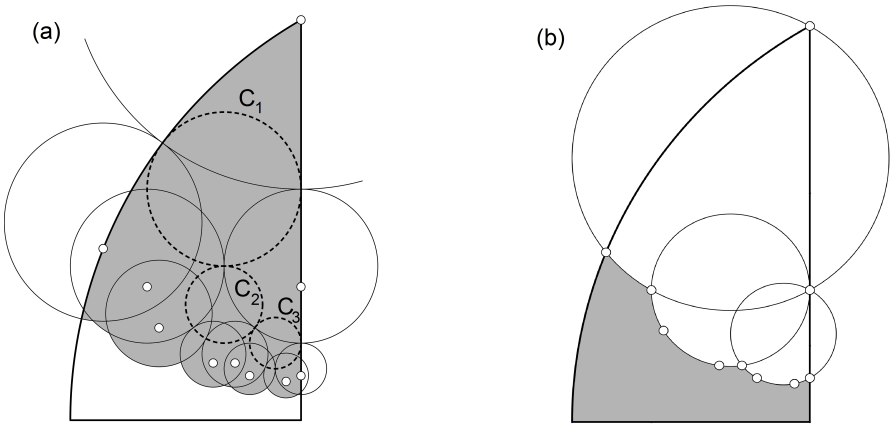


FIGURE 4. (a) The region Σ is shadowed. (b) The region Σ^* is shadowed.

Let Σ^* be the set of complex numbers in the normalized region under circumferences with centers $\frac{1}{3}+\frac{\sqrt{2}}{3}i$, $\frac{1}{3}+\frac{\sqrt{2}}{6}i$ y $\frac{4}{9}+\frac{\sqrt{2}}{9}i$ and radius $\ln\left(\frac{\sqrt{2}+\sqrt{6}}{2}\right)$ in the hyperbolic sense (see Figure 4(b)). The following proposition concludes our research.

Proposition 2.2. *If $z \in \Sigma^*$ and $z' \in \Gamma_z$, then $\alpha' \geq \alpha/c$, where* $c = \dfrac{\arctan\left(\frac{3\sqrt{3}}{13}\right)}{\arctan\left(\frac{\sqrt{3}}{12}\right)} \approx 2.6527.$

3 Conclusions

In this paper the non-degeneracy property of the LE-trisection method has been settled, as well as a sharp lower bound of the angles of triangles constructed by trisecting the longest-side. Hyperbolic geometry has been applied successfully to study the quality of the triangular mesh refinement based on LE-trisection.

References

[1] I. G. Rosenberg and F. Stenger, A lower bound on the angles of triangles constructed by bisecting the longest side, *Math. Comp.* **130** (1975), 390–395.

[2] A. Plaza, S. Falcón and J. P. Suárez, On the non-degeneracy property of the longest-edge trisection of triangles, *Appl. Math. Comput.* **216** (2010), 862–869.

[3] A. Hannukainen, S. Korotov and M. Křížeck, On global and local mesh refinements by a generalized conforming bisection algorithm, *J. Comp. Appl. Math.* **235** (2010), 419–436.

[4] M.-C. Rivara, N. Hitschfeld and B. Simpson, Terminal-edges Delaunay (small-angle based) algorithm for the quality triangulation problem, *Computer-Aided Design* **33** (2001), 263–277.

[5] F. Perdomo, E. Quevedo, A. Plaza and J. P. Suárez, A proof of convergence of the longest-edge trisection method for triangles, *Math. Comp. Simul.* (submitted April 2011).

[6] C. Gutiérrez, F. Gutiérrez and M. C. Rivara, Complexity of the bisection method, *Theor. Comp. Sci.* **382** (2007), 131–138.

[7] A. Plaza, J. P. Suárez and G. F. Carey, A geometric diagram and hybrid scheme for triangle subdivision, *Comp. Aid. Geom. Des.* **24** (2007), 19–27.

[8] G. Tóth, *Glimpses of Algebra and Geometry*, Undergraduate Texts in Mathematics – Readings in Mathematics, Springer-Verlag, New York, 2002.

Meshes preserving minimum feature size

Greg Aloupis[1], **Erik D. Demaine**[2], **Martin L. Demaine**[2], **Vida Dujmović**[3], **John Iacono**[4]

[1] Academia Sinica, Taipei, Taiwan
aloupis.greg@gmail.com

[2] Massachusetts Institute of Technology, Cambridge, Massachusetts, USA
{edemaine,mdemaine}@mit.edu

[3] Carleton University, Ottawa, Ontario, Canada
vida@cs.mcgill.ca

[4] Polytechnic Institute of New York University, Brooklyn, New York, USA
jiacono@poly.edu

Abstract. The *minimum feature size* of a planar straight-line graph is the minimum distance between a vertex and a nonincident edge. When a polygon is partitioned into a mesh, the *degradation* is the ratio of original to final minimum feature size. We show that some planar straight-line graphs cannot be triangulated with constant degradation, even with an unbounded number of Steiner points and triangles. This result answers a 14-year-old open problem by Bern, Dobkin, and Eppstein. For an n-vertex input, we obtain matching worst-case lower and upper bounds on degradation of $\Theta(\lg n)$. Our upper bound comes from a new meshing algorithm that uses $\mathcal{O}(n)$ triangles and $\mathcal{O}(n)$ Steiner points. If we allow triangles to have Steiner points along their sides, a construction is presented that achieves $\mathcal{O}(1)$ degradation.

Introduction

In this paper[5], we study the problem of polygon triangulation, with the possible aid of Steiner vertices. Our goal is to not introduce small distances between vertices and nonincident edges, compared to distances already existing in the shape. To compare the input and output, we use the *minimum feature size* of a planar straight-line graph G, denoted by $\mathrm{mfs}(G)$. This is the minimum distance between a vertex and a nonincident edge. We are interested in decomposing a polygon P into a planar straight-line graph (more specifically, a triangulation) G such that the minimum feature size of G is as close as possible to that of P. We call the ratio $\mathrm{mfs}(P)/\mathrm{mfs}(G)$ the *degradation* of the decomposition of P into G. Note that mfs does not distinguish between the interior and exterior of P when measuring distances.

Minimum feature size is a parameter well suited for describing the resolution needed to visually distinguish elements in a mesh. For example, it measures the maximum thickness that the edges in a mesh can be drawn. Also, mfs measures the amount of error allowed in the placement of vertices, so that a drawing preserves its topology. This could be useful in manufacturing, as well as in finite element simulation.

One important issue here is the type of desired triangulation. This choice has a large effect on the results that can be achieved; see Figure 1. The most common decomposition

[2]Research supported by NSF grants CCF-1018370, CCF-0430849 and by an Alfred P. Sloan fellowship. Research partially completed as a visiting professor at MADALGO (Center for Massive Data Algorithmics, a Center of the Danish National Research Foundation), Department of Computer Science, Aarhus University, IT Parken, Åbogade 34, DK-8200 Århus N, Denmark.

[5] The research presented in this paper was initiated at the 24th Annual Winter Workshop on Computational Geometry at the Bellairs Research Institute of McGill University, at which the presence of Ferran Hurtado was appreciated and is hereby acknowledged.

of a polygon is the *classic triangulation*, where noncrossing chords are added between vertices of P, until the interior of P is partitioned into triangles. If we allow Steiner points, a *proper triangulation* is such that any two edges that lie on the same interior face and are incident to a common vertex are not collinear. A *non-proper triangulation* simply partitions P into triangles, with no restrictions.

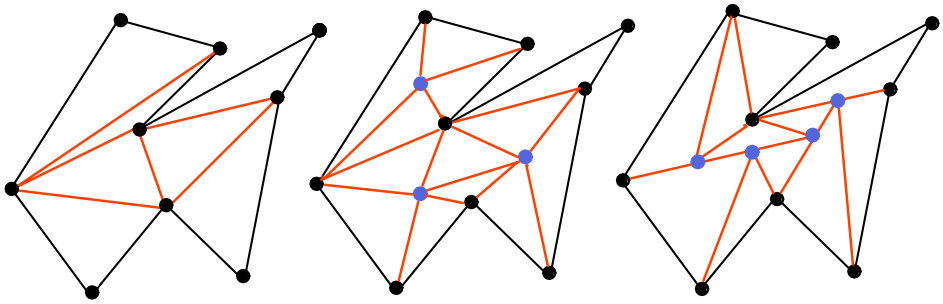


FIGURE 1. Triangulation types: classic, proper, non-proper. Steiner points are blue.

Bern, Dobkin, and Eppstein [**BDE95**] studied this problem, using the notion of *internal feature size* $\mathrm{ifs}(P)$, which is the minimum distance inside P between a vertex and a nonincident edge. (Note that "internal feature size" is called "minimum feature size" in [**BDE95**].) They proved that every polygon P (possibly with holes) has a non-proper triangulation in which every triangle has height $\Omega(\mathrm{ifs}(P))$.

For a planar straight-line graph, triangulating all of its faces with triangles that have height at least h is equivalent to guaranteeing that the triangulation itself has minimum internal feature size at least h. Notice that the internal feature size of a triangle equals its height. Thus $\mathrm{ifs}(P)$ is an upper bound on the smallest height of a triangle in any triangulation of P, so this bound is the best possible up to constant factors. However the method does not guarantee that the minimum feature size of the resulting triangulation is bounded by a function of $\mathrm{mfs}(P)$. Thus, partitioning both the inside and the outside of a polygon into triangles whose height is bounded by a function of $\mathrm{mfs}(P)$ is not guaranteed either. Consequently, the first open problem the authors list is whether their result can be generalized to planar straight-line graphs, that is, whether such graphs can be triangulated while preserving their minimum feature size.

We answer this open problem negatively. Specifically, we provide a simple polygon G such that every proper triangulation of G has degradation $\Omega(\lg n)$, independent of the number of Steiner points and triangles. We match this lower bound by providing an algorithm for properly triangulating any given planar straight-line graph G so that degradation is $\mathcal{O}(\lg n)$. Our algorithm uses $\mathcal{O}(n)$ Steiner points and hence $\mathcal{O}(n)$ triangles. Steiner points are necessary to obtain a degradation smaller than a linear factor; Bern and Eppstein [**BE95**] showed that all classic triangulations of a regular n-gon have a minimum feature size degradation of $\Omega(n)$. This can be extended trivially to quadrangles or any decomposition with constant size faces.

Until now, no meshing algorithm with a constant degradation was known. Ruppert's Delaunay mesh refinement algorithm claims such a bound [**Rup93**, Theorem 1], but the constant actually depends on the minimum angle of the input graph (as well as the minimum triangle angle guaranteed by the algorithm).

What causes the need for logarithmic degradation in proper triangulations of planar straight-line graphs? We show that the essential issue is forbidding Steiner points along

the sides of a triangle. However, by allowing Steiner points along the sides of triangular elements (what we call *non-proper triangulation*), $\mathcal{O}(1)$ degradation is actually possible.

1 Non-proper triangulations can preserve minimum feature size

In this section we show how to construct a non-proper triangulation for any polygon P, such that the minimum feature size degradation is $\Theta(1)$. We use $\Theta(n)$ Steiner points, and the construction can be computed in linear time.

We provide a brief overview of our construction. There are two distinct regions of P which will be triangulated separately. The two regions will be separated by a polygon Q interior to P whose boundary remains at distance $\Theta(\text{mfs})$ to that of P. The region between P and Q is called the *tube*. The algorithm places all Steiner points on or interior to Q; none are placed on P.

The polygon Q is constructed to have the following properties with respect to absolute constants c_1, c_2, c_3: (1) All points on Q are at most c_1 away from the closest point on P, and at least c_2 away from the nearest point on P. (2) All vertices on Q have y-coordinates which are multiples of c_3 (i.e., they are on a c_3 horizontal grid). (3) There are $\mathcal{O}(n)$ vertices (initially) on Q. The details of how to find such a Q are omitted, but we note that the *grassfire* transformation plays a vital part in the construction.

Next, an edge from each vertex of Q is added to the closest vertex on P; now the region between the P and Q is subdivided into triangles and quadrangles. The interior of Q is then quadrangulated by performing a trapezoidal decomposition of the interior of the tube; this will introduce new Steiner vertices on Q. However, since all vertices on Q are at least c_3 separated, this does not cause any issues with respect to minimum feature size. The decomposition at this point contains triangles and quadrangles; the quadrangles need to be triangulated. The most difficult cases are the quadrangles in the tube; these will have one edge from P and one edge from Q. The edge from Q may have many Steiner vertices introduced by the trapezoidal decomposition, separated by a distance of at least c_3. The edge from P does not have any Steiner vertices, and it cannot be assumed that it is safe to put any Steiner vertices on P because there could be another edge of P arbitrary close. For example, see Figure 2 (D), where a constant fraction of the boundary of P is off-limits to the introduction of Steiner points). The other two edges connecting P and Q do not have Steiner points. Figure 2 (A) shows how to triangulate a rectangle subject to these restrictions while maintaining constant minimum feature size; this construction can be slightly modified to decompose any of the needed quadrangles and complete the construction, yielding the following theorem.

Theorem 1.1. *Every polygon has a non-proper triangulation with constant minimum feature size degradation.*

2 Degradation upper bound for proper triangulations

The decomposition method is the same as for non-proper triangulations, with the exception that the decomposition of quadrangles shown in Figure 2 (A) can not be used as it is a non-proper triangulation. Instead the construction of Figure 2 (C) is used; this yields $\mathcal{O}(\log n)$ degradation.

Theorem 2.1. *Every n-vertex polygon has a proper triangulation with $\mathcal{O}(\log n)$ minimum feature size degradation.*

3 Lower bounds

Due to lack of space and complexity of proofs, we only state our main results of this section.

Let P_n be the generalized version of the $n+6$ vertex polygon illustrated in Figure 2 (D). This polygon has width 2, height $n+1$, and minimum feature size of 1. Let R_n be the rectangular region of P_n shaded in Figure 2 (D). A *τ-grid* is a set of τ vertical lines. Let a *τ-grid triangulation* of P_n be a non-proper triangulation of P_n where all Steiner vertices are on a τ-grid.

Theorem 3.1. *For every τ and n, every τ-grid (non-proper) triangulation G of P_n has degradation*

$$\Omega\left(\min\left\{\frac{\lg n}{\lg\lg n}, \frac{\lg n}{\lg \tau}\right\}\right).$$

When considering proper triangulations, our (omitted) proof for Theorem 3.1 simplifies to a $\Omega\left(\frac{\lg n}{\lg\lg n}\right)$ bound on degradation. However, we are able to improve as follows.

Theorem 3.2. *For every n, every proper triangulation G of P_n has degradation $\Omega(\lg n)$.*

This can be shown to extend to a bound of $\Omega(\lg_r n)$ for proper r-rangulations.

References

[BDE95] Marshall Bern, David Dobkin, and David Eppstein. Triangulating polygons without large angles. *International Journal of Computational Geometry & Applications*, 5(1–2):171–192, March–June 1995.

[BE95] Marshall Bern and David Eppstein. Mesh generation and optimal triangulation. In Ding-Zhu Du and Frank Kwang-Ming Hwang, editors, *Computing in Euclidean Geometry*, number 4 in Lecture Notes Series on Computing, pages 47–123. World Scientific, second edition, 1995.

[Rup93] Jim Ruppert. A new and simple algorithm for quality 2-dimensional mesh generation. In *Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 83–92, Austin, Texas, 1993.

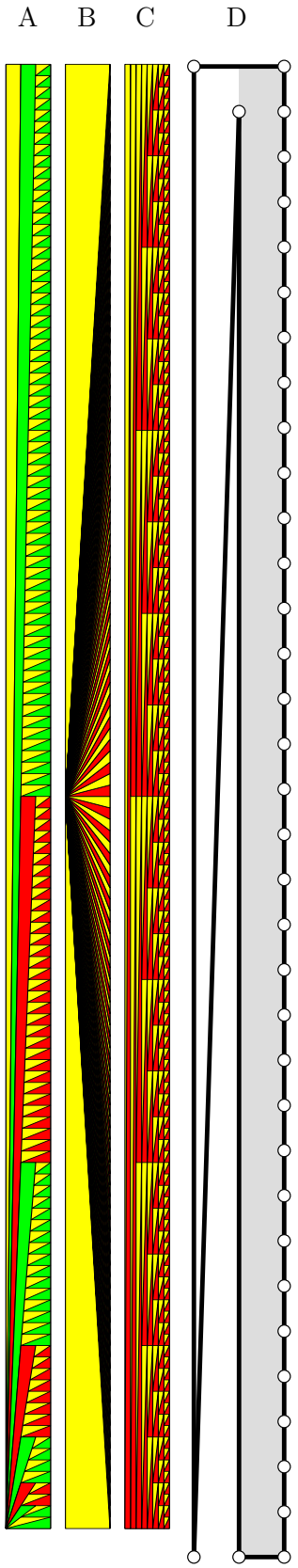


FIGURE 2. (A)–(C) are triangulations of the same rectangle with 129 vertices on the right edge. (A) is a non-proper triangulation, with $\mathcal{O}(1)$ degradation. For comparison, (B) is a simple proper fan triangulation with $\mathcal{O}(n)$ degradation. (C) is a proper triangulation with $\mathcal{O}(\log n)$ degradation. Observe how for (A) the interiors of all the triangles are clearly visible; in (C) it is more difficult to discern the individual triangles, and in (B) it is impossible. (D) is a polygon that shows Steiner vertices cannot be placed on a significant fraction of the boundary close to the reflex vertex.

Heavy non-crossing increasing paths and matchings on point sets

Toshinori Sakai[1], **Jorge Urrutia**[2]

[1] Research Institute of Educational Development, Tokai University, Japan
sakai@tokai-u.jp

[2] Instituto de Matemáticas, Universidad Nacional Autónoma de México, México
urrutia@matem.unam.mx

Abstract. Let S be a permutation of $I_n = \{1, \ldots, n\}$. The weight of a subsequence of S is the sum of its elements. We prove that any permutation S of I_n always contains an increasing or a decreasing subsequence of weight greater than $n\sqrt{n/3}$; our bound is asymptotically tight. We also show that S contains a *unimodal* subsequence of weight at least $n\sqrt{2n/3} - O(n)$. Our problem arises in the following geometric setting: Let P be a set of n points whose elements are labelled with the integers in I_n. A simple path of P is an increasing path if when we traverse it starting at one of its endpoints, the labels of its elements always increase. The weight of a path is the sum of the labels of its elements. We study the problem of finding simple increasing paths with large weight. We also study the problem of finding non-crossing matchings of P with large weight, where the weight of an edge with endpoints $i, j \in P$ is $\min\{i, j\}$, and the weight of a matching is the sum of the weights of its edges.

Introduction

Consider any permutation S of $I_n = \{1, \ldots, n\}$. A well-known result of Erdős and Szekeres [3] asserts that any permutation of I_n always contains an increasing or a decreasing subsequence with at least $\lceil \sqrt{n} \rceil$ elements. The weight of a subsequence of S is the sum of its elements. If we consider the permutation $S = \{5, 2, 8, 1, 7, 4, 3, 6\}$, the weight of the increasing subsequence $\{2, 4, 6\}$ is equal to 12. Among all the increasing or decreasing subsequences of S, the one with maximum weight is $\{8, 7, 6\}$, with weight 21. In this paper we study the problem of finding the increasing or decreasing subsequence of a permutation with maximum weight. We prove that any permutation of I_n always contains an increasing or a decreasing subsequence with weight greater than $n\sqrt{n/3}$; our bound is asymptotically tight. The permutations obtained to solve this problem produce efficient packings of squares with areas $1, 2^2, \ldots, n^2$. We also study the problem of finding unimodal subsequences of large weight of permutations of I_n. We show that any permutation of I_n always has a unimodal subsequence of weight at least $n\sqrt{2n/3} - O(n)$.

Our results are motivated by the following problem: Let P be a set of n points on the plane in general position such that its elements are labelled with the integers of I_n. Different elements of P receive different labels (we call this point set a labelled point set, or simply a point set). A path $\mathcal{W}$ whose vertices are elements of P is called *simple* if no two of its edges cross each other. $\mathcal{W}$ is called an *increasing path* if when we traverse it starting at one of its endpoints, the labels of its vertices always increase. The weight of a path is the sum of the labels of its vertices. Finding increasing or decreasing subsequences in permutations of I_n allows us to establish bounds on the weight of the heaviest simple increasing path in labelled point sets. For point sets in convex position, we use unimodal

[2]Partially supported by projects MTM2006-03909 (Spain) and SEP-CONACYT 80268 (Mexico).

or anti-unimodal subsequences of large weight. We also study the problem of finding non-crossing matchings of labelled point sets with large weight, where the weight of the edge joining i and j is the smaller of $\{i, j\}$, and the weight of a matching is the sum of the weights of its edges. We show that a point set in convex position always has a matching of weight at least $n^2/5 - O(n)$. Point sets in general position always have matchings with weight at least $n^2/6 + \Omega(n)$.

Finding structures in point sets on the plane that optimize some given functions has been of interest to many computational geometers for some time. Problems studied so far include finding simple paths, matchings, cycles, and trees of maximum length; see Dumitrescu and Tóth [1]. Pach, Károlyi, and Tóth [5] show that if the edges of a complete geometric graph on $k^2 + 1$ points are colored red or blue, then there always exists a simple red or blue path of length $k + 1$. The problem of finding long simple increasing paths was first studied by Czyzowicz *et al.* [4]. They proved that any labelled point set in convex position contains a simple increasing path of length at least $\sqrt{2n}$. This bound was improved recently by Sakai and Urrutia [6] to $\sqrt{3n - 3/4} - 1/2$.

1 Heavy simple increasing paths

1.1 Heavy increasing subsequences of a permutation

Let $S = \{s(1), \ldots, s(n)\}$ be a permutation of I_n. To each $s(i)$ of S, we associate the point (x_i, y_i) as follows: x_i is the weight of the heaviest increasing subsequence of S ending at $s(i)$, and y_i is the weight of the heaviest decreasing subsequence of S starting at $s(i)$. If $S = \{4, 3, 7, 2, 5, 1, 6\}$, then we associate to $s(3) = 7$ the point $(x_3, y_3) = (4+7, 7+5+1) = (11, 13)$. We also associate to each $s(i)$ the square $SQ(i)$ whose top right vertex is (x_i, y_i) and whose bottom left vertex is the point $(x_i - s(i), y_i - s(i))$.

Observation If $i \neq j$, then $(x_i, y_i) \neq (x_j, y_j)$, and $SQ(i)$ and $SQ(j)$ have disjoint interiors.

Let α be the minimum value such that the square $\mathcal{SQ}$ with vertices $(0, 0), (0, \alpha), (\alpha, \alpha), (\alpha, 0)$ contains all $SQ(i)$. Since the area of $\mathcal{SQ}$ must be at least the total area of the $SQ(i)$, we must have $\alpha > n\sqrt{\frac{n}{3}}$. This implies:

Theorem 1.1. *Any permutation of I_n contains an increasing or a decreasing subsequence whose weight is greater than $n\sqrt{n/3}$. Our bound is asymptotically tight.*

To see that our bound is asymptotically tight, let $k = \sqrt[4]{4n^3/3}$, and $m = \sqrt{3n}/2$. Consider now the following permutation Π:

$$\lceil k \rceil, \lceil k \rceil - 1, \ldots, 1,\ \lceil \sqrt{2}k \rceil, \lceil \sqrt{2}k \rceil - 1, \ldots, \lceil k \rceil + 1,$$
$$\lceil \sqrt{3}k \rceil, \lceil \sqrt{3}k \rceil - 1, \ldots, \lceil \sqrt{2}k \rceil + 1,\ \ldots,\ n, n-1, \lceil \sqrt{m-1}\, k \rceil + 1.$$

Thus, Π consists of m blocks of decreasing integers such that, for each block, the sum of its elements is $n\sqrt{n/3} + O(n)$. On the other hand, the heaviest increasing subsequence of Π is the subsequence containing the elements $\lceil k \rceil, \lceil \sqrt{2}\, k \rceil, \lceil \sqrt{3}\, k \rceil, \ldots, n$, which again has weight $n\sqrt{n/3} + O(n)$. As a consequence, we have:

Theorem 1.2. *Any labelled point set with n elements has a simple increasing path of weight greater than $n\sqrt{n/3}$.*

1.2 Heavy increasing paths of point sets in convex position

A subsequence $\{s(i_1), \ldots, s(i_k)\}$ of a permutation S of I_n is called *unimodal* if there is a j, $1 \leq j \leq k$, such that $s(i_1) < \cdots < s(i_j) > \cdots > s(i_k)$; it is called *anti-unimodal* if for some j, $s(i_1) > \cdots > s(i_j) < \cdots < s(i_k)$. The next result is given without a proof.

Theorem 1.3. *Any permutation of I_n contains a unimodal subsequence of weight greater than $n\sqrt{2n/3} - O(n)$.*

Observe that the problem of finding a simple increasing path of maximum weight for a labelled point set in convex position can be reduced to that of finding a unimodal or anti-unimodal sequence of maximum weight in a permutation of I_n obtained from P by reading its elements starting at a suitable point of P. Thus the next result follows:

Theorem 1.4. *Any labelled point set in convex position has a simple increasing path of weight greater than $n\sqrt{2n/3} - O(n)$.*

The best upper bound we have for the weight of a unimodal or an anti-unimodal subsequence of a permutation is approximately $2n\sqrt{n/3}$, and is given by the permutation used in Theorem 1.1. On the other hand, the best upper bound we have for the weight of a simple increasing path of n labelled points in convex position is approximately $n\sqrt{2n}$. This is given by the following permutation Π' with $n = 2k^2$:

$$\begin{array}{llll} n-k+1, & n-2k+1, & \ldots, & k+1, \quad 1, \\ n-k+2, & n-2k+2, & \ldots, & k+2, \quad 2, \\ & \cdots & & \\ n=2k^2, & n-k, & \ldots, & 2k, \quad k. \end{array}$$

It is easy to see that the maximum weight unimodal or anti-unimodal subsequence of Π' has weight $\approx n\sqrt{2n}$; this weight is achieved by the subsequence

$$n-k+1, n-k+2, \ldots, n, n-k, \ldots, 2k, k.$$

2 Heavy non-crossing matchings

In this section, we study the problem of finding non-crossing matchings of a labelled point set that maximize the sum of the weights of its edges. We give lower bounds of the sums of the weights while efficient upper bounds are still open.

2.1 Point sets in convex position

Lemma 2.1. *The weight of any non-crossing perfect matching of a point set P in general position with $2m$ elements is at least $\binom{m+1}{2}$ and at most m^2. These bounds are tight.*

To prove the tightness of the lower bound, let P be an unlabelled point set with $2m$ elements in *convex* position. Color the elements of P red or blue in such a way that when we traverse the boundary of the convex hull of P the colors of its elements alternate. Observe that any edge of a perfect matching $\mathcal{M}$ of P joins a red and a blue point. Label the red and blue points of P with the integers $\{1, \ldots, m\}$ and $\{m+1, \ldots, 2m\}$, respectively, not necessarily in order. Then the weight of any edge of $\mathcal{M}$ belongs to $\{1, \ldots, m\}$. Since different edges in $\mathcal{M}$ have different weights, the weight of $\mathcal{M}$ is precisely $\binom{m+1}{2}$. The tightness of the upper bound is easy to prove.

Using similar arguments, we can prove:

Lemma 2.2. *Let P be a set of points in general position whose elements have been labelled with the set of integers $\{2r+1, 2r+2, \ldots, 2m\}$. Then the weights of perfect matchings of P have the following bounds, and these bounds are tight:*

- *at least* $(2r+1)+(2r+2)+\cdots+[2r+(m-r)] = 2r(m-r)+\binom{m-r+1}{2}$, *and*
- *at most* $(2r+1)+(2r+3)+\cdots+(2m-1) = 2r(m-r)+(m-r)^2$.

Next we consider matchings that are not necessarily perfect.

Lemma 2.3. *Let P be a point set in convex position such that some of its elements are colored red and the rest blue. Then there is a non-crossing matching $\mathcal{M}$ of P that matches all, but at most two, elements of P such that the endpoints of each edge of $\mathcal{M}$ have the same color.*

We are now ready to prove the main result of this section.

Theorem 2.4. *Let P be a labelled point set in convex position. Then the heaviest non-crossing matching of P has weight at least $n^2/5 - O(n)$.*

Proof. To make our proof easy to understand, let us assume that P has $n = 10s$ elements. Discard from P all the elements with labels in $\{1, \ldots, 2s\}$. Now color with blue and red all the elements of P with labels in $\{2s+1, \ldots, 6s\}$ and $\{6s+1, \ldots, 10s\}$, respectively. By Lemma 2.3, we can find matchings $\mathcal{M}'$ and $\mathcal{M}''$ of $\{2s+1, \ldots, 6s\}$ and $\{6s+1, \ldots, 10s\}$, respectively, that leave at most two elements of $\{2s+1, \ldots, 10s\}$ unmatched.

Suppose first that all the elements of $\{2s+1, \ldots, 10s\}$ are matched. By Lemma 2.2 with $r = s$, and $m = 3s$, the weight of $\mathcal{M}'$ is at least $(2s+1)+(2s+2)+\cdots+(4s-1)+4s$. Similarly, by applying Lemma 2.2 with $r = 3s$ and $m = 5s$, the weight of $\mathcal{M}''$ is at least $(6s+1)+(6s+2)+\cdots+(8s-1)+8s$. It is easy to see now that the sum of the weights of $\mathcal{M}'$ and $\mathcal{M}''$ is at least $n(n+1)/5$.

Observe now that if two elements of $\{2s+1, \ldots, 10s\}$ are unmatched, then the sum of the weights of $\mathcal{M}'$ and $\mathcal{M}''$ decreases from $n(n+1)/5$ by at most $2n$. Hence the weight of $\mathcal{M}' \cup \mathcal{M}''$ is at least $n^2/5 - O(n)$. □

2.2 Point sets in general position

For point sets in general position, by discarding the elements with labels smaller than $n/3 + O(1)$ and applying Lemma 2.2, we obtain:

Theorem 2.5. *Let P be a labelled point set in general position. Then the heaviest non-crossing matching of P has weight at least $n^2/6 + \Omega(n)$.*

References

[1] A. Dumitrescu and C. Tóth, Long non-crossing configurations in the plane, *Discrete Comp. Geom.* **44** (2010), 727–752.

[2] A. Dumitrescu and R. Kaye, Matching colored points in the plane: Some new results, *Computational Geometry* **19** (2001), 69–85.

[3] P. Erdős and G. Szekeres, A combinatorial problem in geometry, *Compositio Math.* **2** (1935), 463–470.

[4] J. Czyzowicz, E. Kranakis, D. Krizanc, and J. Urrutia, Maximal length common non-intersecting paths, *Proc. Eighth Canadian Conference on Computational Geometry*, August 1996, Ottawa, Canada, 1996, 180–189.

[5] J. Pach, G. Károlyi, and G. Tóth, Ramsey-type results for geometric graphs I, *Discrete and Computational Geometry* **18** (1997), 247–255.

[6] T. Sakai and J. Urrutia, Monotonic polygons and paths of weighted point sets. Manuscript, submitted, 2011.

Covering islands in plane point sets

Ruy Fabila-Monroy[1,3], **Clemens Huemer**[2,3]

[1] Departamento de Matemáticas, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, México
ruyfabila@math.cinvestav.edu.mx

[2] Departament de Matemàtica Aplicada IV, Universitat Politècnica de Catalunya
clemens.huemer@upc.edu

Abstract. Let S be a set of n points in general position in the plane. A k-island I of S is a subset of k points of S such that $\text{Conv}(I) \cap S = I$. We show that, for an arbitrary but fixed number $k \geq 2$, the minimum number of k-islands among all sets S of n points is $\Theta(n^2)$. The following related counting problem is also studied: For $l < k$, an l-island covers a k-island if it is contained in the k-island. Let $C_{k,l}(S)$ be the minimum number of l-islands needed to cover all the k-islands of S and let $C_{k,l}(n)$ be the minimum of $C_{k,l}(S)$ among all sets S of n points. We find asymptotic bounds for $C_{k,l}(n)$.

Introduction

Let S be a set of n points in general position in the plane. An island I of S is a subset of S such that the convex hull of I does not contain points of $S \backslash I$. Problems related to islands have been studied recently in **[4, 5]**. A k-island is an island I with $|I| = k$. The set of k-islands of S will be denoted as $I_k(S)$. Thus $I_2(S)$ are the $\binom{n}{2}$ segments connecting pairs of points of S and $I_3(S)$ are the empty triangles of S. Katchalski and Meir **[13]** proved that the minimum number of empty triangles among all sets of n points is $\Theta(n^2)$. Lower and upper bounds for the exact leading constant of the quadratic term have then been improved in **[2, 3, 8, 16]**. We show that also for $k > 3$ the minimum of $|I_k(S)|$ among all sets S of n points is $\Theta(n^2)$. Note that, for $k > 3$, a k-island might contain points in the interior of its convex hull. The special case when all the points of a k-island form a convex polygon, that is to say, they form an empty convex k-gon in the point set, has been studied extensively. For $k = 4$, the minimum number of empty convex k-gons is quadratic **[3]**. For $k = 5$ and $k = 6$ a linear lower bound is known **[9, 10, 14, 17]**. A construction of n points without empty convex heptagons is due to Horton **[11]**. We will see that Horton sets also provide $O(n^2)$-examples for the minimum number of k-islands.

We then study coverings of k-islands. Some problems related to coverings were studied in **[1, 15]**. For $l < k$, an l-island *covers* a k-island if it is contained in the k-island. Let $C_{k,l}(S)$ be the minimum number of l-islands needed to cover $I_k(S)$. Let $C_{k,l}(n)$ be the minimum of $C_{k,l}(S)$ among all sets S of n points in general position in the plane. The problem to determine $C_{k,l}(n)$ is closely related to counting empty convex polygons in point sets. For example, since every set of ten points contains an empty convex pentagon **[10]**, $C_{10,5}(n)$ is at most the minimum number of empty convex pentagons among all sets of n points. Also, the particular case $C_{3,2}(n)$ has been studied recently in **[6]** by considering the question: What is the maximum number of edges that a geometric graph on a point set S can have such that it does not contain empty triangles? There it is shown that $n - 2 + \lfloor \frac{n}{8} \rfloor \leq C_{3,2}(n) \leq O(n \log n)$. The latter bound is achieved by the Horton set. We

[3]Research partially supported by projects MEC MTM2009-07242 and Gen. Cat. DGR 2009SGR1040.

value of l	$\leq \lfloor \frac{k+5}{6} \rfloor$	$\leq \lceil \frac{k}{2} \rceil$	$\geq \lceil \frac{k}{2} \rceil + 1$
$C_{k,l}(n)$	$\Theta(n)$	$O(n \log n)$	$\Omega(n \log n)$

TABLE 1. The obtained bounds for $C_{k,l}(n)$.

will see that also for each $l \leq \lceil \frac{k}{2} \rceil$ the Horton set provides the upper bound $O(n \log n)$ on $C_{k,l}(n)$. Table 1 summarizes our obtained bounds for $C_{k,l}(n)$.

Throughout the text, $k \geq 2$ is an arbitrary but fixed natural number, and n is arbitrarily large with respect to k.

1 Islands of the Horton set

A Horton set, see e.g. **[2, 3, 7, 11]**, is defined recursively as follows: $H(1) = \{(1, 1)\}$ and $H(2) = \{(1, 1), (2, 2)\}$. When $H(n)$ is defined, set

$$H(2n) = \{(2x - 1, y) \mid (x, y) \in H(n)\} \cup \{(2x, y + 3^n) \mid (x, y) \in H(n)\}.$$

We denote the subset of points of a Horton set $H(n)$ with even x-coordinate as the upper set $H^+(n)$, and subset of $H(n)$ with odd x-coordinate as the lower set $H^-(n)$. Horton sets have the following property: Any line connecting two points from $H^+(n)$ leaves all points from $H^-(n)$ below, and any line connecting two points from $H^-(n)$ leaves all points from $H^+(n)$ above.

We say that an island I of a point set S is *open from above* if the vertical stripe bounded by the leftmost point and rightmost point of I contains no points of $S \setminus I$ above I. Likewise I is *open from below* if there are no points of $S \setminus I$ below I in this stripe. Denote with $I_k^+(S)$ and $I_k^-(S)$ the set of k-islands of S open from above and open from below, respectively.

We estimate the number $|I_k^+(H(n))|$ of k-islands of the Horton set $H(n)$ open from above. The following lemma also applies to the number $|I_k^-(H(n))|$ of k-islands open from below. The proof is omitted due to lack of space.

Lemma 1.1. *There exist positive constants c_k and c'_k that only depend on k such that $c'_k n \leq |I_k^+(H(n))| \leq c_k n$.*

Lemma 1.2. *The number of k-islands of the Horton set $H(n)$ is $\Theta(n^2)$.*

Proof. The k-islands that are contained entirely in the upper set $H(n)^+$ or in the lower set $H(n)^-$ can be counted recursively. A k-island that has points in both of $H(n)^+$ and $H(n)^-$ consists of an i-island open from below in $H(n)^+$ and a $(k - i)$-island open from above in $H(n)^-$, for some $i \in \{1, \ldots, k - 1\}$. We thus obtain the recurrence for the number $|I_k(H(n))|$ of k-islands of the Horton set:

$$|I_k(H(n))| = 2|I_k(H(\tfrac{n}{2}))| + \textstyle\sum_{i=1}^{k-1} |I_i^-(H(\tfrac{n}{2}))| \cdot |I_{k-i}^+(H(\tfrac{n}{2}))|.$$

Since k is a constant, $|I_i^-(H(\frac{n}{2})|$ and $|I_{k-i}^+(H(\frac{n}{2}))|$ are both in $\Theta(n)$ by Lemma 1.1. Thus,

$$|I_k(H(n))| = 2|I_k(H(\tfrac{n}{2}))| + \Theta(n^2).$$

It follows that $|I_k(H(n))|$ is $\Theta(n^2)$. □

Lemma 1.3. *Let S be a set of n points in general position in the plane, and let $k \geq 2$. Then $|I_k(S)| = \Omega(n^2)$.*

Proof. For a point $p \in S$, sort the points of $S\backslash\{p\}$ cyclically around p. Divide these points into $\Omega(n)$ groups of $k-1$ consecutive points in this order. Each group together with p forms a k-island. Repeating this for every point of S, we count $\Omega(n^2)$ k-islands; each one is counted at most k times. □

Lemmas 1.2 and 1.3 imply the following result:

Theorem 1.4. *The minimum number of k-islands among all sets of n points in general position is $\Theta(n^2)$.*

2 Covering islands

We now show asymptotic bounds for $C_{k,l}(n)$.

Theorem 2.1. *For $l \leq \lfloor \frac{k+5}{6} \rfloor$, $C_{k,l}(n)$ is $\Theta(n)$. For $l \leq \lceil \frac{k}{2} \rceil$, $C_{k,l}(n)$ is $O(n \log n)$. For $l \geq \lceil \frac{k}{2} \rceil + 1$, $C_{k,l}(n)$ is $\Omega(n \log n)$.*

The proof is split into the following three cases.

- For $l \leq \lfloor \frac{k+5}{6} \rfloor$, there exist sets of $n = ml$ points, for any m, such that their k-islands can be covered with m l-islands.

Proof. The construction starts with a Horton set of m points; if m is not a power of 2 then take a larger Horton set and only consider its m leftmost points. Then $l-1$ additional points are placed in an ε-neighborhood of each point of the Horton set, where $\varepsilon > 0$ is chosen small enough. It remains to provide a covering of all k-islands of this point set with m l-islands: Choose each point of the given Horton set together with its $l-1$ additional nearest points as an l-island of the covering. Thus, m l-islands are chosen. Assume the point set has a k-island I that is not covered by some chosen l-island. Note that $k \geq 6(l-1)+1$. Also note that I contains at most $l-1$ points of each chosen l-island. We now use the fact that Horton sets have no empty convex heptagons. Therefore, I cannot have points from more than six l-islands. But since $k > 6(l-1)$, this gives a contradiction. □

- For $l \leq \lceil \frac{k}{2} \rceil$, the k-islands of the Horton set $H(n)$ can be covered with $O(n \log n)$ l-islands.

Proof. To cover all the k-islands, choose all the l-islands open from below of $H(n)^+$ and all the l-islands open from above of $H(n)^-$. By Lemma 1.1, the number of these l-islands is $O(n)$. Moreover, these l-islands cover all the k-islands that have points in both $H(n)^+$ and $H(n)^-$. Recursively cover the k-islands of $H(n)^+$ and $H(n)^-$. Let $T(n)$ denote the number of l-islands in this covering. We obtain the recurrence:

$$T(n) \leq 2T(\tfrac{n}{2}) + O(n),$$

which gives $O(n \log(n))$ as an upper bound. □

- For $l \geq \lceil \frac{k}{2} \rceil + 1$ and for every set S of n points, at least $\Omega(n \log n)$ l-islands are needed to cover all the k-islands of S.

Proof. Consider a halving line ℓ_h for S. We want to find $\Omega(n)$ pairwise disjoint k-islands crossing ℓ_h such that at most $\lceil \frac{k}{2} \rceil$ points of each island are on each side of ℓ_h. But this follows easily as a special case of the Equitable Subdivision Theorem [**12**]: assume the points of S on one side of ℓ_h are colored red; the remaining ones are colored blue. Possibly

ignore the leftmost and rightmost points of S (up to $\Theta(n)$ points) so that there are $g\lceil\frac{k}{2}\rceil$ red points and $g\lfloor\frac{k}{2}\rfloor$ blue points, where g is $\Theta(n)$. Then, there exists a subdivision of the plane into g disjoint convex polygons such that each of them contains exactly $\lceil\frac{k}{2}\rceil$ red points and $\lfloor\frac{k}{2}\rfloor$ blue points. Each such polygon gives a k-island. We thus have $\Omega(n)$ pairwise disjoint k-islands crossing ℓ_h and therefore need $\Omega(n)$ pairwise disjoint l-islands to cover these k-islands. Since $l \geq \lceil k/2\rceil + 1$, also each l-island crosses ℓ_h. We iterate on both sides of ℓ_h and obtain the following recurrence, which gives the claimed lower bound for $C_{k,l}(n)$ of $\Omega(n \log n)$:

$$C_{k,l}(n) \geq 2C_{k,l}(\tfrac{n}{2}) + \Omega(n). \qquad \square$$

We finally determine $C_{k,l}(H(n))$, for $l > \lceil\frac{k}{2}\rceil$. Based on the following lemma, whose proof is omitted, we believe that, for $l > \lceil\frac{k}{2}\rceil$, $C_{k,l}(n)$ is $\Theta(n^2)$.

Lemma 2.2. *For $l \geq \lceil\frac{k}{2}\rceil + 1$, the number of l-islands needed to cover the k-islands of the Horton set $H(n)$ is $\Theta(n^2)$.*

References

[1] B. Aronov, M. Dulieu, F. Hurtado, Witness (Delaunay) graphs, *Comput. Geom.* **44** (2011), 329–344.

[2] I. Bárány, Z. Füredi, Empty simplices in Euclidean space, *Canad. Math. Bull.* **30** (1987), 436–445.

[3] I. Bárány, P. Valtr, Planar point sets with a small number of empty convex polygons, *Stud. Sci. Math. Hung.* **41** (2004), 243–269.

[4] C. Bautista-Santiago, J. Cano, R. Fabila-Monroy, D. Flores-Peñaloza, H. González-Aguilar, D. Lara, E. Sarmiento, J. Urrutia, On the diameter of a geometric Johnson type graph, *26th European Workshop on Computational Geometry* (2010), 61–64.

[5] C. Bautista-Santiago, J. M. Díaz-Báñez, D. Lara, P. Pérez-Lantero, J. Urrutia, I. Ventura, Computing maximal islands, *25th European Workshop on Computational Geometry* (2009), 333–336.

[6] C. Bautista-Santiago, M. A. Heredia, C. Huemer, A. Ramírez-Vigueras, C. Seara, J. Urrutia, On the number of edges in geometric graphs without empty triangles, submitted to journal, January 2011.

[7] O. Devillers, F. Hurtado, G. Károlyi, C. Seara, Chromatic variants of the Erdős–Szekeres Theorem, *Comput. Geom.* **26** (2003), 193–208.

[8] A. Dumitrescu, Planar sets with few empty convex polygons, *Stud. Sci. Math. Hung.* **36** (2000), 93–109.

[9] T. Gerken, Empty convex hexagons in planar point sets, *Discrete Comput. Geom.* **39** (2008), 239–272.

[10] H. Harborth, Konvexe Fünfecke in ebenen Punktmengen, *Elem. Math.* **33** (1978), 116–118.

[11] J. D. Horton, Sets with no empty convex 7-gons, *Canad. Math. Bull.* **26** (1983), 482–484.

[12] A. Kaneko, M. Kano, Discrete geometry on red and blue points in the plane – a survey, in: *Algorithms Combin.* **25**, Springer, 2003, 551–570.

[13] M. Katchalski, A. Meir, On empty triangles determined by points in the plane, *Acta Math. Hung.* **51** (1988), 323–328.

[14] C. M. Nicolás, The empty hexagon theorem, *Discrete Comput. Geom.* **38** (2007), 389–397.

[15] T. Sakai, J. Urrutia, Covering the convex quadrilaterals of point sets, *Graphs Combin.* **23** (2007), 343–357.

[16] P. Valtr, On the minimum number of empty polygons in planar point sets, *Stud. Sci. Math. Hungar.* **30** (1995), 155–163.

[17] P. Valtr, On empty hexagons, in: *Surveys on Discrete and Computational Geometry, Twenty Years Later*, AMS, 2008, 433–441.

Separated matchings in convex point sets with small discrepancy

Viola Mészáros[1]

[1] University of Szeged, Bolyai Institute, Aradi vértanúk tere 1, 6720 Szeged, Hungary
viola@math.u-szeged.hu

Abstract. Consider $2n$ points in the plane in convex position, where n points are red and n points are blue. Edges are straight line segments connecting points of different color. A separated matching is a geometrically non-crossing matching where all edges can be crossed by a line. Separated matchings are closely related to non-crossing, alternating paths. Abellanas et al. and independently Kynčl et al. constructed convex point sets allowing at most $\frac{4}{3}n+O(\sqrt{n})$ points on any non-crossing, alternating path. We present a coloring with constant discrepancy parameter where the number of points in the maximum separated matching is very close to $\frac{4}{3}n$. When the dicrepancy is at most three we show that there are at least $\frac{4}{3}n$ points in the maximum separated matching.

Introduction

Consider a $2n$-element point set with a balanced coloring (n points red and n points blue) in the plane. Edges will be straight line segments connecting points of different color. Erdős posed the following problem: How many points are there on the longest non-crossing, alternating path in an arbitrary balanced $2n$-element convex point set in the plane? Without loss of generality we may assume that the points are on a circle C.

Erdős constructed a convex point set that allows at most $\frac{3n}{2}+2$ points on the longest non-crossing, alternating path. He conjectured that his configuration was asymptotically extremal. Erdős' conjecture was disproved. Kynčl, Pach and Tóth gave a single construction in 2008 [**7**]. They showed the $\frac{4}{3}n+O(\sqrt{n})$ upper and the $n+\Omega(\sqrt{n/\log n})$ lower bound. Abellanas et al. found a similar construction independently of the previously mentioned researchers [**2**]. It is conjectured that the presented upper bound is asymptotically tight. Hajnal and Mészáros improved the lower bound to $n+\Omega(\sqrt{n})$ and gave a class of configurations for the $\frac{4}{3}n+O(\sqrt{n})$ upper bound [**5**].

In the non-convex version of the problem you may find results in the following papers: [**1**], [**3**], [**4**] and [**6**].

The proof techniques introduced the notion of *separated matchings*, that is, geometrically non-crossing matchings where all edges can be crossed by a single line. In some sense, separated matchings form a building element to alternating paths, as each separated matching can be easily completed to a non-crossing, alternating path when the points are in convex position.

An advantage of separated matchings is that we may consider point sets with small discrepancy. We say that the *discrepancy* is d if on any interval on the circle C the difference between the cardinality of color classes is at most d.

Small discrepancy coloring draws attention to the separated matching conjecture [**7**], that is formulated as follows. Let $2k$ denote the number of alternations between the two

[1]Partially supported by OTKA Grant K76099 and by the Centre Interfacultaire Bernoulli at EPFL in Lausanne.

colors in a $2n$-element point set on C. 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.

So far no one was concerned with the discrepancy parameter, since small discrepancy means many alternations among the two colors and this alone guarantees a long noncrossing, alternating path [**5**]. However, when we consider separated matchings, it is reasonable to investigate this case. We believe it might shed light on the difficulties of the original Erdős problem.

We will present a coloring where the discrepancy parameter is constant and the number of points in the maximum separated matching is very close to the conjectured value. Furthermore, if we restrict the discrepancy, we obtain an interesting result. For discrepancies two and three, we show that there are at least $\frac{4}{3}n$ points in the maximum separated matching. This result suggests that the order of magnitude in the separated matching conjecture is feasible. All these results can be found in [**8**] in detail, together with other related results in this area.

When the discrepancy is relatively small, the truth might be much closer to $2n$ than $\frac{4}{3}n$. Although the case of small discrepancy looks very promising, unfortunately already the analysis of discrepancy three is rather long by the current techniques. New ideas could yield further interesting results on small discrepancy colorings.

1 Coloring

First we introduce some necessary definitions to describe our coloring on C. Let our $2n$-element convex point set with a balanced coloring be denoted by P. An *arc* is an interval of points on $C \cap P$. The *size* of an arc is the number of its elements. In an arc the points are ordered —we always read the order in clockwise direction. A *run* is a maximal set of consecutive points on C of the same color. The *length* of a run is the number of its elements.

The previous configurations contained long runs colored red or blue and at most two arcs consisting of alternating short runs of the two colors. We will present a coloring with arbitrary many arcs of alternating short runs. The idea originates from the Kynčl–Pach–Tóth construction. We cut that construction into two pieces. We repeat the two pieces in arbitrary order an equal number of times along the circle.

We describe two special arcs called *blocks*. They will be the building elements of our configuration. The *bluish* block will consist of a red run of length s and a blue run of length $2s$. We denote the bluish block by $(s, 2s)$ block. The *reddish* block will consist of a red run of length s followed by a *mixed* arc M. The mixed arc M consists of $2s$ points alternating in color. Hence, the reddish block will contain $2s$ red and s blue points. We denote the reddish block by $(s, s(1,1))$ block.

The construction is a class of coloring $\mathcal{C}(s,t)$: Take t many $(s, 2s)$ blocks and t many $(s, s(1,1))$ blocks in arbitrary order along C. In other words, the same number of bluish and reddish blocks are placed along the circle in an arbitrary order.

2 Results

In this section we describe the results. First we take such a coloring from the previously given class where the discrepancy parameter is a large constant. Then we will estimate the number of points in the maximum separated matching in that coloring. Later we will recall a general theorem about our class of coloring $\mathcal{C}(s,t)$.

Before we proceed to our claims, we need to introduce another definition. Let the *size* of a separated matching be the number of points participating in it. Thus, it is twice the number of edges in the matching.

Now we are ready to make our statements.

Observation 2.1. *Let C_2 be that coloring from $\mathcal{C}(1000, t)$ where the reddish and bluish blocks alternate. Then the size of the largest separated matching in C_2 is at most* $1.34n$.

Consider the following theorem in [**8**]:

Let C_1 be any coloring from $\mathcal{C}(s, t)$. Then the size of every separated matching in C_1 is at most $\frac{4}{3}n + O(s + t)$.

In this theorem, we have $O(s + t)$ as the remainder term. We can choose s and t so that $s, t = O(\sqrt{n})$ and the order of magnitude of $O(s + t)$ becomes negligible. This is how the reader should think about this theorem.

Observation 2.1 is a special case of this general theorem described above. We choose a setting where s is a large constant and t is $\epsilon \cdot n$. So $O(s+t)$ is very small. The reason for choosing such a setting is that in C_2 the discrepancy of the coloring is constant (2000). At the same time the size of the optimal matching is very close to the conjectured value.

Now we will present our results on small discrepancy colorings.

Theorem 2.2. *For any coloring with dicrepancy at most three there is a separated matching of size at least $\frac{4n}{3}$.*

Proof. The colored point set can be viewed as follows: for each red point take a unit *up* line segment and for each blue point a unit *down* line segment. (When the discrepancy is one, then these up and down segments alternate.)

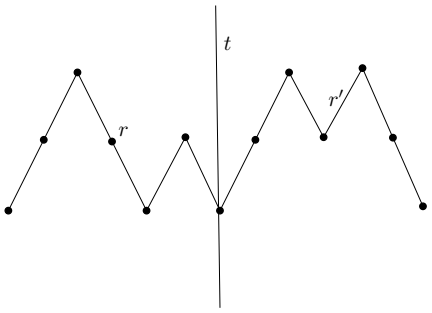


FIGURE 1. When the discrepancy is two, at most $\frac{1}{3}$ of the points will not participate in the constructed separated matching.

When the discrepancy is two, we will not choose a good axe that divides our point set. We can be given any axe that halves the number of runs and we will construct a separated matching of the desired size.

There are two types of runs regarding their length: runs of length 1 and runs of length 2. Each run contains at most two up and at most two down segments; see Figure 1. Let us take a drawing for any case of discrepancy two and halve the number of runs by taking an axe t. Then we pair up all the runs. The run r will have pair run r' if r and r' are on different sides of t but for the same distance to t regarding the number of runs. We make the separated matching S so that each run will face only its pair in the

matching. Therefore, all runs of length 1 will be fully covered in S. Now consider the runs of length 2. If a run r of length 2 faces a run r' of length 1, then $\frac{2}{3}$ of all the vertices of r and r' will be in S. Otherwise, the run r is also fully covered in S. Hence, it follows that when the discrepancy is two, there exists a separated matching of size at least $\frac{4n}{3}$.

The case of discrepancy three uses similar ideas but it includes a more sophisticated pairing of the runs. As it is quite extensive, we omit it from this extended abstract. □

3 Closing thoughts

Small discrepancy colorings have an importance in trying to achieve a better lower bound for the problem of long non-crossing, alternating paths. Although at separated matchings the truth might be much closer to $2n$ when the discrepancy parameter is small, by our current methods it is not easy to reveal it. For discrepancies at most three, a pairing algorithm of intervals yields that there are at least $\frac{4n}{3}$ points in the maximum separated matching. It would be feasible to improve this result by new ideas. Also the case of the subsequent relatively small discrepancies seems promising. Just by our current methods it gets rather extensive.

Small discrepancy colorings also suggest the separated matching conjecture stated in the Introduction. A more appealing version of this conjecture was formulated in [**5**]:

Conjecture 3.1. *Every balanced coloring of* $2n$ *points on* C *admits a separated matching of size* $\frac{4}{3}n + o(n)$.

Regarding the remainder term, even $O(\sqrt{n})$ is feasible. It would be an interesting result to settle this conjecture in the affirmative. That would also prove the conjecture related to the upper bound for non-crossing, alternating paths.

References

[1] M. Abellanas, J. García, G. Hernández, M. Noy, P. Ramos, Bipartite embeddings of trees in the plane, *Discrete Appl. Math.* **93** (1999), 141–148.

[2] M. Abellanas, J. García, F. Hurtado, J. Tejel, Caminos alternantes (in Spanish), *Proc. X Encuentros de Geometría Computacional*, Sevilla, June 2003, pp. 7–12. English version available on Ferran Hurtado's web page.

[3] J. Cibulka, J. Kynčl, V. Mészáros, R. Stolař, P. Valtr, Hamiltonian alternating paths on bicolored double-chains, in: I. G. Tollis, M. Patrignani (eds.), Graph Drawing 2008, *Lecture Notes in Computer Science* **5417**, Springer, New York, (2009), pp. 181–192.

[4] J. Cibulka, J. Kynčl, V. Mészáros, R. Stolař, P. Valtr, Universal sets for straight-line embeddings of bicolored graphs, submitted to the *Geometric Graph Theory* volume that János Pach is editing for Springer.

[5] P. Hajnal, V. Mészáros, Note on noncrossing alternating path in colored convex sets, accepted to *Discrete Mathematics and Theoretical Computer Science*.

[6] A. Kaneko, M. Kano, Discrete geometry on red and blue points in the plane – a survey, in: B. Aronov et al. (eds.), Discrete and Computational Geometry, The Goodman–Pollack Festschrift, Springer, *Algorithms Comb.* **25** (2003), 551–570.

[7] J. Kynčl, J. Pach, G. Tóth, Long alternating paths in bicolored point sets, *Discrete Mathematics* **308**, 19 (2008), 4315–4321.

[8] V. Mészáros, Separated matchings in colored convex sets, preprint, 2011.

Convex blocking and partial orders on the plane

J. M. Díaz-Báñez[1], M. A. Heredia[2], C. Peláez[2], J. A. Sellarès[3], J. Urrutia[4], I. Ventura[1]

[1] Departamento de Matemática Aplicada II, Universidad de Sevilla, Spain
dbanez@us.es, iventura@us.es

[2] Posgrado en Ciencia e Ingeniería de la Computación, Universidad Nacional Autónoma de México, Mexico
marco@ciencias.unam.mx, canek@ciencias.unam.mx

[3] Institut d'Informàtica i Matemàtica Aplicada, Universitat de Girona, Spain
sellares@ima.udg.es

[4] Instituto de Matemáticas, Universidad Nacional Autónoma de México, Mexico
urrutia@matem.unam.mx

Abstract. Let $C = \{c_1, \ldots, c_n\}$ be a collection of disjoint closed convex sets in the plane. Suppose that one of them, say c_1, represents a valuable object we want to uncover. We are allowed to pick a direction $\alpha \in [0, 2\pi)$ along which we can translate (remove) the elements of C one at a time while avoiding collisions. In this paper we solve the problem of finding the direction α_0 that minimizes the number of elements of C that have to be removed before we can reach c_1, in $O(n^2 \log n)$ time.

Introduction

Consider a set $C = \{c_1, \ldots, c_n\}$ of pairwise disjoint closed convex sets, and a direction $\alpha \in [0, 2\pi)$; e.g., the vertical upwards direction. It is well known that the elements of C can be translated (removed) one at a time by moving them upwards while avoiding collisions with other elements of C **[4, 6]**. Suppose that c_1 is a special object that we want to uncover, and that we are allowed to choose a direction α along which we can remove the elements of C one at a time while avoiding collisions.

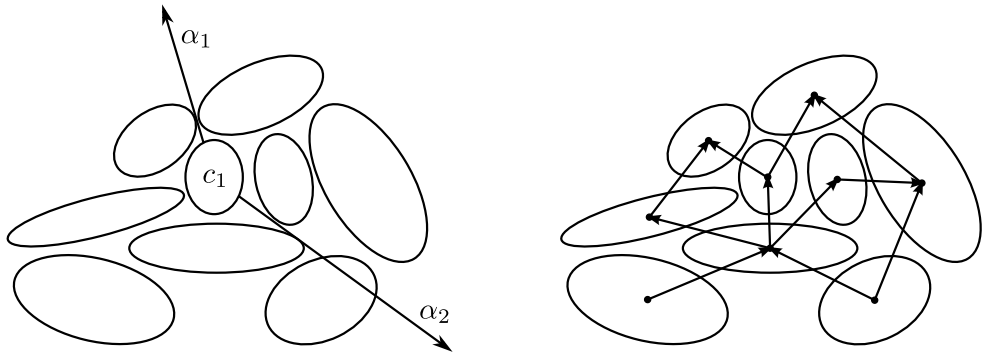
We want to find the direction α_0 that minimizes the number of elements of C that have to be removed before we can remove c_1 itself. For example, in Figure 1(a) for α_2 four elements of C have to be removed, while for α_1 we only need to remove two.

This problem can be seen as a variant of the problem known in computational geometry as the "separability problem" **[1, 2, 5]**. It is also related to *spherical orders* determined by light obstructions **[3]**.

1 Preliminaries

Given $c_i, c_j \in C$ and a direction α, we say that c_j is an α-cover of c_i if any directed line segment with direction α, starting at a point in c_i and ending at a point in c_j, does not intersect any other set in C. Observe that, if c_j is an α-cover of c_i, then c_i is an

[1]Partially supported by Spanish Government under MEC Project MTM2009-08652.
[2]Partially supported by CONACYT of Mexico.
[3]Partially supported by the Spanish MCI grant TIN2010-20590-C02-02.
[4]Partially supported by SEP-CONACYT of Mexico, Proyecto 80268, and by Spanish Government under MEC Project MTM2009-08652.



(a) A set C of convex sets. (b) Truncated lattice of C for $\alpha = \pi/2$.

FIGURE 1. A set of convex sets and its truncated lattice.

$(\alpha + \pi)$-cover of c_j. We say that c_j blocks c_i in the direction α, written as $c_j \succ_\alpha c_i$, if there is a sequence $c_{\sigma(1)} = c_i, c_{\sigma(2)}, \ldots, c_{\sigma(k)} = c_j$ of elements of C such that $c_{\sigma(i+1)}$ is an α-cover of $c_{\sigma(i)}$, $i = 1, \ldots, k-1$.

For each α, the blocking relation $\succ_\alpha$ is a partial order on C, which is a *truncated planar lattice* [**6**]. An ordered set is called a *lattice* if any two elements have a unique supremum and infimum. A lattice is called *planar* if its Hasse diagram can be drawn without intersecting edges. Finally, a finite order P is called a *truncated lattice* if, by adding to P both a least and a greatest element, the resulting order is a lattice.

The planar diagram of such a truncated lattice has the elements of C as vertices in which two elements c_i and c_j are joined by an arc oriented from c_i to c_j if c_j is an α-cover of c_i (Figure 1(b)). The elements of C that we need to remove in the direction α before reaching an element c_i of C are those convex sets c_j such that $c_j \succ_\alpha c_i$. Such a set is usually called the upper set of c_i in $\succ_\alpha$, or for short, the *up-set* of c_i.

Lemma 1.1. *Let c_i and c_j be two convex sets in C. The set of directions in which c_j blocks c_i forms a unique non-empty interval $\mathcal{I}_{i,j}$. The endpoints of any such $\mathcal{I}_{i,j}$ are directions determined by lines tangents to pairs of elements of C.*

Then, there are at most $4\binom{n}{2}$ combinatorially distinct values of α where the truncated lattice changes. These changes occur in slopes determined by lines tangent to pairs of elements of C. The search space for α_0 is reduced then to the set $\mathcal{D} = \{\gamma_1, \ldots, \gamma_{4\binom{n}{2}}\}$ of these directions. For the sake of clarity, we suppose that no two internal tangents are parallel and that the elements of $\mathcal{D}$ are ordered such that $\gamma_i < \gamma_j$ if $i < j$.

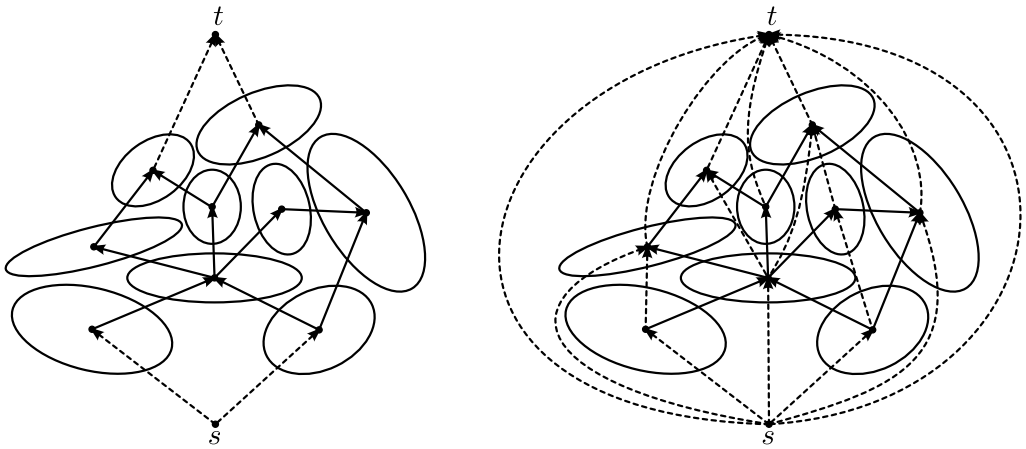
2 The transitive triangulation

Our problem can be trivially solved by calculating the truncated lattice for every direction in $\mathcal{D}$, obtaining the up-set of c_1 in each one, and selecting the γ_i with the smallest up-set. Since calculating the truncated lattice has a cost of $O(n \log n)$ time for each of the $4\binom{n}{2}$ directions, this yields an $O(n^3 \log n)$ time algorithm.

To improve this complexity, we calculate the truncated lattice only for the first direction in $\mathcal{D}$, and for each γ_i we update a data structure containing the truncated lattice for γ_{i-1} in constant time, with $i > 1$.

For each direction $\alpha \in [0, 2\pi)$, we complete the truncated lattice for $\succ_\alpha$ to a lattice by adding two special vertices, a source s and a sink t. These vertices will be such that for each maximal element c_i of the relation $\succ_\alpha$, we have $t \succ_\alpha c_i$, and for each minimal c_j we have $c_j \succ_\alpha s$. For a fixed direction we can picture t as a very large convex set standing above all of C, and s as a very large convex set standing below all of C (Figure 2(a)).

For each α, we now extend such a lattice to a triangulation $\mathcal{T}_\alpha$, which we will call the *α-transitive triangulation*, by adding oriented arcs (compatible with $\succ_\alpha$) between pairs of elements of C which are α-visible; see Figure 2(b).



(a) The lattice of C for $\alpha = \pi/2$. (b) Transitive triangulation $\mathcal{T}_\alpha$.

FIGURE 2. Complete lattice and its corresponding transitive triangulation.

By Lemma 1.1, there are at most $4\binom{n}{2}$ such lattices, and we want to know how $\mathcal{T}_\alpha$ changes as α goes from γ_i to γ_{i+1}. This leads to the following lemma:

Lemma 2.1. *Given the transitive triangulation $\mathcal{T}_{\gamma_i}$, the transitive triangulation $\mathcal{T}_{\gamma_{i+1}}$ can be obtained by* flipping an arc *in $\mathcal{T}_{\gamma_i}$. Moreover, such an arc flip either adds or removes an arc between the convex sets c_j and c_k that define γ_{i+1}.*

We omit the proof of Lemma 2.1 because of space restrictions, but an illustration of this fact is shown in Figure 3. Note that the arc flip can be done preserving transitivity.

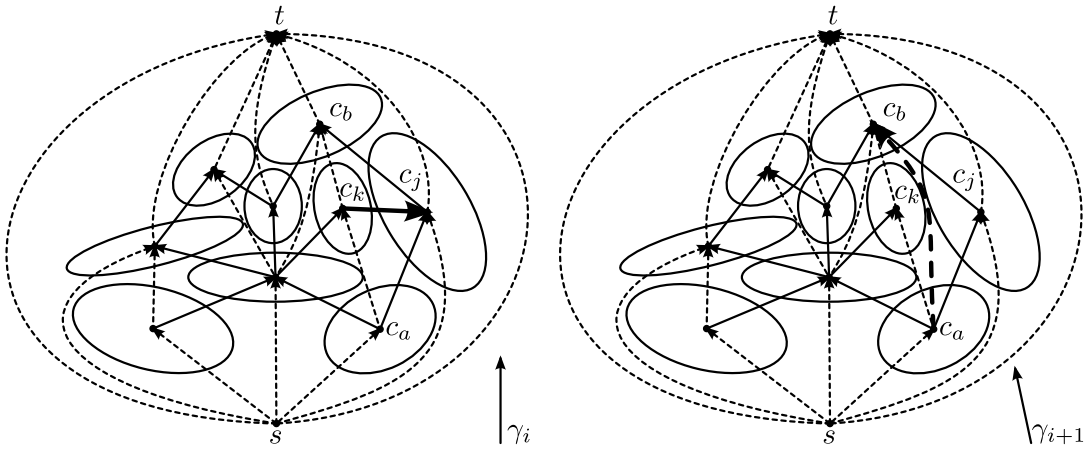
3 An $O(n^2 \log n)$ time algorithm to find α_0

Theorem 3.1. *A direction α_0 minimizing the up-set of c_1 can be computed in $O(n^2 \log n)$.*

The proof of this theorem uses the following results, given without proof:

Lemma 3.2. *As we rotate from the direction γ_1 to $\gamma_{4\binom{n}{2}}$, the up-set of c_i changes at most a linear number of times.*

Lemma 3.3. *Given $c_j, c_k \in C$, we can answer the query of whether c_j is in the up-set of c_k in $O(\log n)$ time.*



(a) The arc $c_k \to c_j$ before flipping. (b) The arc $c_a \to c_b$ after flipping.

FIGURE 3. An example of an arc flip: The arc $c_a \to c_b$ replaces $c_k \to c_j$ as c_j no longer blocks c_k in the γ_{i+1} direction. Note that $c_a \to c_b$ preserves transitivity.

The set $\mathcal{D}$ can be calculated in $O(n^2 \log n)$, if we suppose that the internal tangents between any two convex sets in C can be determined in constant time. For each γ_i we store the indexes j, k of the convex sets (c_j and c_k) that determine it. After computing the up-set of c_1 for γ_1 (in $O(n \log n)$ time using Lemma 3.3), as we move from γ_i to γ_{i+1}, we can update the up-set of c_1 as follows: Let k and j as defined above for γ_i. There are three types of events that can arise when moving from γ_{i-1} to γ_i:

(1) c_k and c_j are not in the up-set of c_1 in the direction γ_{i-1}. In this case the up-set of c_1 remains unchanged.
(2) c_k and c_j belong to the up-set of c_1 for γ_{i-1}. If c_k and c_j become comparable, the up-set of c_1 remains. Suppose that c_k and c_j become uncomparable. It can be proved that by checking a constant number of the neighbors of each of c_k and c_j, we can verify whether they remain in the up-set of c_1. If they do, then the up-set of c_1 remains. If not, then we recalculate the up-set of c_1 ($O(n \log n)$).
(3) A similar process happens when exactly one of c_k and c_j is in the up-set of c_1.

By Lemma 3.2, we have to update the up-set of c_1 at most a linear number of times, and thus the whole process takes $O(n^2 \log n)$ time. This proves Theorem 3.1.

References

[1] N. G. de Bruijn, Aufgaben 17 and 18 (in Dutch), *Nieuw Archief voor Wiskunde* **2** (1954), 67.
[2] L. Fejes-Tóth and A. Heppes, Über stabile Körpersysteme (in German), *Compositio Mathematica* **15**(2) (1963), 119–126.
[3] S. Foldes, I. Rival and J. Urrutia, Light sources, obstructions and spherical orders, *Discrete Mathematics* **102**(1) (1992), 13–23.
[4] L. J. Guibas and F. F. Yao, On translating a set of rectangles, in: *Proceedings of the Twelfth Annual ACM Symposium on Theory of Computing*, 1980, 154–160.
[5] G. Toussaint, Movable separability of sets, in: *Computational Geometry*, North-Holland, Amsterdam, 1985, 335–375.
[6] I. Rival and J. Urrutia, Representing orders on the plane by translating convex figures, *Order* **4**(4) (1988), 319–339.

Generalizing the Steiner-Lehmus theorem using the Gröbner cover

Antonio Montes[1], **Tomás Recio**[2]

[1] Universitat Politècnica de Catalunya, Spain
antonio.montes@upc.edu

[2] Universidad de Cantabria, Spain
tomas.recio@unican.es

Abstract. The notion of Gröbner cover has recently been introduced [**7**], yielding most precise and compact information about parametric polynomial systems of equations. The purpose of our contribution here is to exemplify, through a challenging generalization of the Steiner–Lehmus theorem [**4**], how this tool can be applied to the automatic discovery of geometry theorems.

Introduction

This extended abstract is about a particular role of computations in geometry: those leading to the automatic discovery of the necessary and sufficient conditions that are required for a (perhaps false) geometric statement to become true.

The theorem of Steiner–Lehmus states that if a triangle has two (internal) angle-bisectors with the same length, then the triangle must be isosceles (the converse is, obviously, also true). This is an issue which has attracted along the years a considerable interest, and we refer to [**10**] for a large collection of references and comments on this classical statement and its proof. More recently, its generalization, regarding internal as well as external angle bisectors, has been approached through automatic tools, cf. [**1**], [**8**] or [**9**]. The goal is to find a similar statement concerning triangles verifying the equality of two bisectors (of whatever kind) for different vertices. This generalization has been also achieved through the automatic discovery protocol of [**2**], including the (perhaps new) case describing the simultaneous equality of three (either internal or external) bisectors, placed on each one of the vertices. We refer to [**3**] (in Spanish) and to [**4**] for further details.

On the other hand, a different and more complete protocol for automatic discovery of theorems has been presented in [**6**], particularly well suited for those contexts involving the analysis of indistinguishable objects from a complex-geometry point of view, such as the internal/external bisectors at a vertex. The algebraic engine for the protocol was founded on the idea of Minimal Canonical Comprehensive Gröbner Systems (MCCGS) of [**5**]. Now, since the idea of Gröbner cover, as described in [**7**], represents a radical improvement of the MCCGS concept and algorithm, it deserved being also tested in a challenging automatic theorem proving situation. This is the precise goal of this extended abstract.

[1]Partially supported by the Spanish MCT project MTM2009-07242 and by the Gen. Cat. project 2009SGR1040.

[2]Partially supported by grant MTM2008-04699-C03-03 from the Spanish MICINN.

1 On Gröbner covers

There exist different methods to discuss parametric polynomial system of equations, that can be used to find new geometric theorems. What follows is a concise description of the Gröbner cover procedure **[7]**, that gives precise and compact information about parametric polynomial systems of equations.

Let $\overline{a} = a_1, \ldots, a_m$ be a set of parameters, $\overline{x} = x_1, \ldots, x_n$ a set of variables, and $I \subset K[\overline{a}][\overline{x}]$ an ideal (for example generated by the set of parametric equations of a geometric problem), where K is a computable field (usually $\mathbb{Q}$). Denote by $\overline{K}$ an algebraically closed extension of K (usually $\mathbb{C}$). Then $\overline{K}^m$ is the parameter space.

Selecting a monomial order $\succ$ for the variables, the Gröbner cover of $\overline{K}^m$ with respect to I is a set of pairs $GC = \{(S_i, B_i) : 1 \leq i \leq s\}$, where the S_i, called segments, are locally closed subsets of the parameter space $\overline{K}^m$, and the B_i are sets of I-regular functions $g_{ij} : S_i \to \mathcal{O}(S_i)[\overline{x}]$, that for every point $a \in S_i$ specialize to the reduced Gröbner basis of the specialized ideal I_a.

Moreover, the segments are disjoint and cover the whole parameter space, the set of leading power products on each segment are constant (and characteristic of the segment if the ideal is homogeneous) and the whole description is canonical (independent of the algorithm). It also gives a very compact discussion of the cases.

The GC-segments are given in canonical form (P-representation) providing the irreducible components of S_i and the irreducible components of the non-included points in S_i (holes). The I-regular functions in the basis B_i,

$$g_{ij} : S_i \longrightarrow \mathcal{O}(S_i)[\overline{x}],$$

are described in terms of one or more polynomials in $\mathbb{Q}[\overline{a}][\overline{x}]$ such that, for every point $(a_1, \ldots, a_m) \in S_i$, if one of them does not specialize to 0, then it specializes (after normalizing) to the corresponding polynomial of the reduced Gröbner basis, and at least one of these polynomials specializes to non-zero.

2 Automatic discovery of geometric theorems

Very roughly speaking (see **[6]** for examples and details), assume we are given a geometric construction depending on a set of points $A_1, \ldots, A_s$, whose coordinates are taken as parameters $\overline{a}$. For instance, we are given a triangle and the A-points are its vertices. Moreover, it might happen that the construction includes some new points $P_1, \ldots, P_r$, whose coordinates are taken as (dependent) variables $\overline{x}$, such as the end points of the bisector segments for each vertex.

The main problem related to theorem discovery is determining the configuration of the points A, the parameters $\overline{a}$ varying in the parameter space $\mathbb{C}^m$, in order that the points P verify some property (for example, a thesis concerning the equality of lengths for some of the segments they determine). For this purpose, we write the equations reflecting the given geometric construction and thesis, and consider the corresponding parametric ideal $I \subset \mathbb{Q}[\overline{a}][\overline{x}]$.

Let $\{(S_i, B_i) : 1 \leq i \leq s\}$ be the Gröbner cover of the parameter space with respect to I. Assuming the given thesis does not generally hold, since the locus of points A where it is satisfied will have dimension less than that of the whole parameter space. Then, the generic segment must correspond to $\mathrm{lpp} = \{1\}$. The generic segment will be of the form $S_1 = \overline{K}^m \setminus \bigcup_i V(\mathfrak{p}_i)$. The remaining segments will be all in $\bigcup_i V(\mathfrak{p}_i)$.

In most cases, we expect the locus to be associated to segments S_2 corresponding to a solution in $\overline{x}$ whose reduced Gröbner basis has the set of variables as lpp. But there can exist segments with more than one solution that we need to analyze. Moreover, there can also exist segments corresponding to degenerate constructions in which we are in general not interested. The important fact about Gröbner cover is that it provides —in a compact and concise way— all the essential pieces (a finite number of them) on the parameter space, allowing to determine those that correspond to the validity of the given statement.

3 Generalizing the Steiner-Lehmus theorem

Let ABC be a triangle. To construct the bisectors, say, at A, we consider the circle with center A and radius $\overline{AC}$. There are two intersection points P and P' of the circle with line AB, and thus two middle points Q and Q' of $\overline{CP}$ and $\overline{CP'}$ determining the bisectors $\overline{AM}$ and $\overline{AM'}$ whose length we are interested in. Setting coordinates $A(0,0)$, $B(1,0)$, $C(x,y)$ and $(p,0)$ for the intersection of the circle centered at A passing through C (i.e., points P or P'), and (a,b) for the feet of the bisectors (i.e., points M or M') we obtain the equations determining (a,b) in terms of (x,y). Notice that the sign of p determines which bisector (internal or external) of A is actually described by these equations. The length l_A of the corresponding bisector satisfies $l_A^2 = a^2 + b^2$.

Similarly we describe the bisectors of B and, then, in order to determine the conditions for which a bisector of A is equal to a bisector of B, we consider a set of equations giving the bisectors of A, the corresponding ones for B, plus the condition that one bisector (of whatever type) in A and one in B have equal length, i.e., $a^2 + b^2 = (m-1)^2 + n^2$. Then we build the Gröbner cover for the ideal given by these equations. Take the point $C(x,y)$ as a parametric point, for which we want to obtain the conditions for the system with variables a, b, m, n, p, r to have solutions. These solutions will correspond to one bisector of A being equal to one bisector of B, but the conditions over x, y will not distinguish between internal and external bisectors. Only looking for extra information on the solutions (signs of p and $1-r$) will give information about which bisectors are coincident: when p is positive, the bisector of A will be internal and it will be external if p is negative. The same happens for $1-r$ for the bisector of B. See the colors in the figure below.

The Gröbner cover algorithm is used taking the grevlex(a,b,m,n,p,r) order for the variables. Then it automatically yields a collection of segments (which is too large to be described in detail here), involving the following curves and points; see Figure 1: $\mathcal{C}_1 = \mathbb{V}(8x^{10} + 41x^8y^2 + 84x^6y^4 + 86x^4y^6 + 44x^2y^8 + 9y^{10} - 40x^9 - 164x^7y^2 - 252x^5y^4 - 172x^3y^6 - 44xy^8 + 76x^8 + 246x^6y^2 + 278x^4y^4 + 122x^2y^6 + 14y^8 - 64x^7 - 164x^5y^2 - 136x^3y^4 - 36xy^6 + 16x^6 + 31x^4y^2 + 14x^2y^4 - y^6 + 8x^5 + 20x^3y^2 + 12xy^4 - 4x^4 - 10x^2y^2 - 6y^4 + y^2)$, $\mathcal{C}_2 = \mathbb{V}(2x-1)$, $\mathcal{C}_3 = \mathbb{V}(y)$. In fact, placing vertex C at one of these curves yields the equality of lengths for bisectors at A, B. Of course, the curve $\mathcal{C}_1$ is the one providing some new insight into this theorem and leading to its generalization, too involved to be stated here in detail, cf. [**4**].

Acknowledgements

Both authors would like to dedicate this work to Prof. Ferran Hurtado, with occasion of his 60th birthday.

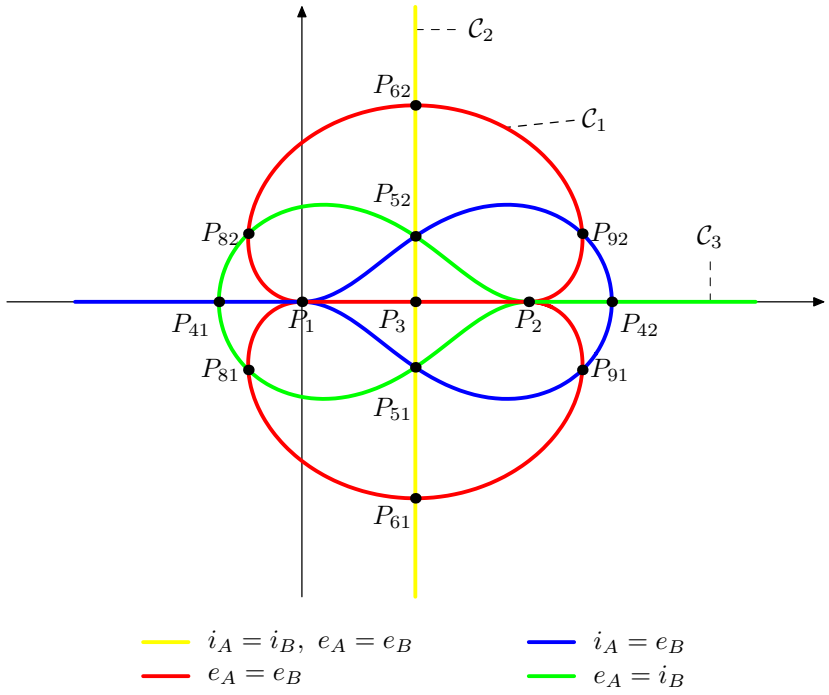


FIGURE 1. Curves $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$ and special points.

References

[1] F. Botana, Bringing more intelligence to dynamic geometry by using symbolic computation, in *Symbolic Computation and Education*, Edited by Shangzhi Li, Dongming Wang and Jing-Zhong Zhang. World Scientific, 2007, pp. 136–150.

[2] G. Dalzotto, T. Recio, On protocols for the automated discovery of theorems in elementary geometry, *Journal of Automated Reasoning*, **43**, 2009, pp. 203–236.

[3] R. Losada, T. Recio, J. L. Valcarce, Sobre el descubrimiento automático de diversas generalizaciones del Teorema de Steiner–Lehmus, *Boletín de la Sociedad Puig Adam*, **82**, 2009, pp. 53–76.

[4] R. Losada, T. Recio, J. L. Valcarce, On the automatic discovery of Steiner–Lehmus generalizations, in *Proceedings ADG 2010*, Edited by J. Richter-Gebert and P. Schreck, Munich, 2010, pp. 171–174.

[5] M. Manubens, A. Montes, Minimal canonical comprehensive Gröbner systems, *Journal of Symbolic Computation*, **44**, 5, 2009, pp. 463–478.

[6] A. Montes, T. Recio, Automatic discovery of geometry theorems using minimal canonical comprehensive Gröbner systems, in *Automated Deduction in Geometry*, Edited by F. Botana and T. Recio, LNAI (Lect. Notes Artificial Intelligence) **4869**, Springer, 2007, pp. 113–139.

[7] A. Montes, M. Wibmer, Gröbner bases for polynomial systems with parameters, *Journal of Symbolic Computation*, 2010, doi:10.1016/j.jsc.2010.06.017.

[8] D. Wang, Elimination practice: software tools and applications, *Imperial College Press*, London, 2004.

[9] W.-t. Wu, X.-l. Lü, Triangles with equal bisectors, *People's Education Press*, Beijing, 1985 (in Chinese).

[10] http://www.mathematik.uni-bielefeld.de/∼sillke/PUZZLES/steiner-lehmus.

Red-blue minimum separating circle with a moving blue point

Yam-Ki Cheung[1], **Ovidiu Dasecu**[1], **Marko Zivanic**[1]

[1] Department of Computer Science, The University of Texas at Dallas, Richardson, TX, USA
{ykcheung,daescu,mxz052000}@utdallas.edu

1 Introduction

Let $\mathcal{R}$ and $\mathcal{B}$ be two finite sets of points in $\mathbb{R}^2$, of size $|\mathcal{R}| = n$ and $|\mathcal{B}| = m$, respectively. We refer to $\mathcal{R}$ as the set of red points and to $\mathcal{B}$ as the set of blue points. In [**2**] the authors define the static version of the circular separability problem, called the *minimum separating circle problem*, as follows. Let $\mathcal{S}$ denote the set of circles such that each circle in $\mathcal{S}$ encloses all points in $\mathcal{R}$ while having the smallest number of points of $\mathcal{B}$ in its interior. The problem asks to find the smallest circle in $\mathcal{S}$, called the *minimum separating circle*. Let $C_{\mathcal{B}}(\mathcal{R})$ denote that circle. See Figure 1 for an illustration.

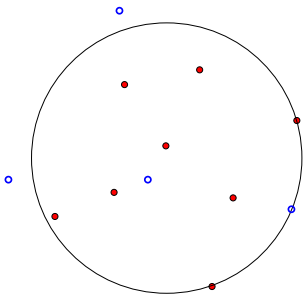


FIGURE 1. The red-blue minimum separating circle.

In this paper, we study a kinetic version of the red-blue minimum separating circle problem, in which one blue point moves with constant speed along a straight line trajectory. We want to find the locus of the minimum separating circle over a period of time.

For the case when two (static) point sets can be separated by a circle, Fisk [**3**] gave a quadratic time and space algorithm to compute the minimum separating circle, which was later improved to optimal linear time and space by O'Rourke et al. [**5**]. The linear separability problem, in which the separator is a hyperplane, reduces to linear programming, which in turn can be solved in linear time for any fixed dimension using Megiddo's algorithm [**4**]. Aronov et al. [**1**] considered the linear separability problem for point sets that may be inseparable, that is, when the points are linearly partitioned into two parts, each part may contain some misclassified points.

Our result. We show that the locus of the center of $C_{\mathcal{B}}(\mathcal{R})$ with one moving blue point has a complexity of $O(mn)$ and can be found in $O(mn \log(mn))$ time.

2 Preliminaries

We first discuss two algorithms proposed in [**2**] to solve the static version of the minimum separating circle problem that are useful for our problem. The first algorithm is based on a sweep procedure on edges of $FVD(\mathcal{R})$, while the second algorithm is based on circular range counting queries.

Given a set of static red points $\mathcal{R}$ and a set of static blue points $\mathcal{B}$, in [**2**] they showed that the minimum separating circle is either the minimum enclosing circle $MEC(\mathcal{R})$ of $\mathcal{R}$ or the circumcircle of two red points and one blue point. It follows that the center of the minimum separating circle is either a vertex of $FVD(\mathcal{R})$ or lies on an edge of $FVD(\mathcal{R})$.

Consider a Voronoi edge e_{ij}, defined by two red points r_i and r_j. The first algorithm in [**2**] starts on e_{ij} by constructing an enclosing circle C of $\mathcal{R}$ which passes through r_i and r_j and has the smallest possible radius. The center c of C is one endpoint of e_{ij}. Then, C is grown by sweeping c along e_{ij} and keeping r_i and r_j on the boundary of C. A point $E \in e_{ij}$ is an event point if, when c sweeps through E, the circle C sweeps through a blue point (see Figure 2 for an illustration).

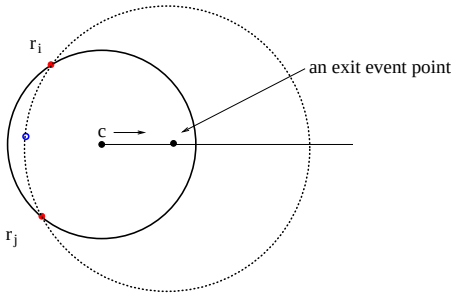


FIGURE 2. An event point on edge e_{ij}.

An event point is an *exit* event point if, at which, a blue point leaves C. It is shown in [**2**] that a blue point in $\mathcal{B}$ defines at most one exit event point on $FVD(\mathcal{R})$. The second algorithm in [**2**] finds the minimum separating circle by examining *exit* event points only.

3 The minimum separating circle with one mobile blue point

We formally define the problem as follows: Let $\mathcal{R}$ be a set of n fixed red points, let S be a set of $m-1$ blue points, and let p be a mobile blue point moving with constant speed along a straight line. Let $\mathcal{B} = S \cup p$. We want to find the locus of the center of the minimum separating circle $C_{\mathcal{R}}(\mathcal{B})$ over a period of time.

If not mentioned otherwise, we assume every point is either stationary or moving along a linear trajectory with constant speed and all stationary points are in general position. We first study a structure called *exit region*, which plays a key role in solving the kinetic version of the minimum separating circle problem. In the sweep algorithm introduced in [**2**], the center c of a separating circle C is swept along a Voronoi edge, while keeping the red points which define the Voronoi edge on its boundary. Certain region of the circle C at its initial state is excluded during the sweep. We define the exit region associated with a Voronoi edge of $FVD(R)$ as the union of points excluded from C during the execution of the sweep algorithm on this edge. Only points within $MEC(\mathcal{R}) \setminus CH(\mathcal{R})$ can be excluded by a sweep, where $CH(\mathcal{R})$ is the convex hull of $\mathcal{R}$.

All exit regions must lie within $MEC(R) \setminus CH(R)$. It is known that all exit regions are piecewise disjoint **[2]**.

Lemma 3.1. *Exit regions give a partition of $MEC(\mathcal{R}) \setminus CH(\mathcal{R})$, which is a dual of $FVD(\mathcal{R})$.*

Proof. Omitted. □

Given a blue point $b \in \mathcal{B}$, if b is enclosed by an exit region associated with a Voronoi edge e_{ij}, which is a part of the perpendicular bisector between two red points r_i and r_j, b generates an exit event point on e_{ij} and the exit event point is the center of the circumcircle $C(b, r_i, r_j)$ of b, r_i, and r_j. If b moves with constant speed along a straight line, the exit event point moves along e_{ij}.

Lemma 3.2. *If the trajectory of the mobile blue point is a straight line, the trajectory of the corresponding mobile exit event point is a cyclic path on $FVD(\mathcal{R})$.*

Proof. Obviously, the trajectory of an exit event point is a continuous curve. We prove this lemma by considering two cases. *Case one:* the trajectory of the mobile blue point p is a line intersecting both $CH(\mathcal{R})$ and $MEC(\mathcal{R})$. When p enters $MEC(\mathcal{R})$, it creates an exit event point at the root of $FVD(R)$, which moves along the edge defining the first exit region it visits. When p enters a new exit region by crossing a circumcircle of three red points, the exit event point crosses a vertex of $FVD(R)$, which is the center of the circumcircle that separates these two exit regions, and moves to a new Voronoi edge. Eventually, the exit event point visits an unbounded Voronoi edge, when it enters an exit region bounded by an edge of $CH(\mathcal{R})$ and one circumcircle, and moves to ∞ as it approaches the boundary of $CH(\mathcal{R})$. Note that p does not have an exit event point when $p \in CH(\mathcal{R})$. Similarly, when p exits from $CH(\mathcal{R})$, its exit event point re-appears at ∞, and travels along a path on $FVD(\mathcal{R})$, and eventually reaches the root of $FVD(R)$ when p crosses $MEC(\mathcal{R})$ the second time to form a closed path.

Case two: the trajectory of p is a line intersecting $MEC(\mathcal{R})$ but not $CH(\mathcal{R})$. Let r_i and r_j be two red points which define a Voronoi edge e_{ij} in $FVD(\mathcal{R})$. The exit region associated with e_{ij} is bounded by two circular arcs, which are both incident to r_i and r_j. It follows that if the trajectory of p intersects one arc of the exit region, it must intersect the arc exactly twice. Otherwise the trajectory intersects the line segment connecting r_i and r_j, which is enclosed by $CH(\mathcal{R})$. Note that r_i and r_j must be vertices of $CH(\mathcal{R})$. This implies that the trajectory of the corresponding exit event point starts at the root of $FVD(\mathcal{R})$ and traverses a path along edges of $FVD(\mathcal{R})$ and returns back to the root following the same path. □

Next, we give details of the solution for this problem. Observe that each fixed blue point defines at most one static exit event point, which in turn defines a fixed candidate separating circle. The number of blue points enclosed by such candidate separating circle changes only when the mobile blue point p enters or leaves the circle. The mobile blue point p defines a mobile exit event point. Not only the center and radius of the corresponding candidate circle changes continuously, but the number of blue points enclosed by the corresponding candidate separating circle changes over time, as well. We need to dynamically maintain two classes of structures: (1) the trajectory of the moving exit event point, and (2) the number of blue points enclosed by each candidate separating circle over time. We need to analyze the following events and update the corresponding structures accordingly.

Case 1) The mobile blue point enters or leave a static candidate circle.
Case 2) The exit event point associated with the mobile blue point moves to a new edge of the farthest neighbor Voronoi diagram $FVD(R)$ of $\mathcal{R}$.
Case 3) The candidate circle associated with the mobile blue point encloses or excludes a new blue point.

To avoid ambiguity, we refer to these events as instant events, distinguishing from the event points introduced in Section 2.

Case 1 *instant events.* These are the instant events when the mobile blue point enters or leave a fixed candidate circle.

Lemma 3.3. *There are $O(m)$ case* 1 *instant events, and can be found in constant time each, given that all fixed exit event points are known.*

Proof. Trivial. □

Case 2 *instant events.* These are the instant events when the exit event point associated to the mobile blue point moves to a new edge of the farthest neighbor Voronoi diagram $FVD(\mathcal{R})$ of $\mathcal{R}$.

Lemma 3.4. *We have $O(n)$ case* 2 *instant events, which can be computed in $O(n)$ time.*

Proof. Omitted. □

Case 3 *instant events.* These are the instant events when the candidate circle associated to the mobile blue point encloses or excludes a blue point.

Lemma 3.5. *There are $O(mn)$ case* 3 *instant events.*

Proof. Omitted. □

Thus the trajectory of each exit event point and the count of blue points enclosed by each candidate circle are maintained over time.

Theorem 3.6. *The locus of the minimum separating circle of $\mathcal{R}$ and $\mathcal{B}$ has a complexity of $O(mn)$ can be computed in $O(mn\log(mn))$ time by a sweep algorithm.*

References

[1] B. Aronov, D. Garijo, Y. Núñez, D. Rappaport, C. Seara and J. Urrutia, Measuring the error of linear separators on linearly inseparable data, in *XIII Encuentros de Geometría Computacional*, Zaragoza, España, 2009.
[2] S. Bitner, Y. K. Cheung and O. Daescu, *Minimum separating circle for bichromatic points in the plane*, in *ISVD*, 2011, 50–55.
[3] S. Fisk, Separating point sets by circles, and the recognition of digital disks, in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1986, 554–556.
[4] N. Megiddo, Linear-time algorithms for linear programming in $\mathbb{R}^3$ and related problems, *SIAM Journal on Computing* **12** (1983), 759–776.
[5] J. O'Rourke, S. Kosaraju and N. Megiddo, Computing circular separability, *Discrete Computational Geometry* **1** (1986), 105–113.

Weak separators, vector dominance, and the dual space

Canek Peláez[1], **Adriana Ramírez-Vigueras**[1], **Carlos Seara**[2], **Jorge Urrutia**[3]

[1] Posgrado en Ciencia e Ingeniería de la Computación, Universidad Nacional Autónoma de México
{canek,adriana.rv}@ciencias.unam.mx

[2] Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya
carlos.seara@upc.es

[3] Instituto de Matemáticas, Universidad Nacional Autónoma de México
urrutia@matem.unam.mx

Abstract. In this paper we study two problems related to vector dominance and rectilinear separators of point sets on the plane. We show that the best *weak separator* of a set of bicolored points on the plane can be obtained in $O(n^2)$ time. We also study some problems arising from the rectilinear convex hull of point sets, but in the dual space. This produces some attractive geometric visualizations of staircases and rectilinear separators in that space.

Introduction

In this paper we study two problems arising from the study of the rectilinear convex hull of point sets on the plane. Without loss of generality, and to make our presentation easier, we will assume that all the points in our point sets have positive coordinates.

A *quadrant* of the plane is the intersection of two half-planes whose supporting lines are parallel to the x- and y-axes. Let S be a set of points in the plane in general position. We say that a quadrant is *S-free* if its interior contains no point in S.

The *rectilinear convex hull* of a point set S is defined as

$$\mathcal{RH}(S) = \mathbb{R}^2 - \bigcup_{Q \text{ is an } S\text{-free quadrant}} Q.$$

Observe that, if we rotate the plane around the origin, the rectilinear convex hull of a point set changes. The problem of finding a rotation of the plane that produces a rectilinear convex hull with minimum area was studied in [**1**], where an $O(n^2)$ time algorithm to solve this problem was presented. Their algorithm was improved to $\Theta(n \log n)$ in [**2**].

The rectilinear convex hull of a point set was first studied in [**3**]. A point $p = (a, b) \in S$ is dominated by $q = (c, d) \in S$, $p \neq q$, if $a \leq c$ and $b \leq d$. A polygonal curve C is called *rectilinear* if it consists of a sequence of line segments each of which is horizontal or vertical, and C is called a *staircase* if it is monotone with respect to the x- and y-axes. In the rest of this paper, we will further assume that a staircase is monotonically decreasing with respect to the x-axis.

[1]Partially supported by project SEP-CONACYT of México, Proyecto 80268. The authors would like to thank CONACYT for the support.

[2]Partially supported by projects MTM2009-07242 and Gen. Cat. DGR 2009GR1040.

[3]Partially supported by projects MTM2006-03909 (Spain) and SEP-CONACYT 80268 (México).

Let $S = R \cup B$ be a set of n points on the plane in general position such that the elements of R and B are colored red and blue respectively. In this paper, we are interested in the following problem: Find a staircase that best *classifies* R and B; that is, find a staircase C such that the number of red points below it plus the number of blue points above it is maximized. Such a staircase we called the best *weak staircase separator*. We obtain an $O(n^2)$-time algorithm to solve this problem.

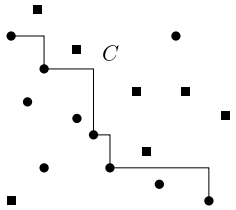


FIGURE 1. A staircase polygonal weak separator C separating red points (●) and blue points (■).

While solving the above problem, we stumbled on the following problem: Can we give an interpretation of the rectilinear convex hull of a set of points in the dual space? What about the concept of rectilinear separability?

Recall that the dual of a point $p = (a, b)$ of the plane, denoted by ℓ_p, is the non-vertical line with equation $y = ax - b$. The dual of ℓ_p is p. It is well known that, under duality, collinear points are mapped to sets of concurrent lines, and concurrent lines are mapped to collinear points **[4, 5]**.

In Section 2, we study this problem, and show an attractive interpretation of the rectilinear convex hull of a point set. We will assume that our point sets are contained in the positive quadrant of the plane, and show that the rectilinear convex hull of a point set looks like a set of rays emanating from a *sun*.

1 Computing the best staircase weak separator

In this section, we outline our algorithm to obtain a best staircase weak separator. Our algorithm is based on dynamic programming. We perform first an $O(n \log n)$ preprocessing on $R \cup B$, and then perform a line sweep from left to right, stopping at every point of R. It is easy to see that the best staircase weak separator can be chosen in such a way that it is determined by a set of points in R, which are the right endpoints of the horizontal lines of the staircase. For every point r_i in R, we maintain the optimal weak separator whose rightmost vertex is precisely r_i.

Let us assume that the elements of S are sorted from left to right according to their x-coordinate, and that the elements of R are labelled $\{r_1, \ldots, r_m\}$ in such a way that, if $i < j$, the point r_i is to the left of r_j. This labeling can be achieved in $O(n \log n)$ time. Recall that using quadratic preprocessing on S **[6]**, we can find in constant time the number of red and blue points of S within any isothethic rectangle. We now sweep a vertical line from left to right stopping at all the points in R.

For each point r_j in R, we find in $O(n)$ time the point r_i such that the optimal weak separator whose last two vertices are r_i and r_j. We can do this in $O(n)$ time since, for each r_k, $k < j$, we can calculate in constant time the number of red and blue points dominated by r_j that are not dominated by r_k. Due to lack of space, the proof of the

correctness of our algorithm is omitted. Our algorithm works in $O(n^2)$ time. Thus we have the following result:

Theorem 1.1. *An optimal staircase weak separator of S can be calculated in $O(n^2)$ time.*

Suppose now that we can rotate the plane. We would like to find an angle θ such that, when we rotate the plane θ degrees around the origin, we obtain the best weak separator over all $\theta \in [0, 2\pi)$. An immediate corollary of Theorem 1.1 is that we can find the best unoriented weak separator in $O(n^4)$ time by getting the best oriented weak separators in each of the $\binom{n}{2}$ combinatorially distinct directions of S, and choosing the best one of all. We believe that finding the angle θ that produces the best unoriented weak separator can be done in $O(n^3 \log n)$ time.

2 Vector dominance and staircases in the dual space

Consider the elements of S under the partial order defined by $(a, b) \succ (c, d)$ if and only if $a \geq c$ and $b \geq d$, $(a, b) \neq (c, d)$. Since all the lines ℓ_p in the dual space are non-vertical, the y-axis splits them into two rays. The ray to the right of y-axis will be denoted by ℓ_p^+, the one to its left ℓ_p^-, and they will be called, respectively, the *positive* and the *negative* semi-lines of ℓ_p.

Observe that a point p dominates another point s in the partial order $\succ$ if and only if the slope of ℓ_p is greater than the slope of ℓ_s, and ℓ_p intersects the y-axis below the point where ℓ_s intersects it. This implies that ℓ_p and ℓ_s intersect each other to the right of y-axis, or simply that ℓ_p^+ intersects ℓ_s^+ (Figure 2).

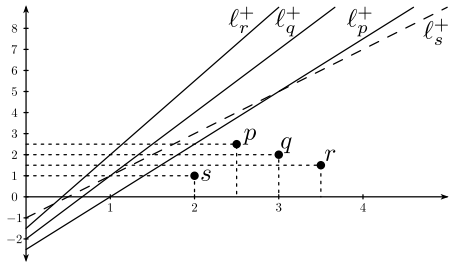
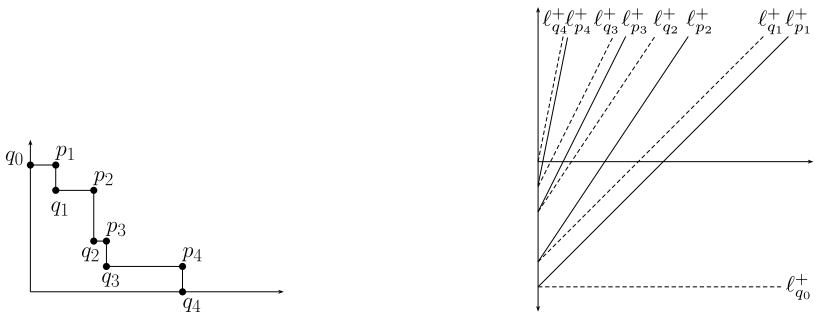


FIGURE 2. The anti-chain formed by p, q, and r, the dominated point s, and their transformations in the dual space.

On the other hand, if two points p and q of S are not comparable in $\succ$, then ℓ_p^+ and ℓ_q^+ do not intersect. If in the dual we consider only the positive semi-lines of the dual lines of the elements of S, then we can see that an anti-chain of points in the partial order generates a set of non-intersecting rays with increasing slopes (Figure 2).

Every anti-chain $p_1, p_2, \ldots, p_k$ of S with respect to $\succ$ determines a staircase polygonal chain $\mathcal{S}$ as shown in Figure 3(a). Define points $q_0, \ldots, q_k$ on the staircase defined by $p_1, p_2, \ldots, p_k$ as in Figure 3(a). Since $p_1, \ldots, p_k$ are pairwise non comparable, $\ell_{p_1}^+, \ldots, \ell_{p_k}^+$ do not intersect each other.

If we traverse $\mathcal{S}$ from q_0 to q_k, we can see that, when we traverse the horizontal segment defined by the points q_i and p_{i+1}, in the dual space we rotate the ray $\ell_{q_i}^+$ until its slope is the same as that of $\ell_{p_{i+1}}^+$. When we traverse the vertical segment defined by



(a) An anti-chain with extra points to form a staircase.

(b) The anti-chain and extra points in the dual space.

FIGURE 3. An anti-chain $p_1, \ldots, p_4$ with extra points $q_0, \ldots, q_4$ and how it looks in the dual space.

the points p_i and q_i in the dual space we translate the ray $\ell^+_{p_i}$ upwards until it reaches $\ell^+_{q_i}$; see Figure 3(b).

3 Conclusions

We point out that our study of vector dominance in the dual space has allowed us to give attractive geometric interpretations of objects such as empty isothetic rectangles with opposite vertices in a fixed point set. In addition, it has enabled us to develop algorithms such as finding the rectilinear convex hull of a point set, or efficiently calculating the set of points below a staircase. These algorithms work directly in the dual space, and usually have the same complexity as those in the *primal space*. In the full version of this paper, we will explore these results in more detail.

References

[1] 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)** (2009), 903–912.

[2] C. Alegría-Galicia, T. Garduño, A. Rosas-Navarrete, C. Seara, J. Urrutia, Rectilinear convex hull with minimum area, in: *XIV Spanish Meeting on Computational Geometry*, CRM Documents, vol. 8, Centre de Recerca Matemàtica, Bellaterra (Barcelona), 2011, 185–188.

[3] T. Ottmann, E. Soisalon-Soininen, D. Wood, On the definition and computation of rectilinear convex hulls, *Information Sciences* **33(3)** (1984), 157–171.

[4] H. Edelsbrunner, J. O'Rourke, R. Seidel, Constructing arrangements of lines and hyperplanes with applications, in: *SFCS'83, Proceedings of the 24th Annual Symposium on Foundations of Computer Science* (1983), 83–91.

[5] H. Edelsbrunner, L. Guibas, Topologically sweeping an arrangement, in: *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing* (1986), 389–403.

[6] F. P. Preparata, M. I. Shamos, *Computational Geometry: An Introduction*. Springer-Verlag, 1985.

Explicit array-based compact data structures for triangle meshes

Luca Castelli Aleardi[1], **Olivier Devillers**[2]

[1] LIX, Ecole Polytechnique, France
amturing@lix.polytechnique.fr

[2] INRIA Sophia-Antipolis, France
olivier.devillers@inria.fr

Abstract. We consider the problem of designing space-efficient explicit data structures for planar and surface meshes. Our main result is a new explicit data structure for compactly representing planar triangulations (corresponding to genus 0 triangle meshes): if one is allowed to permute input vertices, then a triangulation with n vertices requires at most $4n$ references ($5n$ references if vertex permutations are not allowed). Our solution combines existing techniques from mesh encoding with a novel use of Schnyder woods. As far as we know, our solution provides the most parsimonious data structures for triangle meshes, allowing constant time navigation in the worst case.

1 Introduction

The large diffusion of geometric meshes and especially their increasing combinatorial complexity has motivated a huge number of recent works in the domain of graph encoding. Many works addressed the problem from the *compression* **[11, 12]** point of view, where the goal is to reduce the number of bits as much as possible. For applications requiring the manipulation of input data, a number of explicit data structures **[2, 4, 7]** have been developed for most common classes of meshes. Some solutions **[1, 6, 8, 9, 10, 14]** aimed to reduce the redundancy of common explicit representations: the goal is to obtain more *compact explicit* data structures, whose performances (in terms of running time and space efficiency) are of practical interest. Table 1 provides a comparison of most existing mesh data structures.

1.1 Preliminaries

A key ingredient of our work is a fine characterization of planar triangulations given in terms of edge orientations. As pointed out by Schnyder **[13]**, the inner edges of a planar triangulation with root face (v_0, v_1, v_2) can be partitioned into three sets $\mathcal{T}_0$, $\mathcal{T}_1$, $\mathcal{T}_2$, which are plane trees spanning all inner nodes, and rooted at v_0, v_1 and v_2 respectively.

Definition 1.1 ([13]). Let $\mathcal{G}$ be a planar triangulation with root face (v_0, v_1, v_2). A *Schnyder wood* of $\mathcal{G}$ is an orientation and labeling, with labels in $\{0, 1, 2\}$ of the inner edges such that the edges incident to the vertices v_0, v_1, v_2 are all ingoing (and respectively of color 0, 1, and 2). Moreover, each inner vertex v has exactly three outgoing incident edges, one for each color, and the edges incident to v in *counter clockwise* (ccw) order are: one outgoing edge colored 0, zero or more incoming edges colored 2, one outgoing edge colored 1, zero or more incoming edges colored 0, one outgoing edge colored 2, and zero or more incoming edges colored 1.

Data structure	size	navigation	vertex access	dynamic
2D catalogs [**6**]	$7.67n$	$O(1)$	$O(1)$	yes
Compact array-based half-edge [**1**]	$7n$	$O(1)$	$O(1)$	yes
Star vertices [**10**]	$7n$	$O(d)$	$O(1)$	no
TRIPOD [**14**] + reordering	$6n$	$O(1)$	$O(d)$	no
SOT data structure [**9**]	$6n$	$O(1)$	$O(d)$	no
SQUAD data structure [**8**]	$(4+c)n$	$O(1)$	$O(d)$	no
Our DS, Theorem 2.2	$5n$	$O(1)$	$O(d)$	no
Our DS, Theorem 2.3	$4n$	$O(1)$	$O(d)$	no

TABLE 1. Comparison between compact data structures for triangle meshes.

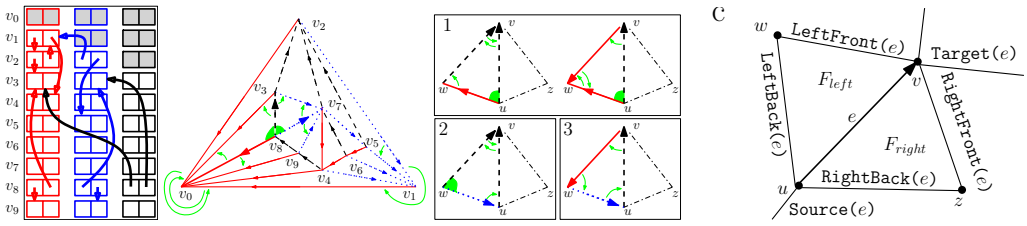


FIGURE 1. Array-based representation using $6n$ references.

2 Compactly representing triangulations

The data structures presented here are edge-based and support efficient navigation operations as in most common mesh representations; operators are illustrated in Fig. 1 (right). One main step, as suggested in [**9**], is to reorder the cells in the mesh (the half-edges in our case), to implicitly represent the map from vertices to edges, and the map from edges to vertices: this saves one reference for each vertex and one reference for each edge). Then we exploit the existence of 3-orientations (edge orientations where every inner vertex has outgoing degree 3) for planar triangulations [**13**]. These two first ideas lead to a compact representation requiring 6 references per vertex, with 2 references for each outgoing edge (a similar approach has been proposed by Snoeyink and Speckmann [**14**]).

Vertices will be identified to integers $0 \leq i < n$ and edges to integers $3 \leq j < 3n$. Our data structure consists of an array T of size $6n$, two arrays of bits S_a, S_b of size $3n$, and an array P of size n storing the geometric coordinates of the points. The entries of T and P are sorted according to the order of input points. By convention, the three edges having vertex i as source are indexed $3i$, $3i+1$ and $3i+2$, where the edge having index $3i+c$ has color c. For each oriented edge we store two references to 2 neighboring edges. References are arranged in table T, in such a way that for each inner node u of $\mathcal{G}$, the outgoing edges associated with u are stored consecutively in T. Then the adjacency relations of edge j are stored in entries $2j$ and $2j+1$ of table T, as follows:

- $T[2j]$ contains the index of `LeftFront(`j`)`, and $T[2j+1]$ = `RightFront(`j`)`.

Arrays S_a and S_b have an entry for each edge and are defined as follows:

- $S_a[j] = 1$ if edge j and `LeftBack(`j`)` have the same source, 0 otherwise;
- $S_b[j] = 1$ if edge j and `RightBack(`j`)` have the same source, 0 otherwise.

Theorem 2.1. *Let $\mathcal{G}$ be a triangulation with n vertices. The representation above requires $6n$ references, while supporting in $O(1)$ time navigation between faces, and the access to a vertex of degree d in $O(d)$ time. The construction phase requires $O(n)$ time.*

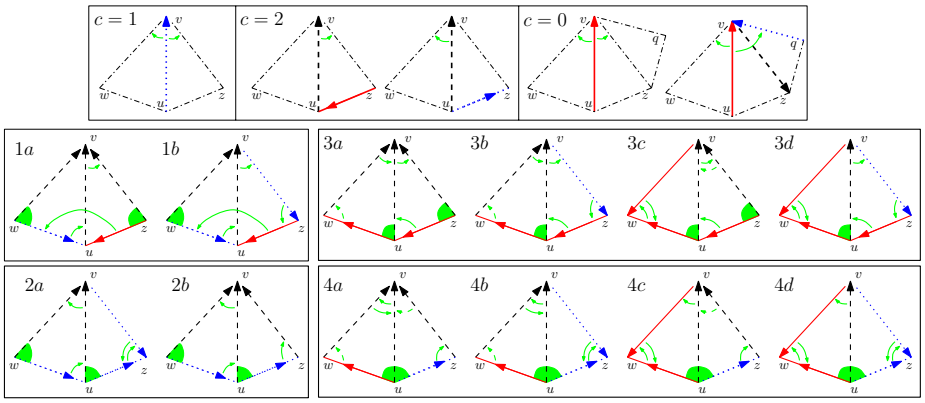


FIGURE 2. A more compact scheme. Neighboring relations between edges are represented by tiny oriented (green) arcs corresponding to stored references, and by filled (green) corners which describe adjacency relations between outgoing edges incident to a same vertex.

2.1 A more compact solution

In order to reduce the space requirements, we exploit the existence of a special Schnyder wood, called *minimal*, without ccw oriented cycles of directed edges —a minimal Schnyder wood can be computed in linear time. In particular, a minimal Schnyder wood does not contain ccw oriented triangles; this property allows to reduce by 1 the number of references per vertex. We slightly modify the representation described in the previous section. Outgoing edges of color 0 and 1 will be still represented with two references each, while we will store only one reference for each outgoing edge of color 2 (different cases are illustrated by Fig. 2, top pictures). Let us denote by $e = (u, v)$ an edge having face (u, v, w) at its left and face (u, v, z) at its right, and let q the vertex defining the triangle (v, z, q). Then, for an edge (u, v) of color c, outgoing from vertex u (for vertex u we store five entries $T[5u], \ldots, T[5u + 4]$):

- for $c = 1$ (as in Theorem 2.1), we store in $T[5u + 2]$ and $T[5u + 3]$ two references, respectively to (v, w) and (v, z) (edges cw and ccw around v);
- for $c = 2$, we store in $T[5u + 4]$ a reference to:
  - edge (v, z), if (z, u) is directed toward u (edge ccw around v);
  - edge (v, w) otherwise (edge cw around v);
- for $c = 0$, we store one reference in $T[5u]$ to (v, w) (edge cw around v) and one reference in $T[5u + 1]$ to:
  - edge (v, q) if (v, q) is of color 1 oriented toward v —and thus (v, z) must be of color 2 (second edge ccw around v);
  - edge (v, z) otherwise (edge ccw around v), as in Theorem 2.1.

Concerning service bits, we need a third array S_c of size $3n$ (which is needed to distinguish the two different cases above when $c = 0$):

- as in the previous section, the values of $S_a[e]$ and $S_b[e]$ describe the orientations of edges `LeftBack`(e) and `RightBack`(e);
- $S_c[e] = 1$ if the second edge which follows e around v in ccw direction is of color 1, and $S_c[e] = 0$ otherwise (see top-right pictures in Fig. 2).

The correctness of the next theorem is based on an involved case analysis (see Fig. 2).

Theorem 2.2. *Let $\mathcal{G}$ be a triangulation with n vertices. There exists a representation requiring $5n$ references, allowing efficient navigation, as in Theorem* 2.1.

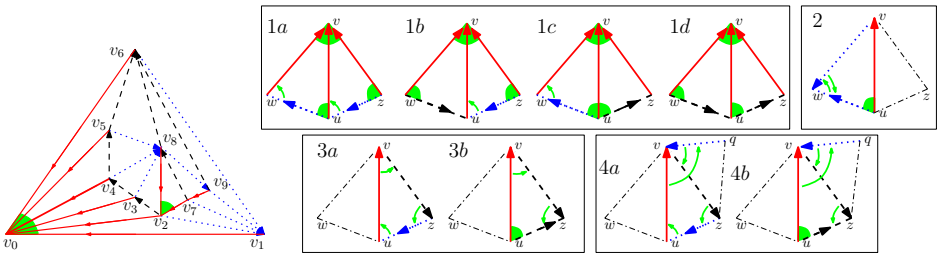


FIGURE 3. Vertices are labeled according to DFUDS order of $\overline{\mathcal{T}}_0$ (left). Most adjacency relations can be described by the DFUDS labels (right).

2.2 Further reducing the space requirement

Let us first recall a result concerning the traversal of plane trees, which has been already applied to the encoding of trees [**3**]. Let $\mathcal{T}$ be a plane tree whose nodes are labeled according to the DFUDS (Depth First Unary Degree Sequence) traversal of $\mathcal{T}$: the children of a given node $v \in \mathcal{T}$ have all consecutive labels. We now allow to exploit a permutation of the input vertices: we re-order all vertices (their associated data) according to their DFUDS label, and we store entries in table T accordingly. This allows us to save one reference per vertex: we do not need to store a reference to `LeftFront` for edges in $\overline{\mathcal{T}}_0 := \mathcal{T}_0 \cup (v_0, v_1)$, which leads to store for each vertex 4 references in table T.

Theorem 2.3. *Let $\mathcal{G}$ be a triangulation with n vertices. If one is allowed to permute the input vertices (their associated geometric data), then $\mathcal{G}$ can be represented using $4n$ references, supporting navigation as in previous representations.*

References

[1] T. J. Alumbaugh and X. Jiao. Compact array-based mesh data structures. In *Proc. of the 14th International Meshing Roundtable (IMR)*, 485–503, 2005.

[2] B. G. Baumgart. Winged-edge polyhedron representation. Technical report, Stanford, 1972.

[3] D. Benoit, E. D. Demaine, J. I. Munro, R. Raman, V. Raman, and S. S. Rao. Representing trees of higher degree. *Algorithmica*, 43(4):275–292, 2005.

[4] J.-D. Boissonnat, O. Devillers, S. Pion, M. Teillaud, and M. Yvinec. Triangulations in CGAL. *Comp. Geometry*, 22:5–19, 2002.

[5] S. Campagna, L. Kobbelt, and H. P. Seidel. Direct edges —a scalable representation for triangle meshes. *Journal of Graphics Tools*, 3(4):1–12, 1999.

[6] L. Castelli-Aleardi, O. Devillers, and A. Mebarki. 2D triangulation representation using stable catalogs. In *CCCG*, 71–74, 2006.

[7] 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.

[8] T. Gurung, D. Laney, P. Lindstrom, and J. Rossignac. *SQUAD:* Compact representation for triangle meshes. In *Proc. of Eurographics 2011*, 2011.

[9] T. Gurung and J. Rossignac. *SOT:* compact representation for tetrahedral meshes. In *Proc. of the ACM Symp. on Solid and Physical Modeling*, 79–88, 2009.

[10] M. Kallmann and D. Thalmann. Star-vertices: a compact representation for planar meshes with adjacency information. *Journal of Graphics Tools*, 6:7–18, 2002.

[11] D. Poulalhon and G. Schaeffer. Optimal coding and sampling of triangulations. *Algorithmica*, 46:505–527, 2006.

[12] J. Rossignac. Edgebreaker: Connectivity compression for triangle meshes. *Transactions on Visualization and Computer Graphics*, 5:47–61, 1999.

[13] W. Schnyder. Embedding planar graphs on the grid. In *SODA*, 138–148, 1990.

[14] J. Snoeyink and B. Speckmann. Tripod: a minimalist data structure for embedded triangulations. In *Workshop on Computational Graph Theory and Combinatorics*, 1999.

Index of Authors