Proceedings

XV Spanish Meeting on Computational Geometry

June 26-28, 2013
Sevilla, Spain

Edited by:
José Miguel Díaz-Báñez
Alberto Márquez
Delia Garijo
Jorge Urrutia

Supported by:

Vicerrectorado de Relaciones Internacionales

Departamento de Matemática Aplicada II

Preface

The XV Spanish Meeting on Computational Geometry (XV EGC,"Encuentros de Geometría Computacional") was held on June 26–28, 2013 in Seville, Spain. The EGC meetings were first held annually from 1990 to 1995, and biennually since then. The meeting combines a strong scientific tradition with a friendly, collegial atmosphere. The original aim of our meeting was to provide a forum where Spanish-speaking researchers and students could present their research activities. Since then the EGC has evolved; in 2011, the XIV EGC was an international gathering for the first time, and the abstracts contained in the 2011 proceedings were peer reviewed.

This volume contains the four-page extended abstracts of the presentations accepted and delivered at the 2013 EGC, and the four invited lectures. A selection of the papers accepted for the EGC will be published in the *International Journal of Computational Geometry and Applications* following a formal review according to the standards established by the latter journal. We would like to thank all the authors and attendees for their participation in the XV ECG. Special thanks are expressed to the members of the Scientific Committee and the referees for their careful review of the papers and their insightful comments. Lastly, we are grateful for the generous support of our sponsors: Departamento de Matemática Aplicada II, Vicerrectorado de Relaciones Institucionales, IMUS (Instituto de Matemáticas) and Departamento de Matemática Aplicada I, all of the Universidad de Sevilla.

The Editors

Committees

Scientific Committee

Oswin Aichholzer
Sergey Bereg
Prosenjit K. Bose
Sergio Cabello
José Miguel Díaz-Báñez (co-chair)
Alfredo García
Ferran Hurtado
Rolf Klein
Stefan Langerman
Mario López
Alberto Márquez (co-chair)
Belén Palop
Pedro Ramos
David Rappaport
Vera Sacristán
Gelasio Salazar
J. Antoni Sellarès
Jorge Urrutia (co-chair)
David Wood

Organizing Committee

Carmen Cortés
María José Chávez
José Miguel Díaz-Báñez (chair)
Isabel Fernández
Delia Garijo
M. Ángeles Garrido
Alberto Márquez
Pablo Pérez-Lantero
Inmaculada Ventura

Contents

Session 7–Friday 09:30–11:10

Session 8–Friday 11:40–13:00

Invited speaker 4–Friday 13:15–14:15

Art gallery problems, old and recent

Jorge Urrutia*

Instituto de Mathemáticas, Universidad Nacional Autónoma de Mexico (UNAM), Mexico City, México.

Abstract

In 1973, Victor Klee posed the following question: How many guards are necessary, and how many are sufficient to patrol the paintings and works of art in an art gallery with n walls? This wonderfully naïve question of combinatorial geometry has, since its formulation, stimulated a plethora of papers, surveys and a books. The first result in this area, due to V. Chvátal, asserts that $\lfloor \frac{n}{3} \rfloor$ guards are occasionally necessary and always sufficient to guard an art gallery represented by a simple polygon with n vertices. Since Chvátal's result, numerous variations on the art gallery problem have been studied, including mobile guards, guards with limited visibility or mobility, illumination of families of convex sets on the plane, guarding of rectilinear polygons, and others. In this talk, we will review some old results in this area of research, and review some new results on Art Galleries, including some recent results in $\mathbb{R}^3$, as well as results using rotating floodlights.

*Email: urrutia@matem.unam.mx. Partially supported by CONACYT of México Grant 178379, and MEC project MTM2009-08652.

Continuous surveillance of points by rotating floodlights

S. Bereg*[1], J. M. Díaz-Báñez†[2], M. Fort‡[3], M. A. Lopez§[4], P. Pérez-Lantero¶[5], and J. Urrutia‖[6]

[1]Department of Computer Science, University of Texas at Dallas, USA.
[2]Departamento Matemática Aplicada II, Universidad de Sevilla, Spain.
[3]Departament d'Informàtica i Matemàtica Aplicada, Universitat de Girona, Spain.
[4]Department of Computer Science, University of Denver, USA.
[5]Escuela de Ingeniería Civil en Informática, Universidad de Valparaíso, Chile.
[6]Instituto de Matemáticas, UNAM, Mexico.

Abstract

Let P and F be sets of $n \geq 2$ and $m \geq 2$ points in the plane, respectively, so that $P \cup F$ is in general position. We study the problem of finding the minimum angle $\alpha \in [2\pi/m, 2\pi]$ such that one can install at each point of F a stationary rotating floodlight with illumination angle α, initially oriented in a suitable direction, in such a way that, at all times, every target point of P is illuminated by at least one light. All floodlights rotate at unit speed and clockwise. We give an upper bound for the 1-dimensional problem and present results for some instances of the general problem. Specifically, we solve the problem for the case in which we have two floodlights and many points, and give an upper bound for the case in which there are many floodlights and only two target points.

1 Introduction

Illumination problems are well known in Discrete and Computational Geometry [4]. Let P be a set of $n \geq 2$ targets and F a set of $m \geq 2$ floodlights, both defined in the plane. We assume that $P \cup F$ is in general position and that all floodlights rotate clockwise at unit speed. We say that F *covers* P *with illumination angle* $\alpha \geq 2\pi/m$ if there is a suitable initial orientation of each light so that, at all times, each target point of P is illuminated by at least one floodlight. We consider the problem of finding the minimum angle $\alpha = \alpha(P, F) \in [2\pi/m, 2\pi]$ and initial orientation of each floodlight so that F covers P.

Our general problem can be considered a discretization of the one studied by Kranakis et al. [2]. Given the locations of m floodlights (i.e. antennae) and a region, their problem asks to schedule the lights so that the entire region is covered at all times. The scheduling of static floodlights for covering a given region was first considered by Bose et al. [1]. Research related to our problem can be found in a number of domains, including art gallery and related problems, multi-target tracking, and multi-robot surveillance tasks [3, 4]. A complete review of these fields can be found in [4].

We present results for some cases of our general problem: the elements of $P \cup F$ are located on a given line (Section 2), the two-dimensional version with two floodlights (Section 3), and the problem in the plane with two target points (Section 4).

Given points u, v in the plane, $\ell(u, v)$ denotes the line containing both u and v. We identify any floodlight by the point where it is installed. For any floodlight f, at any instance of time, the region illuminated by f is delimited by rays f^- and f^+, starting at f^- and ending at f^+ in the clockwise direction. We say that we configure f *with angle* β if the angle between f^+ and the positive x-axis is equal to β. Given that all floodlights rotate at the same speed, it suffices to consider only the interval of time $[0, 2\pi)$.

2 Points and floodlights on a line

We first consider the case in which the points of P and the floodlights of F lie on a line $\mathcal{L}$, say the x-axis. Kranakis et al. [2] considered the case in which the floodlights are located on a line. They showed that the entire line can be illuminated by m rotating floodlights using illumination angle $3\pi/m$ and this bound is tight. This can be viewed as a special case of our problem where $n \geq m + 1$ and each of the $m + 1$ segments determined by F contains at least one point of

*Email: besp@utdallas.edu. Partially supported by project MEC MTM2009-08652.

†Email: dbanez@us.es. Partially supported by project MEC MTM2009-08652 and ESF EUROCORES programme EuroGIGA-ComPoSe IP04-MICINN Project EUI-EURC-2011-4306.

‡Email: mfort@ima.udg.edu. Partially supported by the Spanish MCI grant TIN2010-20590-C02-02.

§Email: mlopez@du.edu.

¶Email: pablo.perez@uv.cl. Partially supported by project CONICYT, FONDECYT/Iniciación 11110069 (Chile), and project MEC MTM2009-08652.

‖Email: urrutia@matem.unam.mx. Partially supported by project MEC MTM2009-08652.

P. We consider other cases and show that the illumination angle is smaller than $3\pi/m$ for some of them.

Partition P into $k-1$ ($k \geq 2$) maximal intervals $s_1, s_2, \ldots, s_{k-1}$, from left to right, each of which contains elements of P but no elements of F. Let F_1 denote the elements of F to the left of s_1, F_i ($i = 2, \ldots, k-1$), the elements of F between s_{i-1} and s_i, and F_k, the elements of F to the right of s_{k-1}. Let $m_i = |F_i|$ for $i = 1, \ldots, k$. Observe that $m_1, m_k \geq 0$, $m_i \geq 1$ for $i = 2, \ldots k-1$, and $m_1+m_2+\ldots+m_k = m$.

Lemma 1 *Two floodlights f_1 and f_2 with illumination angle α, belonging to the same set among $F_1 \cup F_k, F_2, F_3, \ldots, F_{k-1}$, can be configured to cover P during 2α time in total. Furthermore, if two floodlights of F cover P with angle $\alpha < 3\pi/2$, then they must belong to a same set among $F_1 \cup F_k, F_2, F_3, \ldots, F_{k-1}$.*

Proof. Suppose that both f_1 and f_2 belong to a set F_i ($i = 1, \ldots, k$), and assume w.l.o.g. that f_1 is to the left of f_2. Configure f_1 with angle zero and f_2 with angle π (see top of Figure 1a). At time $t = \pi$ the configuration of f_1 and f_2 is as shown in the bottom of Figure 1a. Since there is no element of P in the segment connecting f_1 and f_2 then all elements of P are illuminated during intervals $[0, \alpha]$ and $[\pi, \pi + \alpha]$, 2α time in total.

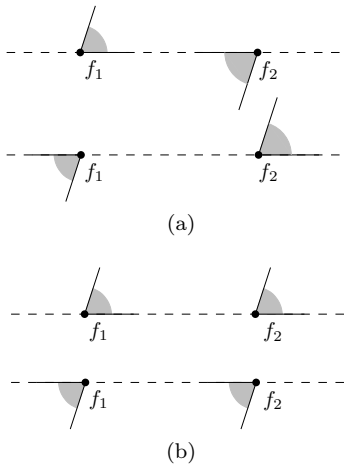


Figure 1: Proof of Lemma 1: (a) Case where f_1 and f_2 belong a same set F_i. (b) Case where $f_1 \in F_1$ and $f_2 \in F_k$.

Suppose now that $f_1 \in F_1$ and $f_2 \in F_k$. Configure both f_1 and f_2 with angle zero (see top of Figure 1b). At time $t = \pi$ the configuration of f_1 and f_2 is as shown in the bottom of Figure 1b. Since all elements of P belong to the segment connecting f_1 and f_2 then all elements of P are illuminated during intervals $[0, \alpha]$ and $[\pi, \pi + \alpha]$, 2α time in total.

For the second part of the lemma, let f and g be the two lights that cover P. Clearly $\alpha \geq \pi$. Assume, w.l.o.g., that $g \notin F_1 \cup F_k$ and that f is to the left of g. Also, let $Q \subseteq P$ denote the set of targets to the right of f. Light f alone can cover Q for an interval of length α, while g alone can cover Q for $\alpha - \pi$ only. Since $\alpha < 3\pi/2$, an interval of length $2\pi - \alpha > \pi/2$ is unaccounted for by f, but g can pick up at most $\alpha - \pi < \pi/2$ of this. Hence, f and g must belong to the same set. The claim does not hold if $\alpha \geq 3\pi/2$. □

Lemma 2 *Three floodlights f_1, f_2, and f_3 with illumination angle $\alpha < \pi$, can be configured so that, together, they cover the whole line $\mathcal{L}$ (hence, P) during 2α time in total.*

Proof. The proof can be obtained from [2]. Assume, w.l.o.g., that f_1, f_2, and f_3 appear in this order from left to right. Configure f_1, f_2, and f_3 with angle zero, π, and zero, respectively (top of Figure 2). At time $t = \pi$ the configuration is as shown at the bottom of Figure 2. During interval $[0, \alpha]$ the line is illuminated by f_1 and f_2, and during interval $[\pi, \pi + \alpha]$, by f_2 and f_3. The result follows. □

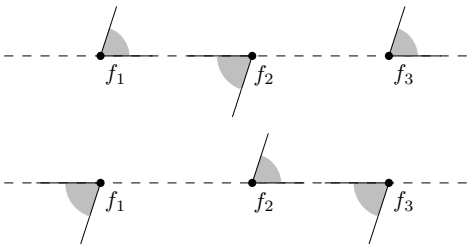


Figure 2: Proof of Lemma 2

Theorem 3 *If all floodlights belong to the same set among $F_1 \cup F_k, F_2, F_3, \ldots, F_{k-1}$, then $\alpha(P, F) = 2\pi/m$, which is optimal. Otherwise, $\alpha(P, F)$ satisfies:*

$$\alpha(P, F) \leq \min\left\{\frac{3\pi}{m}, \frac{2\pi}{m - Q + 2\lfloor \frac{Q}{3} \rfloor}\right\} \qquad (1)$$

where Q denotes the number of odd numbers in the set $\{m_1 + m_k, m_2, \ldots, m_{k-1}\}$.

Proof. Obviously $\alpha(P, F) \geq 2\pi/m$ in all cases. All lights belong to the same set iff $k = 2$, or $k = 3$ and $m_1 = m_3 = 0$. In both cases, illumination angle $\alpha = 2\pi/m$ is sufficient. Assume first that $k = 2$ and let $F_1 = \{f_1, \ldots, f_{m_1}\}$ and $F_2 = \{f'_1, \ldots, f'_{m_2}\}$. Floodlight f_i is configured with angle $(i-1)\alpha$ for $i = 1, \ldots, m_1$, and floodlight f'_j, with angle $\pi - j\alpha$ for $j = 1, \ldots, m_2$ (see Figure 3). Then, at any time $t \in [0, m_1\alpha)$ P is covered by a member of F_1 and, at any time $t \in [m_1\alpha, 2\pi)$, by a member of F_2. Assume now that $k = 3$ and $m_1 = m_3 = 0$. Then, $F = F_2 = \{f_1, \ldots, f_m\}$. By configuring f_i with angle $2i\pi/m$, P is covered by F. This proves the optimal result when all lights belong to the same set.

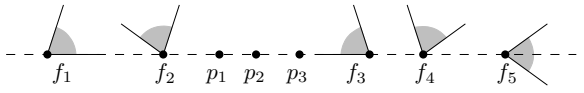


Figure 3: Two groups of floodlights $F_1 = \{f_1, f_2\}$ and $F_2 = \{f'_1, f'_2, f'_3\} = \{f_3, f_4, f_5\}$, where $m_1 = 2$ and $m_2 = 3$, and the configuration with angle $\alpha = 2\pi/5$.

If neither of the two cases above occurs, we configure the floodlights to satisfy inequality (1) as follows. We consider the case $m_1 = m_k = 0$ and $m_i = 1$ $(i = 2, \ldots, k-1)$ separately because the result follows immediately from [2], since our problem is equivalent to illuminating the whole x-axis. In this case $m = Q$ and $\alpha(P, F) = 3\pi/m = \min\{\frac{3\pi}{m}, \frac{2\pi}{m-Q+2\lfloor \frac{Q}{3} \rfloor}\}$ which is also optimal.

For the remaining cases, we can only obtain an upper bound on the optimal illumination angle. We proceed as follows. Pair the elements of F into $N = \lfloor \frac{m_1+m_k}{2} \rfloor + \lfloor \frac{m_2}{2} \rfloor + \ldots + \lfloor \frac{m_{k-1}}{2} \rfloor = \frac{m-Q}{2}$ pairs $(f_{1,1}, f_{1,2})$, $(f_{2,1}, f_{2,2})$, $\ldots, (f_{N,1}, f_{N,2})$ so that the elements of each pair belong to a same set among $F_1 \cup F_k, F_2, F_3, \ldots, F_{k-1}$. Group the remaining Q floodlights into $M = \lfloor \frac{Q}{3} \rfloor$ triples $(f'_{1,1}, f'_{1,2}, f'_{1,3}), \ldots, (f'_{M,1}, f'_{M,2}, f'_{M,3})$ (leaving at most two ungrouped). Let $\alpha = \frac{2\pi}{m-Q+2\lfloor \frac{Q}{3} \rfloor} = \frac{2\pi}{2N+2M}$. We now schedule the floodlights as follows. Configure $f_{i,1}$ and $f_{i,2}$ with angles $(i-1)\alpha$ and $\pi + (i-1)\alpha$, respectively, for $i = 1, \ldots, N$; and configure $f'_{j,1}$, $f'_{j,2}$, and $f'_{j,3}$ with angles $(N+j-1)\alpha$, $(N+j-1)\alpha$, and $\pi + (N+j-1)\alpha$, respectively, for $j = 1, \ldots, M$. Finally, arbitrarily configure the remaining floodlights (at most two). The correctness of this configuration follows from lemmas 1 and 2. □

3 Many points and two lights

In this section we consider the case of two floodlights f_1 and f_2, i.e., $m = 2$. Let $p_1, \ldots, p_n$ denote the elements of P. Assume w.l.o.g. that line $\ell(f_1, f_2)$ is horizontal and that f_1 is located to the left of f_2. Given any target point p_i $(i = 1, \ldots, n)$, let $\theta_i \in [0, \pi)$ denote the angle at p_i in the triangle $\triangle p_i f_1 f_2$. If there are points from P on both sides of the line $\ell(f_1, f_2)$, then we define two angles β^+ and β^- as the maximum of θ_i over all points p_i above and below $\ell(f_1, f_2)$, respectively (see Figure 4a). Otherwise, all the points of P are on the same side of $\ell(f_1, f_2)$ and we define two angles, β_{max} and β_{min}, as the largest and smallest θ_i over all points p_i, respectively (see Figure 4b).

Theorem 4 (Two floodlights) *For $m = 2$, $n \geq 2$:*
(1) If there are points of P on both sides of $\ell(f_1, f_2)$ then $\alpha(P, F) = \pi + \frac{\beta^+ + \beta^-}{2}$.
(2) If all the points of P lie on one side of $\ell(f_1, f_2)$ then $\alpha(P, F) = \pi + \frac{\beta_{max} - \beta_{min}}{2}$.

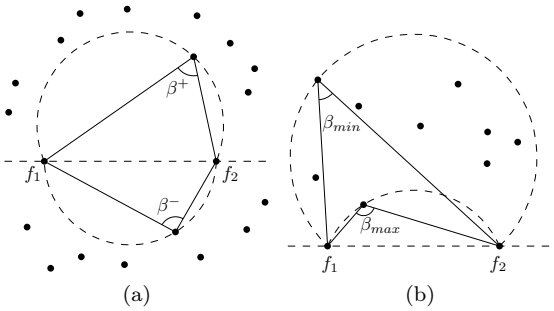


Figure 4: (a) β^+ and β^-. (b) β_{max} and β_{min}.

Proof. First, we prove part (1) of the theorem. We configure floodlights f_1 and f_2 initially as follows. Let Af_1Bf_2 be the quadrilateral such that angle $\angle f_1Af_2$ is equal to β^+, angle $\angle f_2Bf_1$ is equal to β^-, and points f_1 and f_2 are symmetric with respect to the line $\ell(A, B)$ as shown in Figure 5a.

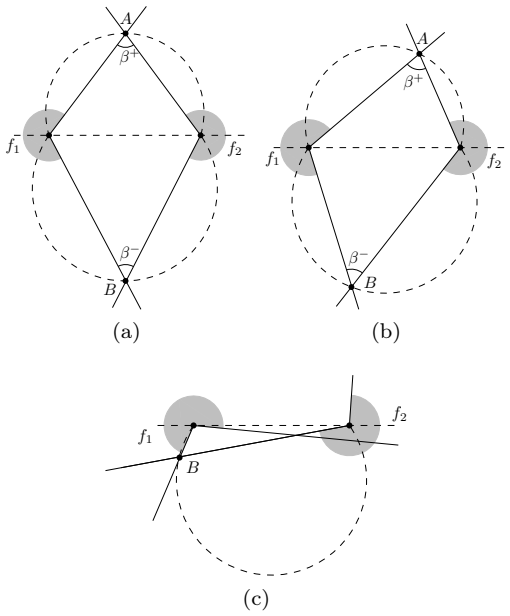


Figure 5: (a) Initial position. (b),(c) General position.

Since f_1 and f_2 rotate at unit speed, it always holds that $\angle f_1Af_2 = \beta^+$ and $\angle f_2Bf_1 = \beta^-$ (see Figure 5b). Furthermore, the region not illuminated by the lights is always a subset of the interior of the union of the triangles $\triangle f_1Af_2$ and $\triangle f_2Bf_1$ (see Figures 5b and 5c), and it never contains points of P by the definition of β^+ and β^-. It remains to show that any illumination angle smaller than $\pi + \frac{\beta^+ + \beta^-}{2}$ is not feasible.

Suppose that, initially, floodlight $f_i, i = 1, 2$ covers angles in the interval $[\alpha_i, \beta_i]$. First we show that these intervals cover all possible directions in $[0, 2\pi]$. If, to the contrary, a direction t is not covered by $[\alpha_1, \beta_1] \cup [\alpha_2, \beta_2]$, then there is a rotation such that p_i is not illuminated, where p_i is the point above the x-axis and $\angle f_2 p_i f_1 = \beta^+$, a contradiction. Therefore, the

floodlight intervals overlap as shown in Figure 6a.

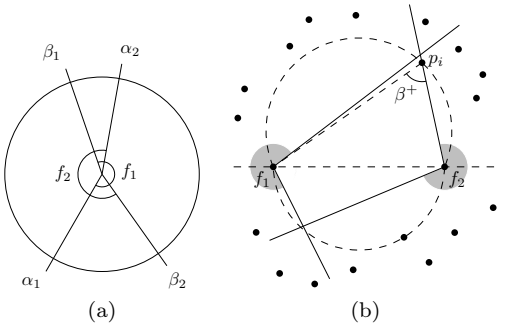


Figure 6: The floodlight intervals.

We show now that the overlapping interval $[\alpha_2, \beta_1]$ has length at least β^+. Suppose to the contrary that it is smaller than β^+. Consider the rotation by the angle γ such that the β_2-ray of floodlight f_2 passes through the point p_i defining β^+ (see Figure 6b). Then the α_1-ray of floodlight f_1 will not cover p_i since the angle between the two rays is less than β^+. Therefore p_i is not covered if the rotation angle is slightly smaller than γ, a contradiction. Similarly, the overlapping interval $[\alpha_1, \beta_2]$ has length at least β^-. If the illumination angle is α then $2\alpha \geq 2\pi + \beta^+ + \beta_i$. The claim of part (1) follows.

To prove part (2) of the theorem we configure floodlights f_1 and f_2 at the beginning as follows. Let Af_1Bf_2 be a quadrilateral such that $\angle f_1Af_2 = \beta_{min}$ and $\angle f_2Bf_1 = \beta_{max}$ and points f_1 and f_2 are symmetric about line $\ell(A, B)$ as shown in Figure 7a. The argument is similar to the proof of part (1) since the points A and B move along arcs shown in Figure 7a. The uncovered part is either a region below Cf_1Bf_2D, shown in Figure 7a, or the wedge XYZ, shown in Figure 7c. In any case the area between the two arcs defined by β_{min} and β_{max} is always illuminated.

The optimality of angle $\pi + \frac{\beta_{max}-\beta_{min}}{2}$ can be shown similarly to the proof of part (1). □

4 Many lights and two points

Let p_1 and p_2 denote the elements of P and $f_1, \ldots, f_m$ denote the elements of F. Let θ_i $(i = 1, \ldots, m)$ denote the angle by which line $\ell(f_i, p_1)$ has to be rotated clockwise with center f_i to become line $\ell(f_i, p_2)$. Assume w.l.o.g. that $\theta_1 \leq \theta_2 \leq \ldots \leq \theta_m$.

Lemma 5 *If $n = 2$ then $\alpha(P, F) \leq \frac{2\pi}{m} + \frac{\theta_m - \theta_1}{m}$.*

Proof. It suffices to prove that for $\alpha = \frac{2\pi}{m} + \frac{\theta_m-\theta_1}{m}$ the m floodlights can be configured properly. Configure f_1 arbitrarily and, for $i = 1, \ldots, m-1$, configure f_{i+1} to start illuminating point p_2 at the time f_i stops illuminating it. Since $\alpha \geq 2\pi/m$ then p_2 is always illuminated. Observe that p_1 is illuminated at some time by both f_i and f_{i+1} since $\theta_i \leq \theta_{i+1}$. Then the time spam in which p_1 is illuminated by f_i and not by f_{i+1} is equal to $\alpha + \theta_i - \theta_{i+1}$. Since $\sum_{i=1}^{m-1}(\alpha + \theta_i - \theta_{i+1}) + \alpha = m\alpha + \theta_1 - \theta_m = 2\pi$ then p_1 is always illuminated. □

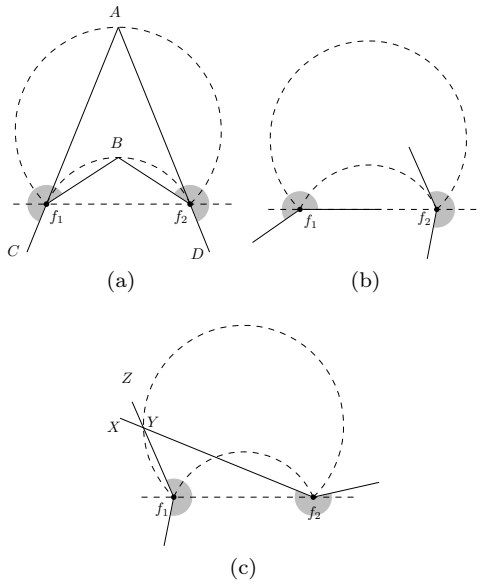


Figure 7: The floodlight intervals.

One can build examples, as the following one, in which $\theta_1 < \theta_m$ and α is the theoretical minimum (i.e. $\alpha = 2\pi/m$), showing that $\alpha = \frac{2\pi}{m} + \frac{\theta_m-\theta_1}{m}$ is not always optimal. Let $m = 8$ and $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7, \theta_8) = (\frac{\pi}{4}, \frac{\pi}{4}, \frac{\pi}{2}, \frac{\pi}{2}, \frac{\pi}{2}, \frac{\pi}{2}, \frac{3\pi}{4}, \frac{3\pi}{4})$. Then $f_1, \ldots, f_8$ can be configured with $\alpha = 2\pi/m$ so that p_1 is illuminated in the (circular) order $f_1, f_3, f_8, f_2, f_4, f_5, f_6, f_7$ and p_2 in the order $f_6, f_1, f_7, f_3, f_2, f_8, f_4, f_5$. Thus, it becomes interesting to decide whether $\alpha(P, F) = 2\pi/m$ when $\theta_1, \ldots, \theta_m$ are all multiples of $2\pi/m$.

Acknowledgements. The problem studied here was introduced and partially solved during the VI Spanish Workshop on Geometric Optimization, June 2012, El Rocío, Huelva, Spain. The authors would like to thank other participants for helpful comments.

References

[1] J. Bose, L. Guibas, A. Lubiw, M. Overmars, D. Souvaine, and J. Urrutia. The floodlight illumination problem. *Int. J. Comp. Geom.*, 7:153–163, 1997.

[2] E. Kranakis, F. MacQuarie, O. Morales, and J. Urrutia. Uninterrupted coverage of a planar region with rotating directional antennae. In *ADHOC-NOW*, pages 56–68, 2012.

[3] K. Sugihara, I. Suzuki, and M. Yamashita. The searchlight scheduling problem. *SIAM J. Comput.*, 19(6):1024–1040, 1990.

[4] J. Urrutia. Art gallery and illumination problems. In *Hdbk. of Comp. Geom.*, pages 973–1027, 2000.

Some results on open edge guarding of polygons

Antonio L. Bajuelos*[1], Santiago Canales[2], Gregorio Hernández†[3], Mafalda Martins*‡[1], and Inês Matos*§[1]

[1]Universidade de Aveiro & CIDMA, Portugal
[2]Universidad Pontificia Comillas de Madrid, Spain
[3]Universidad Politécnica de Madrid, Spain

Abstract

This paper focuses on a variation of the Art Gallery problem that considers open edge guards. The "open" prefix means the endpoints of an edge where a guard is are not taken into account for visibility purposes. This paper studies the number of open edge guards that are sufficient and sometimes necessary to guard some classes of simple polygons.

Introduction

The well known Art Gallery problem studies the minimum number of guards that are needed to fully cover a polygon P, that is, the number of guards from which every point of P is visible. Ideally, guards may be placed anywhere on P but usually they are restricted to vertices of the polygon or its edges. In the first case such guards are called vertex guards and in the second edge guards. Moreover, a point guard is a guard that can be placed anywhere on the polygon. Lee et al. [3] proved that finding the minimum number of guards to fully cover a polygon without holes is NP-hard for all three variations of guards. Toussaint conjectured that $\lfloor \frac{n}{4} \rfloor$ edge guards are sufficient to cover any simple polygon of n vertices, except for small values of n. Later, Shermer [4] proved that $\lfloor \frac{3n}{10} \rfloor$ edge guards are sufficient to cover any simple polygon, except for $n = 3, 6, 13$ where an extra edge guard might be needed. Shermer actually proved a combinatorial result: any triangulation of a polygon with n vertices can be dominated by $\frac{3n}{10}$ edge guards.

In this paper guards are assumed to be placed along open edges of a polygon, that is, the endpoints of any edge are not taken into account for visibility purposes. Therefore, a point p is covered by an edge e if p is visible from some interior point of e. As shown in Figure 1, open edge guards can see considerably less polygon area than the usual edge guards, and are therefore an interesting topic of research on their own.

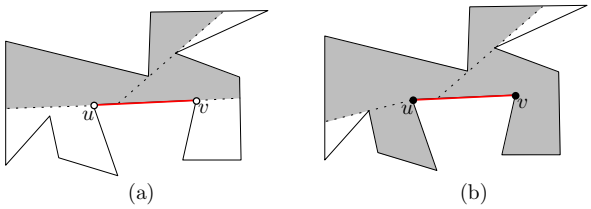


Figure 1: (a) The area covered by the open edge guard $\overline{uv}$ is shown in grey. (b) The area covered by the closed edge guard $\overline{uv}$ is shown in grey.

Open edge guarding is a variation of the Art Gallery problem that was first introduced by Viglietta [6] as a way to guard 3D polyhedra. This work was built on by Benbernou et al. [7] and Tóth et al. [5]. The latter studied open edge guards and proved that there are polygons that need at least $\lfloor \frac{n}{3} \rfloor$ guards to be covered, and that $\lfloor \frac{n}{2} \rfloor$ are always sufficient.

This article is structured in the following way. Section 1 introduces the concept of open edge guards and presents results on the number of guards that cover orthogonal and spiral polygons. Some results related to the Fortress problem on simple and orthogonal polygons are also presented. The paper concludes with Section 2.

1 Open edge guards

Given a simple polygon P, $\mathcal{G}_{OE}(P)$ is the minimum number of open edge guards that fully cover P and let $\mathcal{G}_{OE}(n) := \min\{\mathcal{G}_{OE}(P) : P \text{ is a polygon of } n \text{ vertices}\}$. Consequently, this section is devoted to calculate $\mathcal{G}_{OE}(n)$ for different classes of polygons.

*Research supported by *FEDER* funds through *COMPETE*–Operational Programme Factors of Competitiveness, CIDMA and FCT within project PEst-C/MAT/UI4106/2011 with COMPETE number FCOMP-01-0124-FEDER-022690.

†Research supported by ESF EUROCORES programme EuroGIGA - ComPoSe IP04 - MICINN Project EUI-EURC-2011-4306.

‡Research also supported by FCT grant SFRH/BPD/66431/2009.

§Email: ipmatos@ua.pt. Research also supported by FCT grant SFRH/BPD/66572/2009.

1.1 Orthogonal polygons

Bjorling-Sachs [1] proved that $\lfloor \frac{3n+4}{16} \rfloor$ closed edge guards are sufficient and sometimes necessary to fully cover an orthogonal polygon. This section shows that $\lfloor \frac{n}{4} \rfloor$ open edge guards are sometimes necessary and always sufficient to fully cover an orthogonal polygon.

Given an orthogonal polygon P with n vertices, the edges of P can be divided into four categories: north, south, west and east edges. North edges see the interior of the polygon below them, south edges see it above them, east edges see it to their left and west edges see it to their right. Each of these four sets represents a group of open edge guards that completely covers P. In order to see this, choose any point p of P. From p, it is always possible to draw vertical segments that will hit a north edge if it goes up from p and a south edge if it goes down. The reasoning for the horizontal directions is similar. Therefore, the smallest of these four sets of edges proves the upper bound: any orthogonal polygon can be covered by $\lfloor \frac{n}{4} \rfloor$ open edge guards. Furthermore, there is an example of an orthogonal polygon that needs $\lfloor \frac{n}{4} \rfloor$ open edge guards to be fully covered. Such polygon is very similar to the one depicted in Figure 2, but where each spike only hides one point since there are no holes. These two bounds prove the following theorem.

Theorem 1 *Any orthogonal polygon with n vertices can be covered by $\lfloor \frac{n}{4} \rfloor$ open edge guards, and in some cases this number is necessary.*

Observe that this result essentially holds for orthogonal polygons with holes, and the upper bound can be obtained in the same way it was explained above. In Figure 2 there is an example of a polygon with holes that needs $\lfloor \frac{n}{4} \rfloor - 1$ open edge guards to be fully covered, since each marked point is seen by a different open edge guard.

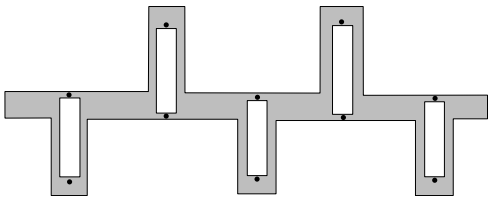


Figure 2: A polygon with holes that needs $\lfloor \frac{n}{4} \rfloor - 1$ open edge guards to cover it, $n = 44$.

1.2 Spiral polygons

This section studies open edge guarding of spiral polygons, which will also be called spirals when it eases the reading of the text. According to a previous work, $\lfloor \frac{n+2}{5} \rfloor$ closed edge guards are sufficient and sometimes necessary to cover spiral polygons [8].

1.2.1 Tight bound on the number of open edge guards

In the example in Figure 3(a), each point marked on the polygon needs a different open edge guard to cover it. Since this spiral has only one possible triangulation and there is one of these points per four triangles, this polygon needs $\lceil \frac{n-2}{4} \rceil$ open edge guards in order to be fully covered. Therefore, $\mathcal{G}_{OE}(n) \geq \lceil \frac{n-2}{4} \rceil$. This lower bound can be rewritten as $\lfloor \frac{n+1}{4} \rfloor$, and it is proven below that this bound is tight.

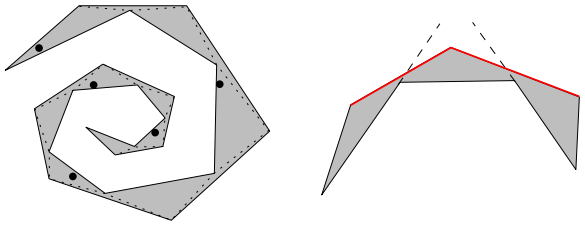


Figure 3: (a) No two of the marked points can be covered by the same open edge guard. (b) The two red open edge guards cover the whole spiral.

The boundary of each spiral can be decomposed into a sequence of consecutive reflex vertices (called a reflex chain) and a sequence of consecutive convex vertices (called a convex chain). The proof that any spiral can be covered by $\lfloor \frac{n+1}{4} \rfloor$ open edge guards uses induction on the number of edges of the polygon.

The base case comprehends four cases. When the spiral has four, five or six edges it is easy to see that one open edge guard covers the whole polygon. Spirals with seven edges can be covered by two open edge guards, since it suffices to place the guards on the edges of the convex chain that are intersected by the extensions of the first and last edges of the reflex chain (see Figure 3(b)).

For the inductive step, suppose $\lfloor \frac{n+1}{4} \rfloor$ open edge guards are sufficient to cover any spiral of n' vertices with $n' < n$ edges, $n > 7$. Let P be a spiral with $n > 7$ vertices, whose reflex chain is formed by the vertices $\{r_1, r_2, \ldots, r_k\}$. Now extend the edge $\overline{r_1 r_2}$ until it intersects some edge of the convex chain. Let v be the rightmost endpoint of the edge of the convex chain just intersected as shown in Figure 4(a). The proof is now divided into four cases depending on the number of edges of the convex chain from c_1 to v: (a) five or more than five edges, (b) four edges, (c) three edges and (d) two edges. For case (a), suppose the convex chain from c_1 to v has at least five edges. Then draw the diagonal between r_1 and c_5, the fifth vertex of the convex chain. In this way, the spiral is partitioned into two spiral polygons: P' that has six edges and so can be guarded with one open edge guard and P'' with $n-4$ edges. For case (b), suppose the convex chain from c_1 to v has four edges as shown

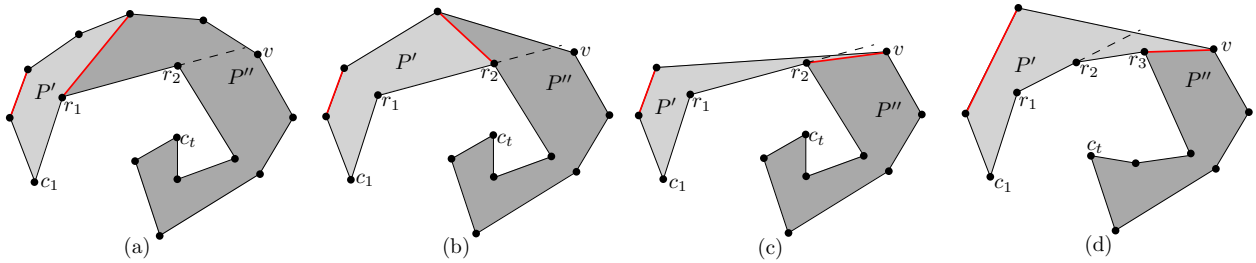


Figure 4: (a) The convex chain from c_1 to v is formed by six edges. (b) The convex chain from c_1 to v is formed by four edges. (c) The convex chain from c_1 to v is formed by three edges: the first reflex vertex visible from v is r_2; (d) the first reflex vertex visible from v is r_3.

in Figure 4(b). Then draw the diagonal between r_2 and c_4, the fourth vertex of the convex chain. This partitions the spiral into two spiral polygons: P' that has six edges and therefore can be guarded with one open edge guard and P'' with $n-4$ edges.

In case (c) the convex chain from c_1 to v has three edges and the situation is slightly different. As shown in Figures 4(c) and 4(d), draw the diagonal between v and the first visible reflex vertex starting from r_2 (note that r_2 can be such a vertex). This procedure partitions the spiral into two polygons: P' that can be guarded with one open edge guard and P'' with at most $n-4$ edges.

Finally, case (d) in which the convex chain from c_1 to v has only two edges. In this case, draw the diagonal from v to the first visible reflex vertex after r_2. If there are no visible reflex vertices left, then the reflex chain is over and an open edge guard placed on the second edge of the convex chain covers the whole spiral (see Figure 5(a)). If there is one visible reflex vertex then draw the diagonal as before, which will partition the spiral into two polygons: P' that can be guarded with one open edge guard and P'' with at most $n-4$ edges (see Figure 5(b)).

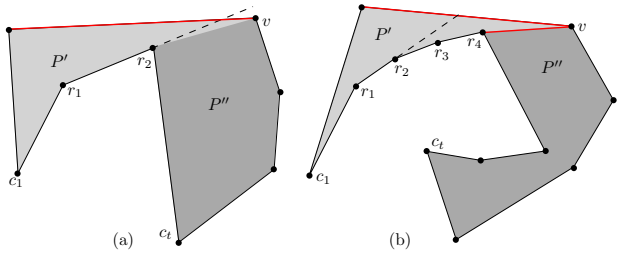


Figure 5: The convex chain from c_1 to v has two edges. (a) There is no reflex vertex visible from v besides r_2. (b) The first reflex vertex visible from v is r_4.

All the four cases described above end with a polygon P'' that has at most $n-4$ edges, which means the inductive hypothesis can be applied. Therefore, P'' can be covered by $\lfloor \frac{(n-4)+1}{4} \rfloor = \lfloor \frac{n+1}{4} \rfloor - 1$ open edge guards. Since polygon P' is covered exactly by one open edge guard, the whole spiral is covered by $\lfloor \frac{n+1}{4} \rfloor$ open edge guards and this concludes the proof.

Theorem 2 *Any spiral polygon with n vertices can be covered by $\lfloor \frac{n+1}{4} \rfloor$ open edge guards, and in some cases this number is necessary.*

1.2.2 Placing the minimum number of open edge guards

This section presents an algorithm to place the minimum number of open edge guards that cover a spiral polygon P. The main idea of the algorithm is to build two sets simultaneously: G, which is the set of open edge guards, and $H = \{h_1, h_2, \ldots\}$ that is the set of points that guarantees G is of minimum size. The points that form set H are placed on the polygon in such a way that each open edge guard can see only one of them and is therefore associated with it. Consequently, $|G| = |H|$. Let $\{r_1, r_2, \ldots, r_k\}$ be the reflex chain and $\{c_1, c_2, \ldots, c_{n-k}\}$ the convex chain of P. Moreover, let $\{c_1, c_2, \ldots, c_{n-k}, r_k, r_{k-1}, \ldots, r_1\}$ be the sequence of n vertices of a spiral polygon P. The steps of the algorithm to place the minimum number of open edge guards to cover P are depicted in Figure 6 and explained in the following.

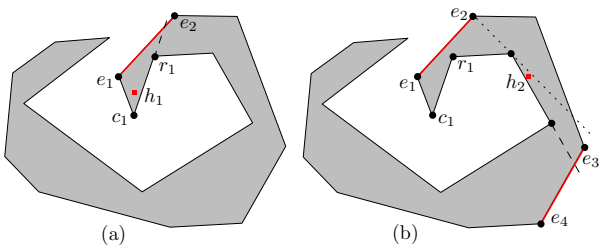


Figure 6: (a) Finding an edge of the convex chain that covers point h_1. (b) Finding an edge of the convex chain that covers point h_2.

Set G is empty to start with. Let $h_1 \in P$ be a point very close to c_1, which has to be covered. Draw the ray $\overrightarrow{c_1 r_1}$ that will intersect some edge of the convex chain that sees point h_1. Let such edge be denoted by (e_1, e_2) and assign $G \leftarrow \{(e_1, e_2)\}$. Secondly, find

the last reflex vertex r_j that can be seen from e_2 and consider point h_2, which is very close to r_j along the edge (r_j, r_{j+1}). Then draw the ray $\overrightarrow{r_j r_{j+1}}$ that will intersect some edge of the convex chain that sees point h_2. Let such an edge be denoted by (e_3, e_4) and assign $G \leftarrow G \cup \{(e_3, e_4)\}$. Repeat this last step until all reflex vertices and c_{n-k} are guarded.

This algorithm selects the edges $\overline{e_j e_{j+1}}$ of the convex chain that will be part of set G, which fully covers any spiral polygon since it totally covers its convex chain.

Lemma 3 *The algorithm described in this section builds a set H of points interior to P such that $\mathcal{G}_{OE}(P) \geq |H|$.*

Theorem 4 *The algorithm described in this section places the minimum number of open edge guards needed to cover a spiral polygon in $\mathcal{O}(n)$ time.*

Proof. Let G be the set of open edge guards built by the algorithm to cover spiral polygon P and H the set of points of P in which each point is covered by a different guard. That is, the set of points that are placed in such way that the According to Lemma 3, $\mathcal{G}_{OE}(P) \geq |H|$ but since $|H| = |G|$ then $\mathcal{G}_{OE}(P) \geq |G|$ and therefore G is a minimum set of open edge guards. Regarding the time complexity, each edge of the convex chain is only processed once whilst analysing the rays $\overrightarrow{r_j r_{j+1}}$. In the same way, each edge of the reflex chain is checked once to find the last reflex vertex that is visible from the chosen edges. Consequently, each vertex of the spiral polygon is analysed just once by the algorithm and therefore it runs in linear time. □

1.3 Fortress problem

This section is devoted to another variation of the Art Gallery problem called the *Fortress Problem*. Instead of guarding the interior of a simple polygon, the Fortress Problem variation focuses on monitoring the exterior of a polygon. Choi et al. [2] proved that the exterior of any simple polygon can be covered by $\lceil \frac{n}{3} \rceil$ closed edge guards and that these guards are necessary to cover the exterior of convex polygons. In the case of open edge guards, this problem is trivial since it is easy to see that every edge will be needed as a guard to cover the exterior of a convex polygon.

The natural following step is to study orthogonal polygons. Again, Choi et al. [2] proved that the exterior of any orthogonal polygon can be covered by $\lfloor \frac{n}{4} \rfloor + 1$ edge guards and that this number can be necessary. The proof of the following theorem is omitted, but it is based on the technique of dividing the edges according to their orientation, as introduced in Section 1.1. The lower bound is given by the orthoconvex polygon depicted in Figure 7.

Theorem 5 *The exterior of any orthogonal polygon with n vertices can be covered by $\frac{n}{2} + 2$ open edge guards, and in some cases this number is necessary.*

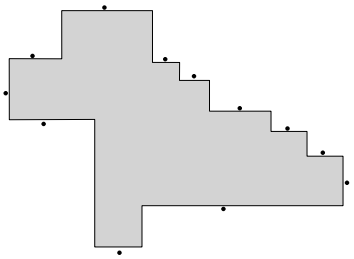


Figure 7: Orthoconvex polygon that needs $\frac{n}{2} + 2$ open edge guards to be covered.

2 Final remarks

We studied other classes of polygons, such as monotone polygons, as well as other geometric configurations. There are monotone polygons that need $\lfloor \frac{n}{3} \rfloor$ open edge guards in order to be fully covered and we believe this bound is tight. Furthermore, this bound is proved for open mobile guards, which can patrol edges and diagonals of a polygon. This type of coverage has also been studied for other polygons.

References

[1] I. Bjorling-Sachs. Edge guards in rectilinear polygons. *Comput. Geom. Theory Appl.*, 11(2):111–123, 1998.

[2] A. Choi and S. Yiu. Edge guards for the fortress problem. *Journal of Geometry*, 72:47–64, 2001.

[3] D. Lee and A. Lin. Computational complexity of art gallery problems. *IEEE Transactions on Information Theory*, 32(2):276–282, 1986.

[4] T. Shermer. Recent results in art galleries. *Proceedings of the IEEE*, 80(9):1384–1399, 1992.

[5] C. D. Tóth, G. T. Toussaint, and A. Winslow. Open guard edges and edge guards in simple polygons. In A. Márquez, P. Ramos, and J. Urrutia, editors, *Computational Geometry*, volume 7579 of *Lecture Notes in Computer Science*, pages 54–64. Springer Berlin Heidelberg, 2012.

[6] G. Viglietta. *Guarding and Searching Polyhedra*. PhD thesis, University of Pisa, 2012.

[7] G. Viglietta, N. Benbernou, E. D. Demaine, M. L. Demaine, A. Kurdia, J. O'Rourke, G. T. Toussaint, and J. Urrutia. Edge-guarding orthogonal polyhedra. In 23^{rd} *Canadian Conference on Computational Geometry*, 2011.

[8] S. Viswanathan. Tight bounds for the number of edge guards for spiral polygons. *Journal of Geometry*, 51:178–186, 1994.

Guarding the vertices of thin orthogonal polygons is NP-hard

Ana Paula Tomás*

DCC & CMUP, Faculdade de Ciências
Universidade do Porto, Portugal

Abstract

An orthogonal polygon of P is called "thin" if the dual graph of the partition obtained by extending all edges of P towards its interior until they hit the boundary is a tree. We show that the problem of computing a minimum guard set for the vertices of a thin orthogonal polygon is NP-hard either for guards lying on the boundary, or on vertices or anywhere in the polygon.

Introduction

The classical art gallery problem for a polygon P asks for a minimum set of points $\mathcal{G}$ in P such that every point in P is seen by at least one point in $\mathcal{G}$ (the guard set). Many variations of art gallery problems have been studied over the years to deal with various types of constraints on guards and different notions of visibility. In the general visibility model, two points p and q in a polygon P *see* each other if the line segment $\overline{pq}$ contains no points of the exterior of P. The set $V(v)$ of all points of P visible to a point $v \in P$ is the *visibility region* of v. A guard set $\mathcal{G}$ for a set $S \subseteq P$ is a set of points of P such that $S \subseteq \cup_{v \in \mathcal{G}} V(v)$. Two points v_i and v_j are equivalent for the visibility relation if $V(v_i) \cap S = V(v_j) \cap S$. If $V(v_j) \cap S \subset V(v_i) \cap S$ then v_i strictly dominates v_j, and v_i can replace v_j in an optimal guard set of S. Guards that may lie anywhere inside P are called *point guards* whereas *vertex guards* are restricted to lie on vertices and *boundary guards* on the boundary. Combinatorial upper and lower bounds on the number of necessary guards are known for specific settings (for surveys, refer to e.g. [8, 10]). The fact that some art gallery problems are NP-hard [5, 9] motivates the design of heuristic and metaheuristic methods for finding approximate solutions and also the study of more specific classes of polygons where some guarding problems may be tractable [1, 2, 3, 6]. In this paper, we address the problem of guarding *the vertices* of orthogonal polygons, which is known to be NP-hard for generic orthogonal polygons [4]. We show that the problem is NP-hard also for the family of *thin orthogonal polygons*, which consists of the orthogonal polygons such that the dual graph of the corresponding *grid partition* $\Pi_{HV}(P)$ is a tree. $\Pi_{HV}(P)$ is obtained by adding all horizontal and vertical cuts incident to the reflex vertices of P (see Fig. 1). Our proof is inspired in [4]

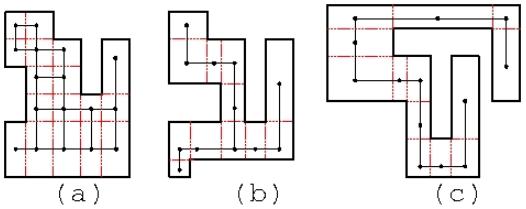


Figure 1: Orthogonal polygons, grid partitions and dual graphs: (a) $\Pi_{HV}(P)$ and its dual graph in general; (b) a thin orthogonal polygon; (c) a thin orthogonal polygon that is a path orthogonal polygon.

although the need to obtain thin orthogonal polygons led to novel aspects in the construction. The class of thin orthogonal polygons contains the class of thin polyomino trees introduced in [1], for which the authors conjecture that the guarding problem under the general visibility model has a polynomial-time (exact) algorithm. To the best of our knowledge, this problem is open. In [12], we give a linear-time algorithm for computing an optimal vertex guard set for any given path orthogonal polygon (for which the dual graph of $\Pi_{HV}(P)$ is a path graph), and prove tight lower and upper bounds of $\lceil n/6 \rceil$ and $\lfloor n/4 \rfloor$ for the optimal solution for the subclass where all horizontal and vertical cuts intersect the boundary at Steiner points. Since the *thin grid orthogonal polygons* belong to this class, our work extends results previously known for the spiral thin grid orthogonal polygons and the MinArea grid orthogonal polygons [6] (for which the minimum vertex guard sets have exactly $\lfloor n/4 \rfloor$ and $\lceil n/6 \rceil$ guards) and somehow explains why the MinArea grid orthogonal polygons were considered representative of extremal behaviour [11]. The result that supports our proof allows us to conclude that a minimum guard set for the vertices of a path orthogonal polygon can be found in linear-time.

*Email: apt@dcc.fc.up.pt. Research partially supported by the European Regional Development Fund through the programme COMPETE and by the Portuguese Government through the FCT – Fundação para a Ciência e Tecnologia under the project PEst-C/MAT/UI0144/2011.

In rest of the paper, we show that computing a minimum guard set for *the vertices* of a thin orthogonal polygon (GVTP) is NP-hard, either for boundary guards, vertex guards or point guards.

1 Hardness for boundary guards

Theorem 1 GVTP *for thin orthogonal polygons is NP-hard for boundary guards.*

For the proof, we define a reduction from the vertex-cover problem in graphs (VC) to GVTP with boundary-guards. VC, known to be NP-complete, is the problem of deciding whether a graph $G = (V, E)$ has a vertex-cover S of size $|S| \leq k$, for k integer. A vertex-cover of G is a subset $S \subseteq V$ such that for each edge $(u, v) \in E$, either $u \in S$, or $v \in S$, or both.

The thin orthogonal polygon we construct for a given graph $G = (V, E)$ is a large square with $|E|$ tiny d-gadgets attached to its bottom. In Fig. 2 we sketch this construction and in Fig. 3 we present the *double gadget* (d-gadget) defined for the proof.

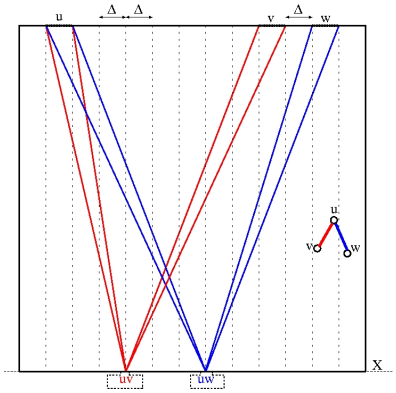


Figure 2: From VC to GVTP with boundary guards: the representation of $G = (\{u, v, w\}, \{(u, v), (u, w)\})$; the edges of G are mapped to the points uv and uw (that will be replaced by tiny d-gadgets) and the vertices are mapped to the segments u, v and w.

We define the side-length of this square to be $L\Delta$, with $L = 1 + 2|V| + 3|E|$ and $\Delta = 10L$. Considering $V = \{v_1, v_2, \ldots, v_n\}$ sorted, we denote by E_i^+ the subset of all edges $(v_i, v_j) \in E$ such that $i < j$, also sorted by increasing value of j. In the construction we follow these orderings: for each i, we represent v_i by a segment of length Δ on the top edge of the square and the edges in E_i^+ as middle points of $|E_i^+|$ consecutive segments of length 2Δ on the bottom edge, placed between the projections of v_i and v_{i+1}, and with separation gaps of length Δ between each other. The square is implicitly divided into L slabs of length Δ, and we leave the first slab empty and an empty slab between consecutive items.

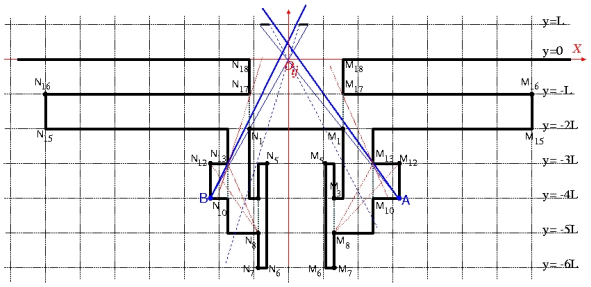


Figure 3: A sketch of the d-gadget Ξ_{ij} defined for the edge (v_i, v_j). The vertices on the left side are labelled from N_1 to N_{18} in CW order and on the right side are labelled from M_1 to M_{18} in CCW order. $B \equiv N_{11}$ and $A \equiv M_{11}$ are the two distinguished vertices.

The d-gadget associated to the edge $(v_i, v_j) \in E_i^+$, denoted by Ξ_{ij}, is defined as follows. Let O_{ij} be the point that represents the edge (v_i, v_j) and $\overline{A_iB_i}$ and $\overline{A_jB_j}$ the segments associated to v_i and v_j. Together with O_{ij}, these segments define two visibility cones with apex O_{ij}. By a slight perturbation, we can decouple the two cones, and move the new apexes to the distinguished vertices (B and A) of a tiny d-gadget Ξ_{ij}. The structure of this gadget will fix segment $\overline{A_iB_i}$ (resp. $\overline{A_jB_j}$) as the portion of the boundary of the polygon that A (resp. B) sees above line X (i.e, above the gadget). We use VC instead of the minimum 2-interval piercing problem used in [4] in order to be able to control the aperture of visibility cones and also the structure of the thin orthogonal polygon obtained in the reduction. Some of the vertices of a d-gadget can only be guarded by a *local guard* (i.e., a guard below line X), for instance, the vertices M_{16}, M_{12}, M_8, M_7 and M_5 on its right part and N_{16}, N_{12}, N_8, N_7 and N_5 on the left part. For every d-gadget, at least three local boundary-guards will be needed to guard these vertices and no three such guards can see both A and B if they see all these vertices. Moreover, one can always locate three local boundary-guards that see all the gadget vertices other than A (namely, at N_8, N_1 and M_8) or other than B (namely, at N_8, M_1 and M_8). Another guard is required to guard the unguarded vertex but it does not need to be local. As we will see, this guard can be located on the portion of the top edge of the polygon seen from the unguarded vertex.

We define the coordinates of the vertices of Ξ_{ij} w.r.t. a cartesian system $\mathcal{R}_{O_{ij}}$ with origin at O_{ij}. By construction, the x-coordinates of the points A_i, B_i and O_{ij} w.r.t. a cartesian system fixed at the bottom left corner of the large square are given by

$$\begin{aligned} x'_{A_i} &= (2i - 1 + 3\textstyle\sum_{k<i} |E_k^+|)\Delta \\ x'_{B_i} &= x'_{A_i} + \Delta \\ x'_{O_{ij}} &= x'_{B_i} + 2\Delta + 3\Delta\,|E_i^+ \cap \{(v_i, v_{j'}) : j' < j\}| \end{aligned}$$

and, consequently, if we define x_i and x_j as $x_i = (x'_{O_{ij}} - x'_{B_i})/\Delta$ and $x_j = (x'_{A_j} - x'_{O_{ij}})/\Delta$, then, w.r.t. the cartesian system $\mathcal{R}_{O_{ij}}$, we have

$$\begin{array}{ll} A_i = (-(x_i+1)\Delta,\, L\Delta) & A_j = (x_j\Delta,\, L\Delta) \\ B_i = (-x_i\Delta,\, L\Delta) & B_j = ((x_j+1)\Delta,\, L\Delta) \end{array}$$

for integers $x_i \geq 2$ and $x_j \geq 2$. Then, we define A and B as the intersection points of the supporting lines of $\overrightarrow{O_{ij}A_i}$ and $\overrightarrow{O_{ij}B_j}$ with the line $y = -4L$, that is, as $A = (4x_i + 4, -4L)$ and $B = (-4x_j - 4, -4L)$. So, the rays $\overrightarrow{AA_i}$ and $\overrightarrow{BB_j}$ share the supporting lines of the initial rays $\overrightarrow{O_{ij}A_i}$ and $\overrightarrow{O_{ij}B_j}$. The aperture of the visibility cone $\mathcal{C}_A = cone(A, \overline{A_iB_i})$ is determined by the vertices M_1 and M_{13}. We selected M_1 as the intersection of $\overrightarrow{AA_i}$ with the line $y = -2L$ and M_{13} as the intersection of $\overrightarrow{AB_i}$ with the line $y = -3L$. Therefore, $M_1 = (2x_i+2, -2L)$ and $M_{13} = (\tau_i, -3L)$, with $\tau_i = 3x_i + 3 + \frac{\Delta}{\Delta+4}$, because the straight lines AA_i and AB_i are given by the following equations.

$$\begin{aligned} AA_i : y &= \frac{-L}{x_i+1}x \\ AB_i : y &= \frac{-L(\Delta+4)}{x_i(\Delta+4)+4}x + \frac{4L\Delta}{x_i(\Delta+4)+4} \end{aligned}$$

Similarly, the vertices N_1 and N_{13} determine the aperture of the visibility cone $\mathcal{C}_B = cone(B, \overline{A_jB_j})$, being $N_1 = (-2x_j - 2, -2L)$ the intersection of $\overrightarrow{BB_j}$ with $y = -2L$ and $N_{13} = (\tilde{\tau}_j, -3L)$ the intersection of $\overrightarrow{BA_j}$ with $y = -3L$, with $\tilde{\tau}_j = -3x_j - 3 - \frac{\Delta}{\Delta+4}$. The coordinates of the vertices of Ξ_{ij} are

$$\begin{array}{ll} M_1 = (2x_i+2, -2L) & M_2 = (2x_i+2, -4L) \\ M_3 = (2x_i+1, -4L) & M_4 = (2x_i+1, -3L) \\ M_5 = (2x_i, -3L) & M_6 = (2x_i, -6L) \\ M_7 = (2x_i+1, -6L) & M_8 = (2x_i+1, -5L) \\ M_9 = (\tau_i, -5L) & M_{10} = (\tau_i, -4L) \\ A = (4x_i+4, -4L) & M_{12} = (4x_i+4, -3L) \\ M_{13} = (\tau_i, -3L) & M_{14} = (\tau_i, -2L) \\ M_{15} = (7L, -2L) & M_{16} = (7L, -L) \\ M_{17} = (2x_i+2, -L) & M_{18} = (2x_i+2, 0) \end{array}$$

with the $N_k = (-\alpha x_j - \beta, \gamma)$ iff $M_k = (\alpha x_i + \beta, \gamma)$, for $1 \leq k \leq 18$. Therefore, the coordinates of the vertices can be defined by rational numbers represented by pairs of integers bounded by a quadratic polynomial function on the size of the graph.

We can prove that: the dual graph of the grid partition of the resulting polygon is a tree; M_{16}, M_{12}, M_8, M_7, M_5, and N_{16}, N_{12}, N_8, N_7 and N_5 require local guards; the boundary of Ξ_{ij} imposes no restriction on the propagation of the corresponding visibility cones $\mathcal{C}_A$ and $\mathcal{C}_B$; the unique point on the boundary of Ξ_{ij} that sees both A and N_{16} is M_1 (similar for B, M_{16} and N_1); the three local guards N_8, N_1 and M_8 jointly see all the gadget vertices other than A (similar for N_8, M_1 and M_8 and B). Lemma 2 states the final result we need to conclude the proof and can be shown as Lemma 2.2. of [4].

Lemma 2 *The thin orthogonal polygon P that is obtained can be guarded by $3|E| + k$ boundary guards if and only if the there is a vertex-cover of size k for the instance graph $G = (V, E)$.*

2 Hardness for vertex guards

Theorem 3 GVTP *is NP-hard for thin orthogonal polygons with vertex guards.*

For the proof, we can adapt the previous construction, following the idea of [4], as sketched in Fig. 4.

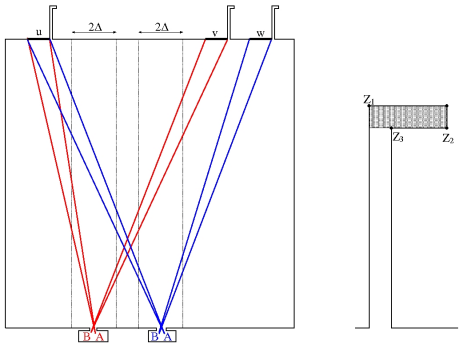


Figure 4: The reduction from VC to GVTP with vertex guards for $G = (\{u, v, w\}, \{(u, v), (u, w)\})$. Ear gadgets are attached to the right endpoints of the segments. Each ear gadget requires a local guard on a vertex of the shaded region (to guard Z_2).

We consider the polygon obtained previously and attach a tiny *ear gadget* to the right endpoint of each line segment $\overline{A_iB_i}$, for each $v_i \in V$. The local vertices of the ear gadget attached to B_j, w.r.t. the cartesian system fixed at the bottom left corner of the large square, can be defined as

$$\begin{aligned} Z_1 &= ((x'_j+1)\Delta,\, L(\Delta+1)+1) \\ Z_3 &= ((x'_j+1)\Delta+1,\, L(\Delta+1)) \\ Z_2 &= ((x'_j+1)\Delta+L,\, L(\Delta+1)) \end{aligned}$$

and $((x'_j+1)\Delta + L,\, L(\Delta+1)+1)$. The separation slabs guarantee that the dual graph of $\Pi_{HV}(P)$ for the new polygon P is still a tree, as required. The ear gadgets are defined in such a way that the vertex A of Ξ_{ij} cannot see any vertex of an ear-gadget except for B_i. Otherwise, A would see points on the boundary of P arbitrarily closed to B_i but to the right of B_i, which is impossible by the definition of the visibility cone $\mathcal{C}_A$. The height of the ear gadget prevents B from

seeing any local vertex of the ear-gadget attached to A_jB_j. For each $j \geq 2$, it is sufficient to guarantee that, for all Ξ_{ij}, the intersection point of the ray $\overrightarrow{BB_j}$ with the vertical edge incident to the vertex labeled Z_3 is below Z_3. This holds for all i if it holds for B in the rightmost d-gadget $\Xi_{i'j}$, since the rays $\overrightarrow{BB_j}$ are sorted by slope around B_j.

Each ear-gadget needs a local guard that must be located in one of the vertices of the shaded region and none of these vertices sees a local vertex of a d-gadget. This means that these guards cannot replace any guard located in a segment. Since any guard located on a segment can move to the segment right endpoint to become a vertex-guard, without loss of visibility, we can adapt the proof of Lemma 2 to show Lemma 4.

Lemma 4 *The thin orthogonal polygon P that is obtained can be guarded by $|V|+3|E|+k$ boundary guards if and only if the there is a vertex-cover of size k for the instance graph $G = (V, E)$.*

3 Hardness for point guards

Theorem 5 GVTP *is NP-hard for thin orthogonal polygons with point guards.*

For the proof, we construct a reduction from the minimum line cover problem (MLCP), as in [4]. MLCP is NP-hard [7]. Given a set $\mathcal{L} = \{l_1, \ldots, l_n\}$ of lines in the plane, MLCP is the problem of finding a set of points of minimum cardinality such that each line $l \in \mathcal{L}$ contains at least one point in that set.

Without loss of generality, we consider that $\mathcal{L}$ contains neither vertical nor horizontal lines. The polygon constructed for the reduction is obtained by attaching *single-gadgets* (called *s-gadgets*) to a bounding box $\mathcal{B}(\mathcal{L})$ that contains all intersection points of pairs of lines in $\mathcal{L}$ in its interior. The idea of this construction is sketched in Fig. 5.

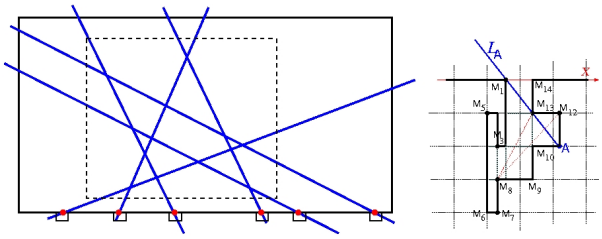


Figure 5: The reduction from MLCP to GVTP with guards anywhere. Each tiny box on the bottom represents an s-gadget (note that not all lines intersected the bottom edge of the dashed bounding box). On the right, an s-gadget in detail (the vertices are labelled from M_1 to M_{14}, in CCW order, $A \equiv M_{11}$).

In order to guarantee that a thin orthogonal polygon is obtained, we define a new type of s-gadget, sketched on Fig. 5, where M_1 and M_{13} reduce the visibility cone $\mathcal{C}_A$ to the line L_A. Moreover, we had to restrict the locations of s-gadgets to the bottom edge of $\mathcal{B}(\mathcal{L})$, in contrast to [4]. This can be done because, for a sufficiently large bounding box, all lines will intersect the bottom edge of $\mathcal{B}(\mathcal{L})$, as there are no horizontal lines in $\mathcal{L}$. At least a local guard is needed for each s-gadget. As for the d-gadgets, taking into account the relative positions of intersections of the lines with the bottom line (i.e., of vertices M_1), and their slopes, we can define the vertices of the tiny s-gadget in such a way that M_8 sees M_{12} and M_7, and all local vertices except for A. We can conclude that the vertices of P can be guarded by $n + k$ guards if and only if there is a cover for $\mathcal{L}$ of size k.

References

[1] T. Biedl, M. T. Irfan, J. Iwerks, J. Kim and J. S. Mitchell, Guarding polyominoes, in: *Proc. SoCG 2011*, ACM, 2011, 387–396.

[2] A. Bottino and A. Laurentini, A nearly optimal algorithm for covering the interior of an art gallery, *Pattern Recognition* **44** (2011), 1048–1056.

[3] M. C. Couto, P. J. de Rezende, and C. C. de Souza, An exact algorithm for minimizing vertex guards on art galleries, *Int. T. Oper. Res.* **18** (2011), 425 – 448.

[4] M. J. Katz and G. S. Roisman, On guarding the vertices of rectilinear domains, *Computational Geometry – Theory and Applications* **39** (2008), 219–228.

[5] D. T. Lee and A. K. Lin, Computational complexity of art gallery problems, *IEEE Transactions on Information Theory* **32** (1986), 276–282.

[6] A. M. Martins and A. Bajuelos, Vertex guards in a subclass of orthogonal polygons, *Int. J. Computer Science and Network Security*, **6** (2006), 102–108.

[7] N. Megiddo and A. Tamir, On the complexity of locating facilities in the plane, *Oper. Res. Lett.* **1** (1982), 194–197.

[8] J. O'Rourke, *Art Gallery Theorems and Algorithms*, Oxford University Press, 1987.

[9] D. Schuchardt and H. Hecker, Two NP-hard problems for ortho-polygons, *Math. Logiv. Quart.* **41** (1995), 261–267.

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

[11] A. P. Tomás, A. Bajuelos and F. Marques, On visibility problems in the plane – solving minimum vertex guard problems by sucessive approximations, in: *Proc. Int. Symp. Artificial Intelligence and Mathematics (ISAIM)*, Florida, 2006.

[12] A. P. Tomás. Guarding path orthogonal polygons, DCC&CMUP, Univ. Porto, 2013 (in prep).

Solving common influence region queries with the GPU

Marta Fort and J.Antoni Sellarès*

Departament Informàtica, Matemàtica Aplicada i Estadística. Universitat de Girona.

Abstract

In this paper we propose and solve *common influence region queries*. We present GPU parallel algorithms, designed under CUDA architecture, for approximately solving the studied queries. We also provide and discuss experimental results showing the efficiency of our approach.

Introduction

Common influence region queries are related to the capacity of two sets of facilities of different type, competitive and collaborative, of attracting customers. Solutions to common influence region queries help decision makers to develop competitive-collaborative opportunities.

Papers [2, 3, 1] deal with a related problem which tries to maximize the area of the Voronoi region of a new non-weighted facility. Fort and Sellarès [4] present an algorithm for optimizing k-nearest/farthest influence regions of a set of non-weighted facilities.

The advancement in GPUs (Graphics Processing Units) hardware design, together with CUDA (Compute Unified Device Architecture), make them attractive to solve problems which can be treated in parallel as an alternative to CPUs. General-Purpose computing on GPUs (GPGPU) is playing an increasing role in scientific computing applications, which range from numeric computing operations to data mining or geometric processing, where the potential of the GPU for delivering real performance gains on computationally complex, large problems is demonstrated.

By working towards practical solutions, since exact algorithms to solve the common influence region queries are hard to implement and quite slow in practice, we explore a GPU parallel approach, designed under CUDA architecture, for solving the queries approximately. We also provide and discuss experimental results obtained with the implementation of our algorithms that show the efficiency and scalability of our approach.

*Email:(mfort,sellares)@imae.udg.edu.

Authors research supported by TIN2010-20590-C02-02.

1 Preliminaries

1.1 The weighted area of a region

Let $\mathcal{R} = \{R_1, \cdots, R_m\}$ be a partition of the domain D. We associate with each region R_i a non-negative number w_i, called the weight of R_i.

We define the weighted area of the region $R \subset D$, denoted $w(R)$, as:

$$w(R) = \sum_{j=1,\cdots,m} w_j \ \mu(R \cap R_j)\,,$$

where $\mu(R \cap R_j)$ denotes the area of the subregion $R \cap R_j$.

1.2 CUDA architecture

CUDA is a parallel computing architecture that makes GPUs accessible for computation like CPUs. The CUDA processors, which can be executed in parallel, are referred as threads, and each thread executes the instructions contained in the so called kernels in parallel. Each thread computation is independent from the others, however, there exist some read-modify-write atomic operations called atomic functions. They read and return the value stored in a memory position, operate on it and store the result without allowing, during the whole process, any other access to that memory position. These operations allow the users to obtain global results when several threads access to the same memory position. For instance we can obtain a global sum, maximum or minimum in a specific position by using them.

Several types of memories can be used, data stored in global memory are accessible by every thread and are visible from the CPU, global memory is where more data can be allocated, but is the slowest access time memory. Shared memory and registers are the fastest memory, shared memory can be accessed by all the threads of the same block, and registers store the local variables of the threads. The number of accesses to memory are reflected in the execution times of the algorithms. Thus they are provided in the complexity analysis, ν read or written values to global memory are represented by r^g_ν and w^g_ν, and to shared memory they are denoted by r^s_ν and w^s_ν.

2 Influence regions

Let S be a set of n points included in a bounded domain D of the Euclidean plane. Each point $s \in S$ is associated with a positive real weight $w_s > 0$. The weighted distance $d_s(q)$ from an arbitrary point $q \in D$ to the point $s \in S$ is defined as $d_s(q) = (1/w_s)\, d(s,q)$, where $d(p,q)$ denotes the Euclidean distance between points p, q.

The k-influence region of a point $s \in S$, denoted $I_k(s,S)$, is the set of points of D having s among their k-nearest points in S. For any point $q \in D$, denote by $n_k(q,S)$ the weighted distance from q to its k-th nearest point in S, i.e. the weighted distance to the point of S that ranks number k in the ordering of the points by increasing weighted distance from q. We have:

$$I_k(s,S) = \{q \in D \,|\, d_s(q) \leq n_k(q,S)\}\,.$$

From the definition follows that the 1-influence region of a site s is its 1-Voronoi region, and that $I_k(s,S) \subset I_{k+1}(s,S)$.

A k-influence region is bounded by bisectors of points in S. In general, k-influence regions need not to be convex, nor simply connected, nor even connected. Two k-influence regions may share disconnected edges (see Figures 1 a) and b)).

2.1 Common influence regions

Let P and Q be finite disjoint sets of n and m weighted points, respectively, within the bounded domain D of the Euclidean plane.

The (k,k')-Common Influence Region of $p \in P$, $q \in Q$, denoted $C_{k,k'}(p,q,P,Q)$, is the set of points of D having p among their k-nearest points in P and at the same time having q among their k'-nearest points in Q:

$$C_{k,k'}(p,q,P,Q) = I_k(p,P) \cap I_{k'}(q,Q)\,.$$

In Figure 1 c) we can see an example of a $(5,2)$-common influence region painted in green.

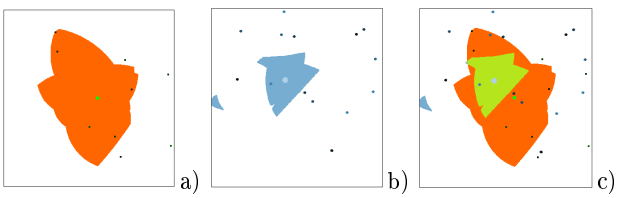


Figure 1: a) 5-influence region, b) 2-influence region, c) $(5,2)$-common influence region.

2.2 Common influence region queries

Let P and Q be finite disjoint sets of n and m weighted points, respectively, within the bounded domain D of the Euclidean plane, and $\mathcal{R}$ be a weighted partition of D. We study the following queries:

Feasible pairs query. Given fixed non-empty sets $P' \subseteq P$ and $Q' \subseteq Q$ and a real positive number W_0, find all pairs (p,q), $p \in P'$ and $q \in Q'$, such that the weighted area $w(C_{k,k'}(p,q,P,Q)) \geq W_0$.

Feasible partners query. Given fixed non-empty sets $P' \subseteq P$ and $Q' \subseteq Q$ and a real positive number W_0, find all points $p \in P'$, such that for each $q \in Q'$ the weighted area $w(C_{k,k'}(p,q,P,Q)) \geq W_0$.

The parameters k, k', W_0, I_0 should be chosen by an expert, according to the localization of p with respect the points of Q' and the cardinality of Q', during an iterative what-if analysis process.

To obtain $w(C_{k,k'}(p,q,P,Q)) = w\big(I_k(p,P) \cap I_{k'}(q,Q)\big)$, we can first compute the overlay intersection between $I_k(p,P)$, $I_{k'}(q,Q)$ and the regions of the weighted partition $\mathcal{R}$. Since each maximal connected region of the resulting intersection is contained in a region of $\mathcal{R}$, it has univocally associated a weight. Consequently, the weighted area $w(C_{k,k'}(p,q,P,Q))$ can be written as a sum of functions that depends on the location of p and q. Since computing the maximal connected regions together with their weighted area is difficult, it is also difficult computing the weighted area of a common influence region and solving common influence region queries. This has motivated us to explore an alternative GPU parallel approach, designed under CUDA architecture, for approximately solving common influence region queries.

3 Solving common influence region queries using CUDA

The weighted areas of the common influence regions are estimated by using a discretization of the domain D. An axis-parallel rectangular grid of size $H \times W$ is superimposed on the domain D defining a set S of r points corresponding to the geometric center of the grid cells which are weighted according to the domain partition $\mathcal{P}$. As it is obvious, the bigger the size of the used grid the more accurate the obtained common influence region. The weighted area is estimated by summing up the weights of all the points of S contained in the common influence region. Notice that, depending on the domain weighted partition, small changes in the common influence region can produce big changes in its weighted area.

Solving the defined queries requires an initial step consisting in computing the k-nearest neighbor distance from each point of S to the points in P and the k' nearest neighbor to the points in Q. Notice that even though the queries subsets P' and Q' are used,

the k and k'-neighbor distances are always referred to P and Q.

3.1 k-nearest neighbor distance computation

We start by computing the weighted distance of each point $s \in S$ to its k-th nearest neighbor in P, considering all the points in parallel. With this aim, we have extended the CudaKNN algorithm described by Liang et al. [5] to handle the weighted case. The algorithm computes for each $p \in P$ the weighted distances to all the points in S using shared memory and making the threads in a block cooperate to determine, after exploring all the point in P, the k-nearest neighbors of s. We transfer the points of S and P from the CPU to the GPU, then we compute the k-weighted nearest distance of each point in S to P and store all them in global memory in a 1D array n_{P_k} of size r.

In the same way, we also transfer the points of S and Q from the CPU to the GPU, compute the k'-weighted nearest distance of each point in S to Q and store them in a 1D array $n_{Q_{k'}}$ of size r.

3.2 Influence region queries resolution

Given $P' \subseteq P$ and $Q' \subseteq Q$ of size n' and m' respectively, and assuming that the k and k'-nearest neighbor distances from each point of S to P and Q have been computed, we explain how the proposed queries can be solved.

Our approach for solving the feasible pairs and the possible partners queries follow a very similar procedure.

Feasible pairs query

The solution of the feasible pair query is reported giving the number of feasible pairs and the list containing them. In order to solve it we consider $n'm'$ threads, an integer to store the number of feasible pairs and an 1D-array of size $n'm'$ to store the feasible pairs. Each thread estimates the weighted area $w_{p,q} = w(C_{k,k'}(p,q,P,Q))$ of one pair (p,q). In the case that $w_{p,q}$ gets the minimum required value W_0, the number of feasible pairs is incremented by one, and the pair is stored in the first empty position of the feasible pairs array.

The weighted area $w_{p,q}$ is estimated by considering all the points $s \in S$, computing the distances $d_p(s)$ and $d_q(s)$ and comparing them with $n_k(s,P)$ and $n_{k'}(s,Q)$ which are the ones stored in n_{P_k} and $n_{Q_{k'}}$. Thus $s \in C_{k,k'}(p,q,P,Q)$ whenever $d_p(s) \leq n_{P_k}(s)$ and $d_q(s) \leq n_{Q_{k'}}(s)$. In such a scenario, where all threads check all the points in S, shared memory should be used. The B threads in a block cooperate loading the space points and their neighbor distances from global to shared memory. This requires using three shared memory arrays of size B per block: S^s which stores B weighted space points, and $n^s_{P_k}$, and $n^s_{Q_{k'}}$ storing their corresponding k and k'-neighbor distances to P and Q, respectively. The i-th thread of the block reads from global memory the point $s_i \in S$ and its corresponding k and k' nearest neighbor distances, and stores them in the i-th position of S^s and $n^s_{P_k}$ and $n^s_{Q_{k'}}$, respectively. When all the threads in the block have finished loading this information, the first B points of S can be analyzed. Then each thread determines which of these points are contained in its corresponding common region $C_{k,k'}(p,q,P,Q)$ and accordingly accumulates their weights in a register. Once all the threads of the block have checked all the already loaded points, the next B points of S and their distances are loaded to shared memory and analyzed. We keep on proceeding similarly until all the points in S have been handled, and consequently w_{pq} has been estimated.

In order to obtain correct results it is important that the threads in a block wait each other in some critical points. In fact, two synchronization points are needed, one after transferring the corresponding weighted point of S with its two associated distances and the other when all the already stored points have been checked, just before starting transferring new information to shared memory. Otherwise there is no guarantee that all the threads in the block have finished with their tasks. The implementation has to be carefully done when r, the number of points in S, is not a multiple of B, the number of threads in a block, to avoid accesses to non-existent memory positions, but guaranteing that all the points of S are loaded to shared memory.

Feasible partners query

To solve this query we use an array f_p, of size n', which is used as a boolean array. At the end of the process, positions containing a 1 correspond to the points of P' having all the points of Q' as feasible partners.

After initializing the array f_p with ones, we proceed as before estimating the weighted common influence areas of all the $m'n'$ pairs in $P' \times Q'$. When a thread finishes estimating its corresponding weight w_{pq}, it checks wether $w_{pq} < W_0$, and in such a case, it sets $f_p[p]$ to 0. Thus, only for those points p achieving the minimum influence value for all the points $q \in Q'$ we will have $f_p[p] = 1$ at the end of the process.

The number, n'_f, and the list, f, of feasible partners can be obtained by performing a prefix sum on the array f_p, which is stored in p_s. After allocating an array of size n'_f, n' threads are considered, each thread checks wether its point of P' is a feasible partner of Q', by looking at its corresponding position of f_p. If it is so, the point of P' is stored in f in the position given by p_s.

3.3 Complexity analysis

Computing simultaneously for the r points of S the k-neighbor distances with respect to the set of n points by using the CudaKNN algorithm [5] requires $O(rnB)$ total work, where B is the number of threads per block. The computation is done in three different steps, that have $O(r)$, $O(B)$ and $O(k + n/B)$ operations per thread, respectively. Thus, the initial step in our case requires $O(r(n + m)B)$ total work.

Solving the feasible pairs query requires $n'm'$ threads performing $O(r)$ operations each, leading to $O(rn'm')$ total work. Concerning memory accesses, in the worst case $O(r^g_{n'm'r/B}+w^s_{n'm'r/B}+r^s_{n'm'r}+w^g_{n'm'})$ accesses are required. Finally, the global memory needed is $3r + n' + m' + n'm' + 1$, while $3B$ floats per block are required in shared memory.

The differences between solving the feasible pairs query and the feasible partners one, before finding the list of feasible partners, do not change the work per thread neither the global work nor the memory accesses. However, the global memory requirements are reduced to $3r + 2n' + m'$. Finding the list f only increases the global memory requirements in $n' + n'_f$.

4 Experimental Results

In this section, we present several experimental results obtained with the implementation of our algorithms. In all the experiments the points of P and Q are randomly distributed in the domain and their weights are randomly obtained integers between 1 and 15. They are based on the weighted domain partition presented in Figure 2. The points of the domain are painted in a purple color gradation, the darker the color the smaller the density value.

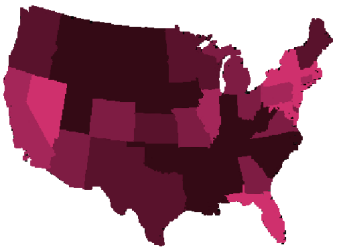
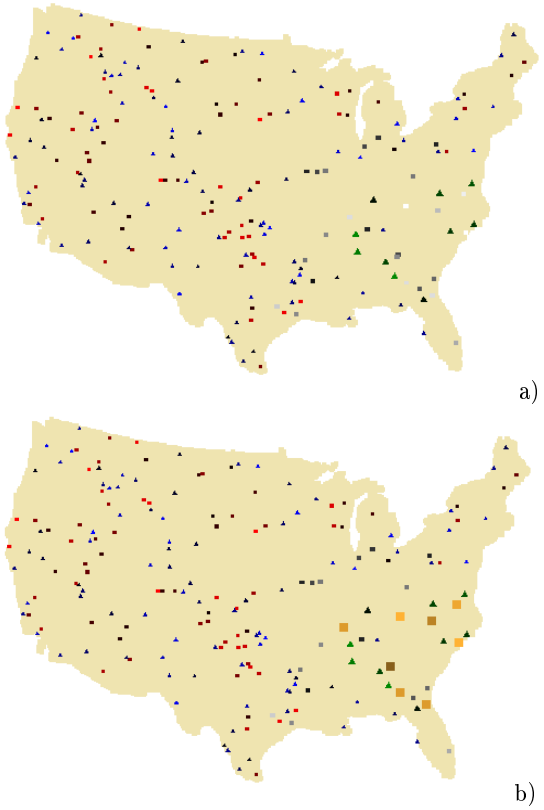


Figure 2: Domain weighted partition according to the population density.

In Figure 3, we provide the solution of a feasible partners query. Figure 3 a) shows the sets P, Q, P' and Q', that have 100, 100, 25 and 10 points, respectively. Points of sets P' and Q' are represented by grey squares and green triangles, and the points of P and Q not in P' and Q', by red squares and blue triangles, respectively. The orange squares of Figure 3 b) are the points of P' conforming the solution of the feasible partners query defined by P, Q, P', Q' for $k = k' = 15$, when the minimum required influence value is 20.



a)

b)

Figure 3: a) Points of the sets P, Q, P', Q'. b) Solutions of the feasible partners query.

For a grid of size 400×400, $n = 500$, $n' = 50$, $m = 200$, $m' = 25$, $k = 20$ and $k' = 20$, the running times needed to solve a feasible pairs query and a feasible partners query are 12 and 16 milliseconds. Using the CPU sequential version of the algorithms the times become 347 and 351 miliseconds, respectively. In the CPU version, the feasible partners query can be solved in 62 miliseconds if point $p \in P'$ is set as a not feasible partner at the first $q \in Q'$ for which $w(C_{k,k'}(p, q, P, Q)) < W_0$. This can not be done in the GPU, because threads cooperate transferring information to shared memory.

References

[1] O. Cheong, A. Efrat, and S. Har-Peled, *Finding a guard that sees most and a shop that sells most*, Discrete Comput. Geom. **37(4)** (2007), 545–563.

[2] M. Denny, *Solving geometric optimization problems using graphics hardware*, Comput. Graph. Forum **22** (2003), no. 3, 441–452.

[3] F. K. H. A. Dehne, R. Klein, and R. Seidel, *Maximizing a Voronoi region: the convex case*, Int. J. Comput. Geometry Appl. 15(5) (2005), 463–476.

[4] M. Fort and J.A. Sellarès, *GPU-Based Influence Regions Optimization*, ICCSA (2012), 253–266.

[5] S. Liang, Y. Liu, C. Wang, L. Jian, *A CUDA-based parallel implementation of k-nearest neighbor algorithm*, IEEE'09 (2009), 291–296.

Reporting flock patterns on the GPU

Marta Fort, J.Antoni Sellarès, and Nacho Valladares*

Departament Informàtica, Matemàtica Aplicada i Estadística. Universitat de Girona.

Abstract

In this paper we study the problem of finding flock patterns in a set of trajectories of moving entities. A flock refers to a large enough subset of entities that move close to each other for a given time interval. We present a parallel approach, to be run on a Graphics Processing Unit, for reporting maximal flocks. We also provide experimental results that show the efficiency and scalability of our approach.

1 Introduction

A trajectory is a sequence of sampled time-stamped point locations describing the path of a moving entity over a period of time. Assuming that the movement of an entity between two consecutive positions is done at constant speed and without changing direction, its trajectory is modelled by the polygonal line, that can self-intersect, whose vertices are the trajectory points.

In this paper we study the flock pattern [1, 2, 3], that identifies a group of entities moving close together during a given time interval (see Figure 1).

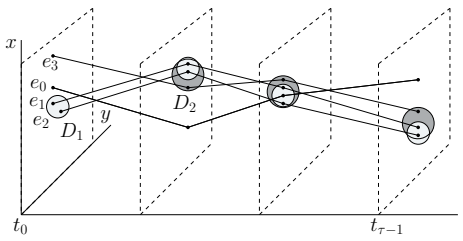


Figure 1: Example of two flocks.

The increasing programmability and high computational rates of Graphics Processing Units (GPUs), together with CUDA (Compute Unified Device Architecture) and some programming languages which use this architecture, make them attractive to solve problems which can be treated in parallel as an alternative to CPUs. GPUs are used in different computational tasks where a big amount of data or operations have to be done, whenever they can be processed or done in parallel. Some recent works show that demanding algorithms in different fields can take advantage of the GPU parallel processing [4, 5, 6].

*Email:(mfort,sellares,ivalladares)@imae.udg.edu.

Authors research supported by TIN2010-20590-C02-02.

In this paper, we present an efficient parallel GPU-based algorithm, designed under CUDA architecture, for reporting maximal flocks. The experimental results obtained with the implementation of our algorithm show the significance of the presented approach according to its performance and scalability.

2 The flock pattern

Let $E = \{e_0, \ldots, e_{n-1}\}$ be a set of n moving entities. The trajectory T_i of the entity e_i is a sequence of τ points in the plane $T_i : e_0^i, \ldots, e_{\tau-1}^i$, where e_j^i denotes the position of e_i at time t_j with $0 \leq j < \tau$. We assume that the positions are sampled synchronously for all the entities, and that entity e_i moves between two consecutive positions e_j^i, e_{j+1}^i with constant speed and without changing direction. Consequently, the trajectory T_i is described by the polygonal line, which may self-intersect, whose vertices are the trajectory points.

Flocks identify groups of entities whose trajectories are close together during a minimum period of time. Formally, according to [2], given a set E of entities, a minimum number of entities $\mu \in \mathbb{N}$, a number of time-steps $\delta \in \mathbb{N}$, a time interval $I_j^\delta = [t_j, t_{j+\delta-1}]$, and a distance $\epsilon \in \mathbb{R}$:

Definition 1 A flock $f(\delta, \mu, \epsilon)$ in a time interval I_j^δ, consists of at least μ entities such that for every discrete time step $t_\ell \in I_j^\delta$, there is a disk of radius ϵ that contains all the μ entities.

The number of reported or determined flocks is a critical issue that can adversely affect the response time and the proper interpretation of the final results. For a time-step $t_\ell \in I_j^\delta$ there may be several circles with radius ϵ that yield a flock with the same subset of entities of E, so we consider that two flocks are different if they involve two different subsets. Since there can be $\Theta(n^2)$ combinatorially distinct ways to place a circle of radius ϵ among n points in the plane, at each time-step there can be $\Theta(n^2)$ flock candidates involving different subsets of entities [9]. An algorithm having as output the position of the entities involved in a flock, would have an output size of $\Theta(n^3)$ per time-step. Moreover, it is easy to observe that a set of entities could be present in many flocks, and even one single entity can be involved in several flocks. For example, a flock of $\mu+1$ entities contains $\mu+1$ flocks of μ entities.

To overcome this problem, as in [3], instead of finding all the existent flocks, we are interested only in maximal flocks, so that the entities of one flock may not be a subset of the entities of another flock. Even in the case of finding maximal flocks, at every time-step there may still asymptotically be $\Theta(n^2)$ flock candidates, although in many real situations the number of candidates decreases considerably.

Definition 2 A maximal flock $mf(\delta, \mu, \epsilon)$ in the time interval I_j^δ, is a maximal flock $f(\delta, \mu, \epsilon)$ in I_j^δ with respect to the subset inclusion.

We will denote $\mathcal{M}_j^\delta$ the family of maximal flocks in the time interval I_j^δ, and $\mathcal{M}^\delta$ the family of all maximal flocks, that is, the flocks of $\mathcal{M}_j^\delta$, for $j = 0, \ldots, \tau - 1$.

2.1 Characterization

Vieira et al. [3] prove that, for each pair of entity locations, there exists two such disks. Thus, $2n^2$ disks per time step must be tested. Since we are interested in those flocks that are maximal, we prove that keeping just one of the two disks is enough. Consequently, only n^2 disks per time step must be tested, thus the number of tests is reduced by half.

The following lemma, similar to one presented in [3], allows us to bound the number of disks to be tested in each time step.

Lemma 3 *Let P be a set of n points and D' be a disk of radius r that covers a subset $P' \subseteq P$, $|P'| = k \leq n$. There exist another disk D'' of radius r such that: a) has at least two points $p_i, p_j \in P'$ on its boundary; b) covers a superset P'' of P', $P' \subseteq P'' \subseteq P$; c) it is located at the right side of points p_i, p_j.*

3 Reporting flock patterns

In this paper study the problem of reporting the family $\mathcal{M}^\delta$ of all maximal flocks. To report $\mathcal{M}^\delta$, we must find the family $\mathcal{M}_j^\delta$ in the time interval I_j^δ, for $j = 0, \ldots, \tau - 1$.

3.1 Computing the family $\mathcal{M}_j^\delta$

Next, we describe the two main steps of the algorithm used to compute the family $\mathcal{M}_j^\delta$ in the time interval I_j^δ.

First step. At each time step t_i in the time interval I_j^δ, we compute the family $\mathcal{F}_i$ of potential flocks containing at least μ entities. Each family $\mathcal{F}_i$ is obtained by using the characterization given in Lemma 3. Next, we find the maximal sets of each family $\mathcal{F}_i$. Abusing of notation, we still denote $\mathcal{F}_i$ the obtained subfamily of potential maximal flocks. At the end of the step, we have the array of families of potential maximal flocks $\mathcal{F}_j^\delta = [\mathcal{F}_j, \ldots, \mathcal{F}_{j+\delta-1}]$ in the time interval I_j^δ.

Second step. We first compute the family $\mathcal{I}_j^\delta$ obtained intersecting the δ families of potential maximal flocks in $\mathcal{F}_j^\delta$. That is, $\mathcal{I}_j^\delta$ contains the sets which are the intersection of δ sets, one of each family $\mathcal{F}_i$ in $\mathcal{F}_j^\delta$. Finally, we find the subfamily of maximal sets of $\mathcal{I}_j^\delta$ that contain at least μ elements. The obtained subfamily is the desired $\mathcal{M}_j^\delta$.

4 GPU implementation

In this Section, we describe how to implement the different steps of the algorithm that computes $\mathcal{M}_j^\delta$, when GPU parallel techniques are used.

4.1 The multi-grid structure $\mathcal{G}$

We denote by $E_i = \{e_i^0, \ldots, e_i^{n-1}\}$ the set of locations of the entities of E at time step t_i, $i \in \{0, \ldots, \tau - 1\}$. In [3], a single grid is used to perform quick disk range searching queries over each set E_i. Instead of using a grid for each time step, we build a multi-grid structure, denoted $\mathcal{G}$, in parallel in the GPU, that allows us to search simultaneously in each one of the δ sets of $E_j^\delta = [E_j, \ldots, E_{j+\delta-1}]$.

The multi-grid structure contains δ regular grids, where the edges of the grid cells have length ϵ. Each grid j contains the set of points corresponding to E_j. In order to maximize the parallel performance, the top-right corner is defined so that the grid is a square, that is, it has the same number of rows and columns (see Figure 2).

The multi-grid structure $\mathcal{G}$ is composed of 7 arrays allocated in the GPU global memory. In Figure 2 we can see an example for $\delta = 3$. The bottom-left and the top-right corners of each grid are stored in the array b_δ of size δ, where $b_\delta[j]$ stores the two corner points of the grid containing E_j. The number of cells per row (or per column) of each grid is stored in $\widetilde{c_\delta}$ and its prefix sum is stored in $\widetilde{p_\delta}$. Both arrays are integer arrays of size δ.

To represent the δ grids, we use the array of integers c_δ, whose size depends on the size of the different grids which directly depend on the distribution of the points. For a given grid cell, we store in the corresponding c_δ position the number of points contained on that cell. The array p_δ of the same size of c_δ is obtained as the prefix sum of c_δ. Note that, for a better understanding, in Figure 2 the arrays c_δ and p_δ are showed as several $2D$ grids, but in the GPU memory they are stored one $1D$ array each, by storing one grid after the other starting from left to right and from top to bottom, thus, the first grid corresponds to the time step t_i and the last one of the time step $t_{i+\delta}$. With this structure we can easily access, in a parallel way, to any element of the δ grids, by using the fact that the set of points of a cell w starts at position $p_\delta[w]$ of e_δ, and is stored in $c_\delta[w]$ consecutive positions of e_δ. In each e_δ position we store the two floats of the

corresponding location. Finally, $\widetilde{e_\delta}$ is an integer array of size $n\delta$ containing at position j the entity id of the $e_\delta[j]$ location.

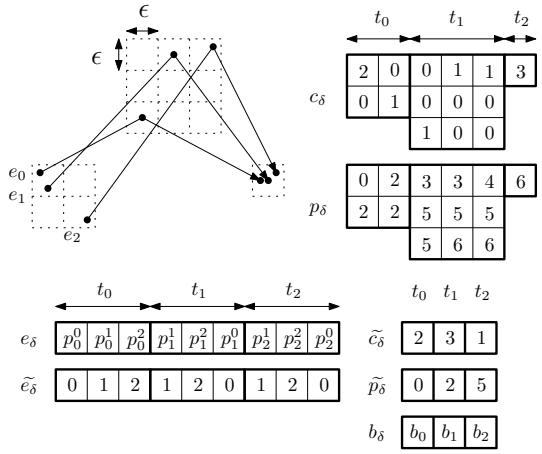


Figure 2: Example of multi-grid $\mathcal{G}$ for $\delta = 3$.

We construct the $\mathcal{G}$ structure in parallel as follows. We allocate CPU arrays $\widetilde{c_\delta}'$, $\widetilde{p_\delta}'$, b'_δ of size δ, and p' and $\widetilde{e_\delta}'$ of size $n\delta$. While we load the input P into p' the arrays $\widetilde{c_\delta}'$, $\widetilde{p_\delta}'$ and b'_δ are accordingly filled. Then $\widetilde{c_\delta}$, $\widetilde{p_\delta}$, b_δ and p are allocated in GPU memory and the $\widetilde{c_\delta}'$, $\widetilde{p_\delta}'$, b'_δ and p' values are transferred to GPU memory to $\widetilde{c_\delta}$, $\widetilde{p_\delta}$, b_δ and p respectively.

Then, c_δ and p_δ are allocated in the GPU memory according to $\widetilde{c_\delta}$ and $\widetilde{p_\delta}$ values. Additionally, the arrays e_δ and $\widetilde{e_\delta}$ are allocated with size $n\delta$ also in GPU memory. Because GPU has no dynamic memory we have to construct $\mathcal{G}$ in two steps.

First, we fill c_δ by counting the number of points contained in each cell. This is done in parallel by launching a kernel with $n\delta$ threads, that is, a thread per point and per time step. Each thread idx reads its $p[idx]$ value and determines its position in the grid by using its position on the plane, the grid corresponding to the time step which the point belongs to, and the $\widetilde{c_\delta}$ and $\widetilde{p_\delta}$ arrays. The corresponding c_δ position is incremented in 1. Because many threads may correspond to the same c_δ position we use atomic operations to ensure no thread interferences.

Once c_δ is computed, p_δ is obtained as the prefix sum of c_δ and we allocate an auxiliary array v with the same size of c_δ, initialized to zeros. Now, we launch a parallel kernel with $n\delta$ threads where each thread idx reads its $p[idx]$ value, determines its position w in the grid and stores $p[idx]$ at $e_\delta[p_\delta[w] + v[w]]$. Every time a thread stores its associated point into e_δ the corresponding v value is incremented in one by using an atomic operation.

4.2 Finding potential flocks

The computation of the array of potential flocks $\mathcal{F}_j^\delta$ of a given interval I_j^δ is done in two main steps. First, we need to find the center of the disks which are obtained applying the characterization given in Lemma 3, and second, we need to report the entities contained in those disks. Additionally, because GPUs do not have dynamic memory, we split the process in several steps.

First, we allocate an array of integers D_c of size $n\delta$ to count how many disks there exist. This is done in parallel by launching a kernel with $n\delta$ threads, that is, a thread per point within I_j^δ. Using the structure $\mathcal{G}$ each thread idx goes through its neighbors and counts the number of points which are at distance $\leq 2\epsilon$. The results are stored in $D_c[idx]$. Because this is computed in parallel, each disk will be reported twice, once for each of the two points defining the disk. In order to avoid this, we force threads to check those points whose entity id is bigger than itself. Then, D_p of size $n\delta$, is computed as the prefix sum of D_c, and the array D is allocated with size $D_p[n\delta-1]+D_c[n\delta-1]$, where we will store the centers of disks. The same process is repeated, but, instead of counting, each thread reports the center of the disk. Let us recall that only the right disk per paired point has to be reported.

In the second step, we actually report the potential flocks. To do this we use $\mathcal{G}$ to perform the range searching queries, using the values of the array D as the center of the queries with radius ϵ. Note that, because in order that the regular grid structure works for range searching queries, we must be able to locate any point within the grid. It may happen that some disk centers were placed outside the grid for a given time step. Thus, before we perform the range searching queries we have to reconstruct $\mathcal{G}$ so that the center of the disks were contained in $\mathcal{G}$. We could modify $\mathcal{G}$, but it is faster to rebuild it than to modify it.

To count the number of points contained on each disk we first allocate P_c, the array of integers of size equal to the total number of disks reported which is $D_p[n\delta-1]+D_c[n\delta-1]$, initialize it to zeros and launch a kernel with one thread per disk. Each thread idx reads the center of its disk $D[idx]$, locates $D[idx]$ on the grid and checks the distance between the center of the disk and the points of its cell neighbors. Each thread counts how many points are at distance $\leq \epsilon$ to its disk center, and if there are at least μ the number is stored in $P_c[idx]$. Finally, we compute P_p as the prefix sum of P_c.

If there is at least one disk per time step with at least μ entities, we continue to report the potential flocks. The last position of P_p plus the last position of P_c gives us ω, the size of the array used to store the potential flocks. Thus, the array P is allocated as an integer array with size ω and is filled in parallel using a thread per disk. The process is the same as before but this time, instead of counting, each thread stores in P the entities contained inside each disk.

Note that, P_c, P_p and P denote a structure contain-

ing δ families of sets. This is the array of potential flocks $\mathcal{F}_j^\delta$, that is, δ families of sets containing the potential flocks of δ consecutive time steps starting at time step t_j. We denote $\mathcal{F}_j$ the family of potential flocks of the time step t_j.

The last step is to reduce each $\mathcal{F}_i \in \mathcal{F}_j^\delta$ so that it only contains the sets which are maximal, with respect to the partial order induced by the subset relation in each family. To this end, we use the parallel algorithm of Fort et al. [7]. When the process finishes, we have in $\mathcal{F}_j^\delta$ the maximal potential flocks with μ or more entities at each time step of the interval I_j^δ.

4.3 Reporting $\mathcal{M}^\delta$ flocks

We want to report all the flocks M_j^δ for all $I_j^\delta, j = [0, \ldots, \tau - \delta]$. Thus, we could perform the same algorithm over each I_j^δ, but many information computed for a given time step within a time interval can be reused for the following time intervals. The idea is to compute the potential flocks for a given interval and compute their intersections in a way that we can reuse the information for the next time interval. To this aim we proceed as follows.

We start at the interval I_0^δ and we compute $\mathcal{F}_0^\delta$. Then, in order to obtain $\mathcal{M}_0^\delta = \mathcal{F}_{\delta-1} \cap \mathcal{F}_{\delta-2} \cap \cdots \cap \mathcal{F}_0$ we start at $\mathcal{F}_{\delta-1}$ and we intersect $F_{\delta-2} \cap \mathcal{F}_{\delta-1}$. Next, we intersect $\mathcal{F}_{\delta-3} \cap \mathcal{F}_{\delta-2} \cap \mathcal{F}_{\delta-1}$ and so on. After each intersection, we compute the maximal sets for each resulting intersection discarding those sets which are non maximal or containing less than μ entities. When we are done, we add, to $\mathcal{M}^\delta$, the family $\mathcal{M}_0^\delta$, which contains the flocks of I_0^δ.

For the following time steps, instead of recomputing all the potential flocks and all the intersections, we reuse the previous computations. If for a given time step t_j, a family of potential flocks $\mathcal{F}_k, j \leq k < j+\delta$ is not inside the array $\mathcal{F}_j^\delta$, we compute $\mathcal{F}_{j+\delta-1}^\delta$. That is, instead of computing the potential flocks of one single time step we compute them in δ intervals. Then, we perform the corresponding intersections. This technique does not decrease the number of intersection we have to do, but, in practice, the δ families we intersect in each step are smaller than the ones containing the potential flocks, leading to faster results. We repeat the process until $j = \tau - \delta$.

The intersection process is also performed in parallel using the GPU algorithm presented by Fort et al in. [8].

5 Experimental results

The experimental results are obtained using an Intel Core2 CPU 6400 with a Nvidia GTX 480. We split the running time into 2 main steps, the potential flocks reporting process and the intersection process. The accumulated value, corresponding to the columns height, gives the total running time of the algorithm.

The algorithm has been tested with a data set extracted from [10], that consists of 145 trajectories of 2 school buses around Athens metropolitan area in Greece with a total of 66,096 time steps.

The results (Figure 3) shows that when we vary ϵ while we maintain $\mu = 5$ and $\delta = 10$, the number of flocks reported increases as long as we increase ϵ. The most affected part is the intersection process. This is because the more flocks we report the more intersections we have to compute. When the parameter μ is varied, while $\epsilon = 1200$ and $\delta = 10$, the condition to form flock is more restrictive. Thus, the number of flocks decreases and, consequently, the running times too.

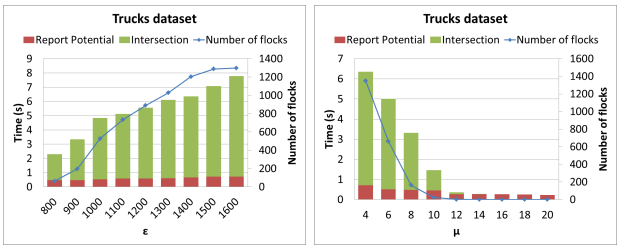


Figure 3: Running times varying ϵ and μ.

References

[1] J. Gudmundsson, M. J. van Kreveld, B. Speckmann, Efficient Detection of Patterns in 2D Trajectories of Moving Points, GeoInformatica 11 (2) (2007) 195–215.

[2] M. Benkert, J. Gudmundsson, F. Hübner, T. Wolle, Reporting flock patterns, Computational Geometry 41 (3) (2008) 111–125.

[3] M. R. Vieira, P. Bakalov, V. J. Tsotras, On-line discovery of flock patterns in spatio-temporal data, in: D. Agrawal, W. G. Aref, C.-T. Lu, M. F. Mokbel, P. Scheuermann, C. Shahabi, O. Wolfson (Eds.), GIS, ACM, 2009, pp. 286–295.

[4] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. E. Lefohn, T. J. Purcell, A survey of general-purpose computation on graphics hardware, Computer Graphics Forum 26 (1) (2007) 80–113.

[5] N. Coll, M. Fort, N. Madern, J. A. Sellarès, Multi-visibility maps of triangulated terrains, International Journal of Geographical Information Science 21 (10) (2007) 1115–1134.

[6] M. Fort, J. A. Sellarès, N. Valladres, Computing popular places using graphics processors, in: Proc. SSTDM'10 in cooperation with IEEE ICDM'10, IEEE Computer Society, 2010, pp. 233–241.

[7] M. Fort, J.A. Sellaès, N. Valladares, Finding extremal sets on the GPU (Submitted).

[8] M. Fort, J.A. Sellaès, N. Valladares, Intersecting two families of sets on the GPU (Submitted).

[9] J. Gudmundsson, M. van Kreveld, B. Speckmann, Efficient Detection of Motion Patterns in Spatio-Temporal Data Sets, in: D. Pfoser, I. F. Cruz, M. Ronthaler (Eds.), GIS, ACM, 2004, pp. 250–257.

[10] Y. Theodoridis, R-Tree portal, `http://www.rtreeportal.org` (2011).

Parallel constrained Delaunay triangulation

Narcís Coll and Marité Guerrieri*

Geometry and Graphics Group. Universitat de Girona

Abstract

In this paper we propose a new GPU method able to compute the 2D constrained Delaunay triangulation of a planar straight line graph consisting of points and segments. The method is based on an incremental insertion, taking special care to avoid conflicts during concurrent insertion of points into the triangulation and concurrent edge flips.

Introduction

The constrained Delaunay triangulation (CDT) is one of the fundamental topics in Computational Geometry and it is used in many areas such as terrain modelling, finite element method, pattern recognition, path planning, etc. A CDT of a planar straight line graph (PSLG) can be constructed in the following way: begin with an arbitrary triangulation of the PSLG points; examine each PSLG segment in turn to see if it is an edge; force each missing PSLG segment to be an edge; then flip non-locally Delaunay edges until all edges are locally Delaunay with the provision that PSLG segments cannot be flipped. There are two ways to force a PSLG segment to be an edge of the triangulation. The first consists of deleting all the edges it crosses, then inserting the segment and retriangulating the two resulting polygons (one from each side of the segment). The second approach consists of iteratively flipping the edges that crosses the segment until the segment is an edge. Qi et al. [2] presented a GPU method for computing the CDT of a PSLG. This method has three phases. The first phase computes the Delaunay triangulation of the PSLG points, while the second inserts the PSLG segments into the triangulation using edge flipping and the third flips the remaining non-locally Delaunay edges. Experimental results show that the method achieves a significant speedup with respect to the best CPU methods. However, since the Delaunay triangulation is constructed using a digital Voronoi diagram computed on a uniform grid, the method needs a huge memory space to allocate the grid and is less efficient when the distribution of the input points is far from being uniform.

*Email: (coll,mariteg)@imae.udg.edu. Authors research supported by TIN2010-20590-C02-02.

Our approach of computing the CDT on the GPU extends the method presented in [1] by simultaneously inserting points and segments into the triangulation.

1 Our approach

Our algorithm is based on an iterative process that finishes when all PSLG elements (points and segments) are inserted into the Delaunay triangulation and no edge needs to be flipped. In each iteration as many elements as possible are inserted with the condition that only one point can be inserted into one single triangle. Each iteration is divided into four steps: location, where the triangle containing every non inserted point is determined; insertion, where at most one point per each triangle is inserted; marking, where some edges of the triangulation are marked to be a segment or marked to be flipped because they are crossed by a segment or they are non-locally Delaunay; and flipping, where some of the marked edges are flipped thus avoiding conflicts between them.

Let n be the number of points and m be the number of segments. In order to use the GPU's resources as efficiently as possible the following arrays allocated in the global memory are used:

`Points`. Positions (x, y) in 2D. Its first three positions corresponds to the three vertices of a large auxiliary triangle that contains all points. Size $n + 3$.

`Segments`. Indices to `Points`. Each two consecutive indices correspond to a segment. Size $2m$.

`Inserted-P`. Binary flags to determine whether a point has been inserted or not. Size $n + 3$.

`Inserted-S`. Binary flags to establish whether a segment has been inserted or not. Size m.

`Triangles`. Indices to `Points`. Each three consecutive indices correspond to a triangle. Position zero of this array corresponds to the auxiliary triangle. Size $3(2n + 1)$.

`Neighbours`. Indices to `Triangles`. Each three consecutive indices correspond to the current neighbours of the triangle. Size $3(2n + 1)$.

`FutureNeighbours`. Indices to `Triangles`. Each three consecutive indices correspond to the neighbours that the triangle will have after executing a point insertion or a flip which affects its current neighbours. Size $3(2n + 1)$.

`ContainingTriangle`. Indices to `Triangles` to record which triangle contains a point, in the case that the point has not been inserted, or otherwise, a triangle incident to the point. Initially, all points are contained in the auxiliary triangle. Size $n+3$.

`PointToInsert`. Indices to `Points` to record which the next point to be inserted in each triangle is. Size $2n+1$.

`Flip`. Flags (0, 1, 2, 3) to discern whether an edge has to be flipped or not. Each three consecutive flags corresponds to a triangle. Flag 0 indicates the edge is not a segment and does not have to be flipped. Flag 1 indicates the edge is not a segment and has to be flipped because it is non-locally Delaunay. Flag 2 indicates the edge is a segment and does not have to be flipped. Flag 3 indicates the edge is not a segment and has to be flipped because it is crossed by a segment. Size $3(2n+1)$.

`EdgeToFlip`. Flags (0,1,2,3,4,5,6) to record, for each triangle, which is the edge that will be flipped (0,1,2), or no edge will be flipped (3) or the edge that will be flipped by an adjacent triangle (4,5,6). In the latter case, it is necessary to subtract 4 to determine the edge that really will be flipped. Size $2n+1$.

Next we explain in detail each step of the algorithm.

1.1 Location step

For each point p the triangle `ContainingTriangle`$[p]$ is updated as follows: If point P=`Point`$[p]$ has not yet been inserted into the triangulation (`Inserted`$[p] = 0$), a walking process is launched until the triangle t that really contains P is reached. If P lies on an edge, t is taken as the triangle adjacent to the edge with the lowest index. Then, `ContainingTriangle`$[p]$ is updated with t and `VertexToInsert`$[t]$ is updated with p. Note that `VertexToInsert`$[t]$ can be updated simultaneously by distinct processors. In this manner, `VertexToInsert`$[t]$ contains the last point reaching t.

1.2 Insertion step

For each triangle t, the point of index p =`VertexToInsert`$[t]$ is inserted into the triangulation as follows: Let p_i =`Triangles`$[3t+i]$ $(i = 0..2)$ be the vertex indices of the triangle t. Triangle t will be triangulated to triangle t with vertex indices p_0, p_1 and p, triangle $2p+1$ with vertex indices p_1, p_2 and p, and triangle $2p+2$ with vertex indices p_2, p_0 and p. To properly determine the neighbours of these three new triangles, this step needs to be subdivided into two parts.

In the first part, for each triangle t with a point to be inserted p, the future neighbours of the triangles `Neighbours`$[3t+1]$ and `Neighbours`$[3t+2]$ are updated according to the insertion of p in t.

In the second part, for each triangle t, if t has a point p to be inserted, the arrays `Triangles`, `Neighbours` and `Flip` corresponding to the triangles t, $2p+1$ and $2p+2$ are updated according to the insertion of p in t. Otherwise, the neighbours of t are simply updated with the triangles previously stored in `FutureNeighbours`$[3t..3t+2]$.

```
foreach p < n+3 do                          /* in parallel */
    if Inserted[i] = 0 then
        t =ContainingTriangle[p];
        P =Point[p];
        Found=false;
        while Found=false do
            [P0, P1, P2] = Points[Triangles[3t..3t+2]];
            if P contained in some segment PjPj+1
            and Neighbours[3t+j]<t then
            Found=true; t=Neighbours[3t+j];
            else if P contained in triangle P0P1P2
            then Found=true;
            else
                C = (P0 + P1 + P2)/3;
                Seek for j such that segment Cp
                intersects segment PjPj+1;
                t =Neighbours[3t+j];
        ContainingTriangle[p]=t;
        VertexToInsert[t]= p;
```

Algorithm 1: Location

```
foreach t < 2n+1 do                         /* in parallel */
    p=VertexToInsert[t];
    if i ≠ −1 then foreach j = 1..2 do
        t′=Neighbours[3t+j];
        j′=opposite(j);
        FutureNeighbours[3t′+j′]=2p+j;
```

Algorithm 2: Insertion. Part 1.

```
foreach t < 2n+1 do                         /* in parallel */
    p=VertexToInsert[t];
    if i ≠ −1 then
        Inserted-P[i]=1;
        [p0, p1, p2]=Triangles[3t..3t+2];
        [n0, n1, n2]=FutureNeighbours[3t..3t+2];
        [f0, f1, f2]=Flip[3t..3t+2];
        Triangles[3t..3t+2]=[p0, p1, p];
        Triangles[3(2p+1)..3(2p+1)+2]=[p1, p2, p];
        Triangles[3(2p+2)..3(2p+2)+2]=[p2, p0, p];
        Neighbours[3t..3t+2]=[n0, 2p+1, 2p+2];
        Neighbours[3(2p+1)..3(2p+1)+2]=
        [n1, 2p+2, t];
        Neighbours[3(2p+2)..3(2p+2)+2]=
        [n2, t, 2p+1];
        Flip[3t..3t+2]=[f0, −1, −1];
        Flip[3(2p+1)..3(2p+1)+2]=[f1, −1, −1];
        Flip[3(2p+2)..3(2p+2)+2]=[f2, −1, −1];
    else
        Neighbours[3t..3t+2]=
        FutureNeighbours[3t..3t+2];
```

Algorithm 3: Insertion. Part 2.

1.3 Marking step

In this step the edges corresponding to a PSLG segment and the candidate edges to be flipped are marked. An edge can be a candidate due to being crossed by a segment or being non-locally Delaunay. Accordingly, this step is subdivided into two parts:

In the first part, for each non inserted segment s whose both endpoints have already been inserted into the triangulation, if the endpoints of s are connected by an edge, this edge is marked as constrained. Otherwise, a walking process starting from the first endpoint is launched until a flippable edge crossed by s and not flipped in the previous flipping step is found. Then, this edge and its opposite edge are marked to be flipped. The same is done starting from the other endpoint.

```
foreach s < m do                                   /* in parallel */
    if Inserted-P[s]=0 and
    Inserted-P[Segments[2s]]=1 and
    Inserted-P[Segments[2s+1]]=1 then
        for j = 0..1 do
            if j = 0 then
            [i0, i1]=Segments[2s..2s+1];
            else [i1, i0]=Segments[2s..2s+1];
            [P0, P1]=Points[i0..i1];
            starting from ContainingTriangle[i0] seek
            for the triangle of index t incident to i0
            intersecting segment P0P1;
            if P1 is a vertex of t then
                j=edge of t connecting P0 and P1;
                t'=Neigbours[3t+j];
                j'=opposite(j);
                Flip[3t+j]=Flip[3t'+j']=2;
                Inserted-P[s]=0;
            else
                Found=false;
                while Found=false do
                    j=edge of t intersecting segment
                    P0P1;
                    if j is flippable and different to
                    EdgeToFlip[t] then
                        Found=true;
                        t'=Neigbours[3t+j];
                        j'=opposite(j);
                        Flip[3t+j]=Flip[3t'+j']=3;
                    else
                        t=Neigbours[3t+j];
```

Algorithm 4: MARKING. PART 1.

In the second part, for each triangle t not crossed by any segment (`Flip`[$3t$] mod 2+`Flip`[$3t+1$] mod 2+`Flip`[$3t+2$] mod 2=0) the non constrained edges that are non Delaunay are marked to be flipped.

1.4 Flipping step

During this step, at most one edge of each triangle t is flipped. To avoid conflicts between concurrent flips, this step needs to be subdivided into three parts.

```
foreach t < 2n+1 do                                /* in parallel */
    if ∑_{k=0}^{2} (Flip[3t+k] mod 2) = 0 then for
    j = 0..2 do
        if Flip[3t+j]≠ 2 and no Delaunay(j) then
        Flip[3t+j]=1;
```

Algorithm 5: MARKING. PART 2.

In the first stage, a marked edge whose opposite edge is the only marked edge in its triangle t' is sought. If this edge exists, let nc be the number of marked edges of t. Then, if $nc \geq 2$ or ($nc = 1$ and $t < t'$) the edge is stored in `EdgeToFlip`[t].

In the second stage, if t has an edge to be flipped (`EdgeToFlip`[t]≤ 2), the future neighbours of the triangles adjacent to the quadrilateral determined by the edge are updated accordingly to the future flip of the edge. Otherwise, if the opposite edge of an edge j of t is to be flipped, $4+j$ is stored in `EdgeToFlip`[t].

In the third stage, if t has an edge to be flipped (`EdgeToFlip`[t]$\leq$ 2), the arrays `Triangles`, `Neighbours` and `Flip` corresponding to the triangles t and t' are updated according to the flip. Otherwise, if any adjacent triangle has an edge to be flipped, the array `Neighbours` is updated.

```
foreach t < 2n+1 do                                /* in parallel */
    EdgeToFlip[t]=3;
    nc=∑_{k=0}^{2} (Flip[3t+k] mod 2);
    if nc > 0 then
        j = 0;
        Found=false;
        while j ≤ 2 and no Found do
            if Flip[3t+j]=1 then
                t'=Neigbours[3t+j];
                j'=opposite(j);
                nc'=∑_{k=0}^{2} (Flip[3t'+k] mod 2);
                if Flip[3t'+j']=1 and nc' = 1 and
                (nc ≥ 2 or (nc = 1 and t < t')) then
                    EdgeToFlip[t]=j;
                    Found=true;
            j++;
```

Algorithm 6: FLIPPING. PART 1.

2 Results

The algorithm was implemented using OpenCl 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. Each one of the algorithm's parts was written in a kernel. The algorithm was executed ten times on two different models (Chairs32×32 and Holland4×4) and compared with *Triangle* [3], one of the most popular computational geometry software. The model Chairs32×32 is formed by 1024 copies of the model showed in the Figure

```
foreach t < 2n + 1 do                    /* in parallel */
    if EdgeToFlip[t]≤ 2 then
        j=EdgeToFlip[t]; j1=j + 1 mod 3;
        n=Neighbours[3t + j1]; nj1=opposite(j1);
        FutureNeighbours[3n + nj1]=t′;
        t′=Neighbours[3t + j]; j′=opposite(j);
        j1′=j′ + 1 mod 3;
        n′=Neighbours[3t′ + j1′]; nj1′=opposite(j1′);
        FutureNeighbours[3n′ + nj1′]=t;
    else
        for j=0..2 do
            t′=Neighbours[3t + j];
            if EdgeToFlip[t′]≤ 2 and
            Neighbours[EdgeToFlip[t′]]=t then
            EdgeToFlip[t]=4 + j;
```

Algorithm 7: FLIPPING. PART 2.

```
foreach t < 2n + 1 do                    /* in parallel */
    if EdgeToFlip[t]≤ 2 then
        j=EdgeToFlip[t];
        t′=Neighbours[3t + j]; j′=opposite(j);
        j1=j + 1 mod 2; j2=j + 2 mod 2;
        j1′=j′ + 1 mod 2; j2′=j′ + 2 mod 2;
        p2=Triangles[3t + j2]; p3=Triangles[3t′ + j2′];
        Triangles[3t + j1]=p3; Triangles[3t′ + j1′]=p2;
        ContainingTriangle[p3]=t;
        ContainingTriangle[p2]=t′;
        n1=FutureNeigbours[3t + j1];
        n2=FutureNeigbours[3t + j2];
        n1′=FutureNeigbours[3t′ + j1′];
        n2′=FutureNeigbours[3t′ + j2′];
        Neigbours[3t + j]=n1′; Neigbours[3t + j1]=t′;
        Neigbours[3t + j2]=n2;
        Neigbours[3t′ + j′]=n1;
        Neigbours[3t′ + j1′]=t;
        Neigbours[3t′ + j2′]=n2′;
        FutureNeighbours[3t..3t + 2]=
        Neighbours[3t..3t + 2];
        FutureNeighbours[3t′..3t′ + 2]=
        Neighbours[3t′..3t′ + 2];
        if Flip[3t + j]=3 then EdgeToFlip[t]=j1;
        else EdgeToFlip[t]=3;
    else if EdgeToFlip[t]= 3 then
        Neighbours[3t..3t + 2]=
        FutureNeighbours[3t..3t + 2];
    else
        j=EdgeToFlip[t]-4; j1=j + 1 mod 2;
        if Flip[3t + j]=3 then EdgeToFlip[t]=j1;
        else EdgeToFlip[t]=3;
    for j=0..2 do if Flip[3t + j]≠ 2 then
    Flip[3t + j]=0;
```

Algorithm 8: FLIPPING. PART 3.

1 arranged in a 32 by 32 array, while the model Holland4×4 is formed by 16 copies of the model showed in the Figure 2 arranged in a 4 by 4 array. The mean running times are presented in Table 1. Our future task is to study the performance of our algorithm versus [2]. However, our approach does not make use of the huge memory space needed by [2] to store the required digital Voronoi diagram.

	Chairs32×32	Holland4×4
Num. Vertices	1444864	1015344
Num. Segments	739328	1007488
Mean time (ms)	4028	2306
Triangle (ms)	12835	10254

Table 1: Behaviour of the proposed algorithm.

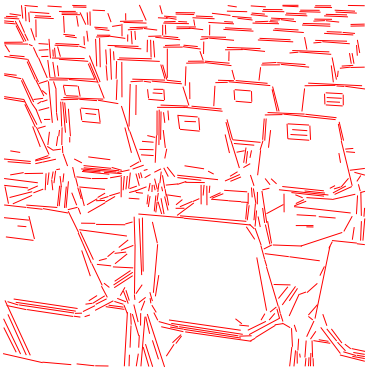


Figure 1: Cell of the model Chairs32×32

Figure 2: Cell of the model Holland4×4

References

[1] N. Coll, M. Guerrieri, Parallel Delaunay triangulation based on Lawson's incremental insertion, in: *Proceedings of the XIV Spanish Meeting on Computational Geometry*, CRM Documents, 8, Centre de Recerca Matemàtica, Bellaterra (Barcelona), 2011, 169–172.

[2] M. Qi, T. Cao, T. Tan, Computing 2D constrained Delaunay triangulation using the GPU, in: *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '12, ACM, New York, NY, USA, 2012, 39–46.

[3] http://www.cs.cmu.edu/ quake/triangle.html.

Metaheuristic approaches for the Minimum Dilation Triangulation problem

Maria Gisela Dorzán*[1], Mario Guillermo Leguizamón†[1], Efrén Mezura-Montes‡[2], and Gregorio Hernández§[3]

[1]Universidad Nacional de San Luis - Argentina
[2]Universidad Veracruzana - México
[3]Universidad Politécnica de Madrid - España

Abstract

We focus on the development of approximated algorithms to find high quality triangulations of minimum dilation because the complexity status of the Minimum Dilation Triangulation problem for a general point set is unknown. We propose an operator to generate the neigborhood which is used in different algorithms: Local Search, Iterated Local Search, and Simulated Annealing. Besides, an algorithm called Random Local Search is presented where good and bad solutions are accepted using the previous mentioned operator. We use the Sequential Parameter Optimization method for tuning the parameters of the SA algorithm. We compare our results with the only available algorithm found in the literature that uses the obstacle value to sort the edges in the constructive process. Through the experimental evaluation and statistical analysis, we assess the performance of the proposed algorithms using this operator.

Introduction

Different measures are adopted to design optimal triangulations. The most popular are the weight, stabbing number, area, and dilation. Although the usage of approximated approaches to solve complex optimization problems is common nowadays, this last measure has not been used as selection criterion when optimizing triangulations by using metaheuristic algorithms [10].

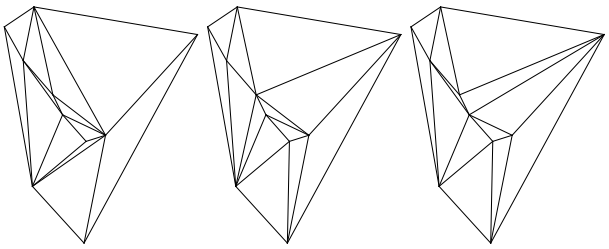
Let S be a finite planar point set, T a triangulation of S and u, v two points in S. There are two distance metrics: the Euclidean distance between u and v, $|uv|$, and the length of the shortest path between u and v with in the triangulation T, $dist_T(u, v)$. The dilation between u and v with respect to T is the ratio between the shortest path and the Euclidean distances between u and v, and is defined as $\Delta_T(u, v) = \frac{dist(u,v)}{|uv|}$.

The maximum over all the dilations between pairs of points in T is called the dilation of T (or stretch factor) and is represented by $\Delta(T)$. The best possible dilation of any triangulation of S is the dilation of S and is denoted by $\Delta(S)$. Thus, we have

$$\Delta(T) = \max_{u,v \in S} \Delta_T(u, v) \text{ and } \Delta(S) = \min_{T \text{of} S} \Delta(T)$$

The triangulation T^* whose dilation is the dilation of S, i.e. $\Delta(T^*) = \Delta(S)$, is called minimum dilation triangulation of S. Note that there are several triangulations of minimum dilation for a given set of points.

Computing the minimum dilation triangulation for a set of points in the plane is listed as an Open Problem in Eppstein's survey [6], i.e., no polynomial algorithm that can build it is known and no proof that the problem is NP-hard is shown. Neither the greedy nor the delaunay triangulation algorithms generate the minimum dilation triangulation of a planar point set, as shown in Figure 1. Therefore, one approach is to use metaheuristic techniques for obtaining approximate solutions to the optimum.



(a) GT (b) DT (c) MDT

Figure 1: A point set for which the MDT neither is the GT nor the DT.

*Email: mgdorzan@unsl.edu.ar. Research supported by CONICET and Research Project No. 22/F014

†Email: legui@unsl.edu.ar. Research supported by Laboratorio de Investigación y Desarrollo en Inteligencia Computacional (LIDIC)

‡Email: emezura@uv.mx. Research supported by CONACyT bilateral project No. 164626.

§Email: gregorio@fi.upm.es. Research supported by ESF EUROCORES programme EuroGIGA - ComPoSe IP04 - MICINN Project EUI-EURC-2011-4306

1 Approximated algorithms for the MDT problem

A set of simple techniques is presented because, to the best of the authors' knowledge, there are no works in the literature where the MDT problem is approached using metaheuristics. We present the general overview of the studied algorithms: Greedy [5], Local Search [1, 11], Iterated Local Search [9], Simulated Annealing [3, 7], and Random Local Search describing their features and parameters. For all the algorithms the solution space $\mathcal{E}$ is represented by all possible triangulations of a set S of n points in the plane. An $n \times n$ matrix of ones and zeros is used to represent a possible solution. A 1 at position (i, j) means that there is an edge connecting points i and j, otherwise a 0 is placed. The objective function, $f : \mathcal{E} \rightarrow R$, assigns a real value to each element of $\mathcal{E}$. For each $E \in \mathcal{E}$, the function f is defined as the maximum dilation among all pairs of points in E.

Greedy Algorithm (*G-MDT*) It starts with an empty solution *Sol* and inserts edges until a triangulation for a given set of point S is generated. Let A be the set of all possible edges that have not been inserted in *Sol*, $u, v \in S$ and $e = uv \in A$. If the dilation of the points u and v determines the dilation of the solution, i.e., $\Delta_{Sol}(u, v) = \Delta(Sol)$, then the edge e is inserted in the solution *Sol*. This reduces the dilation of the points u and v to 1 because $dist(u, v) = |uv|$ and consequently decreases the dilation of the solution *Sol*. The insertion is performed whenever the new edge to insert produces no intersections with the edges already inserted.

Local Search Algorithm (*LS-MDT*) It starts from a random initial solution and then iteratively moves to a neighbor solution. These moves are performed by applying an operator looking for a better solution. If a better solution is found, it replaces the current solution by the new one and it continues the process until it can not improve the current solution. When no improved solutions are present in the neighborhood, local search is stuck at a local optimal solution. The moves in the search space are performed using the *retriang()* operator that works as follows. Let $a, b \in S$, if the dilation of the points a and b determines the dilation of the current solution *Sol*, i.e., $\Delta_{Sol}(a, b) = \Delta(Sol)$ (see Figure 2(a)), then a and b are joined by an edge. The egdes intersected by ab are deleted and the region delimited by these edges (see Figure 2(b)) is retriangulated in a greedy way. The edge whose points have the higher dilation is inserted at each step if it does not intersect with the edges previously added and if a complete triangulation is not achieved (see Figure 2(c)). Therefore the current solution is improved because its dilation has been reduced. Due to the behavior of this operator only a single neighbor is obtained.

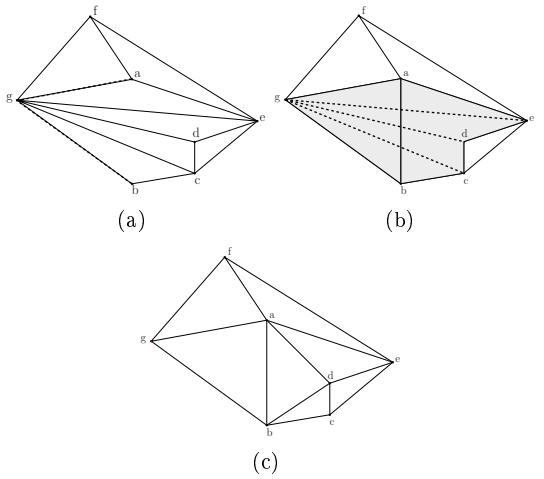


Figure 2: *retriang()* operator. In (a), $\Delta_{Sol}(a, b) = \Delta(Sol)$ is shown in dashed style. In (b), the edge ab is added and the dashed edges intersected by ab are deleted. The grey region is retriangulated in a greedy way (c).

Iterated Local Search Algorithm (*ILS-MDT*) First, a local search is applied to an initial random solution and yields a local optimum using the *LS-MDT* algorithm. At each iteration, a perturbation of *Sol* is carried out and then, a local search is applied to the perturbed solution Sol'. This process iterates until the number of perturbations is less than 50 and the current solution is improved, always keeping the best solution found so far Sol_{best}. The *perturb(Sol)* procedure is performed by the perturbation operator where $n/5$ random local retriangulations over random selected points of S are performed worsening the current solution, i.e., a point $b \in S$ is randomly chosen and all the points adjacent to b form the grey region that has to be retriangulated. The edges belonging to the region are deleted and then this region is retriangulated in a random way. The edges of the region are inserted at random if they do not intersect with the edges previously added and if a complete triangulation is not achieved.

Simulated Annealing Algorithm (*SA-MDT*) From an initial solution, it executes a number of iterations where a neighbor of the current solution *Sol* is generated in each one of them. The moves that improve the cost function are always accepted. Otherwise, the neighbor Sol' is selected with a given probability that depends on the current temperature T. This probability is usually called *acceptance function* and it is evaluated according to $p(T, Sol, Sol') = e^{-\frac{\delta}{T}}$

where $\delta = f(Sol') - f(Sol)$. $M(T_k)$ moves are performed for each temperature T_k. Finally, the value of T_k is decreased at each algorithm iteration k. The algorithm continues this way until the termination condition is met.

We use two operators to obtain the neighborhood of a solution: 1) Local random retriangulation (RR) as used in the perturbation operator in *ILS-MDT* algorithm; 2) Local greedy retriangulation (GR) is similar to the RR described above, but the edges are inserted in a greedy way. A point $b \in S$ is randomly chosen and all the points adjacent to b are recovered. Then the grey region that forms the adjacent points to b is retriangulated using at each step the edges whose points have the higher dilation (if it does not intersect with those previously added). The scheduling of application for these operators is as follows. GR is performed g times and RR is performed r times. We considered $g \in \{4, 6, 8\}$ and $r = 2$ as will be described later in the next section.

Due to the complexity involved in tuning the parameters of metaheuristic techniques, we used Sequential Parameter Optimization (SPO) [2] for tuning the parameters required by SA. All SPO-tuning experiments for the SA-MDT algorithm were performed with the following settings: 50 sequence steps, 5 new design points in each step, up to 2 repeats per design point, and 7 initial design points. Random Forest was used as a fast surrogate model building tool. Latin hypercube sampling was chosen as the generator of design points. We obtained better results with T_k moves at each temperature ($M(T_k) = T_k$), using local retriangulation interleaving with $g = 8$ and $r = 2$ and with a initial temperature equal the maximum length of the paths in the initial solution.

Random Local Search Algorithm (*RLS-MDT*) Initially, the proposed SA algorithm used the operator *retriang()* to move in the solution space. We observed that such algorithm converged too early to suboptimal regions (the best solution was achieved during the first temperature value). Therefore we proposed another operator for the SA algorithm to avoid this condition of premature convergence. On the other hand, the results obtained with operator *retriang()* were encouraging with the LS and ILS algorithms. Therefore, we propose a new algorithm; we called it Random Local Search algorithm, which starts with a random initial solution. $maxEvals = 2n$ evaluations are performed, accepting good and bad solutions. This mechanism allows to explore and exploit the search space, always keeping the best solution found so far Sol_{best}. We consider the operator *retriang()* but the retriangulation of the region is calculated in a random manner because better results were obtained that way.

2 Experimental Evaluation and Statistical Analysis

We generated 20 problem instances, each one is called n-i, where $n = \{40, 80, 120, 160, 200\}$ is the size of the instance and i is the number of the instance ($1 \leq i \leq 4$). The points are randomly generated, uniformly distributed and for each point (x, y), the coordinates $x, y \in [0, 1000]$. For stochastic algorithms 25 runs were carried out with different initial seeds and different initial triangulations for each run.

Table 1 shows the best values obtained with the proposed algorithms: *G-MDT*, *LS-MDT*, *ILS-MDT*, *SA-MDT*, and *RLS-MDT*. The lowest dilations are bolded. Xia proved that the dilation of the Delaunay triangulation of a set of points in the plane is less than 1.998 [12]. All the results obtained by the algorithms are less than 1.998 and the Delaunay triangulation never obtains better results for the instances considered (see the "*DT*" column). We compared our results with those obtained with the algorithm proposed in [8]. We called it *OV-MDT* because it computes the Obstacle Value of the edges for approximating the minimum dilation triangulation. We used the Java-applet available at the Geometry Lab site at the University of Bonn (www.geometrylab.de) to obtain the results. For some instances the applet did not produce a result giving an error and halting the execution of the algorithm. It should be noted that the applet yielded results after several days of execution (see the "*OV-MDT*" column). The *G-MDT* algorithm obtained solutions of poor quality even for the smallest instances. The *RLS-MDT* algorithm obtained the lowest (best) dilations for all problem instances (except for one instance of 120 points). We observed that the *OV-MDT* algorithm did not obtain better results than *RLS-MDT*. Although in some instances equal best values were obtained by some algorithms, the *RLS-MDT* algorithm showed to be less complex and faster than the other algorithms.

As all samples had a non-Normal distribution (Kolmogorov-Smirnov test) we used non-parametric statistical tests to evaluate the confidence on the statistical results provided by the algorithms. In order to carry out a comparison which involved results from more than two algorithms Kruskal Wallis test was used for multiple comparisons with independent samples [4]. We used the post-hoc Tukey test to find which algorithms' results are significantly different from one another. These tests are well-known and they are usually included in standard statistics packages (such as *Matlab*, *R*, etc.). After applying the Kruskal Wallis test, we observed that there was always a significant difference between the algorithms (1.e-16 $< p$-value ≤ 0.0015) being this difference much larger when considering sets with more points. The *ILS-MDT* and *RLS-MDT* algorithms had similar per-

Instance	*DT*	*OV-MDT*	*G-MDT*	*LS-MDT*	*ILS-MDT*	*SA-MDT*	*RLS-MDT*
40-1	1.31871	132.863	1.37203	130.959	**1.29467**	**1.29467**	**1.29467**
40-2	1.37491	**1.36881**	1.37491	**1.36881**	**1.36881**	**1.36881**	**1.36881**
40-3	**1.32268**	**1.32268**	1.36026	**1.32268**	**1.32268**	**1.32268**	**1.32268**
40-4	1.35391	**1.32330**	1.34919	**1.32330**	**1.32330**	1.32557	**1.32330**
80-1	1.33034	**1.32363**	1.39394	**1.32363**	**1.32363**	1.40844	**1.32363**
80-2	1.40500	135.181	1.38199	1.35339	**1.32418**	1.50331	**1.32418**
80-3	1.37791	**1.30519**	1.36180	1.34065	1.31457	1.37791	**1.30519**
80-4	1.42547	134.239	1.39362	1.33176	**1.31583**	1.47561	**1.31583**
120-1	1.37428	**1.34442**	1.42386	1.35398	1.34825	1.72082	**1.34442**
120-2	1.33973	131.194	1.37778	1.31194	**1.30366**	1.65921	1.31194
120-3	1.37724	134.016	1.43027	1.40314	1.31985	1.74912	**1.29786**
120-4	1.38529	**1.33600**	1.39972	1.35152	1.34272	1.68163	**1.33600**
160-1	1.34365	**1.31050**	1.43076	1.35236	1.33384	1.95530	**1.31050**
160-2	1.35409	*NA*	**1.33260**	1.36463	**1.33260**	1.97896	**1.33260**
160-3	1.35797	133.278	1.37723	1.37178	1.36352	1.87574	**1.32995**
160-4	1.37922	**1.34941**	1.37922	1.37922	1.35037	1.87033	**1.34941**
200-1	**1.35751**	*NA*	1.42220	1.41867	**1.35751**	2.08064	**1.35751**
200-2	1.39512	*NA*	1.39512	1.36451	**1.36350**	2.06324	**1.36350**
200-3	141.962	*NA*	1.45763	1.40645	**1.40645**	2.09818	**1.40645**
200-4	1.36965	*NA*	1.40765	1.35499	**1.35251**	2.00110	**1.35251**

Table 1: Best results of *DT*, *OV-MDT*, *G-MDT*, *LS-MDT*, *ILS-MDT*, *SA-MDT*, and *RLS-MDT* algorithms.

formances for the whole set of instances because there were no significant differences between these two algorithms (using the Tukey test).

3 Conclusions

In this work four metaheuristic techniques were implemented to find high quality triangulations of minimum dilation. The set of instances generated and used in the experimental evaluation are available at the research project site (www.dirinfo.unsl.edu.ar/bd2/GeometriaComp/)

After performing the experimental evaluation and statistical analysis, we assessed the applicability of the LS, ILS, SA, and RLS algorithms for the MDT problem. The *RLS-MDT* and *ILS-MDT* algorithms had similar competitive performances with respect to the other algorithms in the problem instances considered. These algorithms achieved a dilation reduction between 0.4% and 9.2% with respect to the greedy strategy (*GT-MDT*). Although both algorithms behave similarly, the *RLS-MDT* algorithm required less evaluations than *ILS-MDT*. It should be noted that the existing algorithm (*OV-MDT*) yielded no results for four of the instances considered as either it returned an error or halted. The *RLS-MDT* algorithm outperformed the *OV-MDT* algorithm in 50% of the instances considered and obtained the same results in the other instances.

References

[1] E. Aarts and J. Lenstra. *Local Search in Combinatorial Optimization*. John Wiley, 1997.

[2] T. Bartz-Beielstein. *Experimental Research in Evolutionary Computation: The New Experimentalism (Natural Computing Series)*. Springer, 2006.

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

[4] J. Derrac, S. García, D. Molina, and F. Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3 – 18, 2011.

[5] J. Edmonds. Matroids and the greedy algorithm. *Mathematical Programming*, 1(1):127–136, 1971.

[6] D. Eppstein. Spanning trees and spanners. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, chapter 9, pages 425–461. Elsevier, 2000.

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

[8] A. Klein. Effiziente berechnung einer dilationsminimalen triangulierung. Master's thesis, 2006.

[9] O. Martin, S. Otto, and E. Felten. Large-step markov chains for the traveling salesman problem. *Complex Systems*, 5:299–326, 1991.

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

[11] C. Papadimitriou. *The complexity of combinatorial optimization problems*. PhD thesis, Princeton, NJ, USA, 1976. AAI7704795.

[12] G. Xia. Improved upper bound on the stretch factor of delaunay triangulations. In *Proceedings of the 27th annual ACM symposium on Computational geometry*, SoCG '11, pages 264–273, New York, NY, USA, 2011. ACM.

Three location tapas calling for CG sauce

Frank Plastria*[1]

[1]MOSI - Vrije Universiteit Brussel,Pleinlaan 2, B 1050 Brussels, Belgium

Abstract

Based on some recent modelling considerations in location theory we call for study of three CG constructs of Voronoi type that seem not to have been studied much before.

Introduction

There is a strong interrelation and mutual enseminations between (continuous) location theory and computational geometry. The first generates interest into distance optimization problems with geometrical interpretations, the second builds geometrical algorithms for efficient solution to the first's problems.

In this talk I will shortly present some recent work stemming from location theory calling for study of three novel (?) computational geometry constructs.

1 Mixed norm shortest paths

The fact that the distance measure may be different from one region to another, contrary to the usual assumption that distance is measured by a single norm, has been acknowledged in only a few location studies. Parlar [14] considers the plane divided by a linear boundary with at one side ℓ_2 and at the other side ℓ_1. Brimberg et al [2] consider a bounded region with a different norm inside and outside, focusing in particular on an axis-parallel rectangular city with ℓ_1 inside and ℓ_2 outside.

Brimberg et al. [1] and Zaferanieh et al. [19] consider location in a space with two distinct ℓ_p norms in complementary halfplanes. The way to calculate the distance in such a space was studied more in detail by Franco et al [7].

Fathali and Zaferanieh [5] extend this work to include more general block norms. Fliege [6] considers differentiably changing metrics similar to Riemann spaces.

Unfortunately part of this work is wrong. All authors consider only two possibilities when calculating distances: when the two points lie in the same halfplane simply use the corresponding distance, and otherwise in two steps via the best possible point on the separating line. Although this is true in some particular cases, e.g. the axis-parallel rectangular city case evoked before (but without inflation factors), Parlar already observed that in general when calculating distance in this naive way distance to a fixed point is not continuous everywhere. Worse: triangle inequality may be violated. This clearly shows that such distance calculation cannot be correct, but none of the authors try to resolve this discrepancy.

What should rather be done is to consider shortest path distance in the space, similar to what is done in the so-called weighted (euclidean) region problem well known in CG since the original paper of Mitchell and Papadimitriou [10] (which depicts a clear counterexample to the naive distance above): the length of the shortest possible piecewise linear path using the adequate measure (speed in that paper) in each piece.

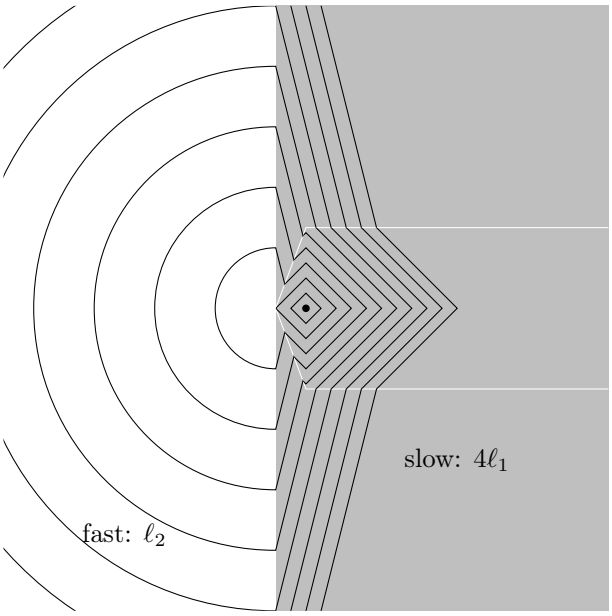
For two halfspaces with arbitrary distinct norms or gauges we studied more in detail the optimality conditions when crossing the boundary, generalizing Snell's law in optics. This has nice geometric interpretation and leads to a geometrical view on deriving such distances [16].

What turns out to be crucial is the comparison between the two norms along (the direction of) the separating line. As long as these are equal things remain relatively simple and the naive assumption about distances is correct. However, when they differ things change. One region is then 'slower' than the other (as measured along the boundary line). And in such a case some paths connecting two points within the slowest region will consist of three pieces. They 'hitch a ride' along the quicker boundary.

Thereby continuity of the distance is again guaranteed, but now convexity is partially lost, as illustrated by the balls in the figure below. At right of the vertical separation line we have the slower region in gray with distance measure $4\ell_1$, at left the faster region with norm ℓ_2. The figure shows balls of increasing mixed-norm radius centered at the dot. For very small radius we have the diamond-shaped ℓ_1 ball. As soon as the radius allows to reach the boundary a part of circular shape arises at left due to two piece shortest paths, which spills over with a linearly moving front at the right corresponding to three piece paths. The white line shows the set of meeting points where one-piece

*Email: Frank.Plastria@vub.ac.be

and three-piece paths yield equal distance.



It should be noted that Brimberg et al [2] correctly prove convexity of the distance from a fixed point to all points of the other region, but do not acknowledge that this convexity (quite crucial for their subsequent optimization approaches) does not extend to the whole plane if the fixed point lies in the slower region.

The linearly separated two-norm situation is of course only the first step. For adequate description of some reality one will have to consider a plane split into cells each with their own norm or gauge (for asymmetry). How to efficiently calculate shortest path distances in such a context seems to be largely open, apart from some work on approximations, see Cheng et al. [3]. Correctly attacking location problems in such mixed norm spaces is another matter all together. A first step will probably be to look at Voronoi diagrams in such environments. Clearly a lot of opportunities for CG.

Interesting applications may be found in fire-fighting using models to simulate the quite different ways fires spread in various circumstances, taking into account vegetation like (ir)regularly spaced plantations of varying types, influence of terrain inclination, and weather conditions, such as winds and humidity, and so forth.

2 Knapsack Voronoi diagrams

Let S be a finite set of sources $s \in \mathbb{R}^2$ with capacities $c_s > 0$. Consider the location of a central point x in the plane that should be connected to sufficiently many of the sources to be able to supply given demand D. The supply-weighted sum of distances should be minimized, possibly together with some additional costs $f(x)$:

$$\min \quad \sum_{s\in S} w_s d(s,x) + f(x) \tag{1}$$

$$\sum_{s\in S} w_s \geq D \tag{2}$$

$$0 \leq w_s \leq c_s \tag{3}$$

$$x \in \mathbb{R}^2 \tag{4}$$

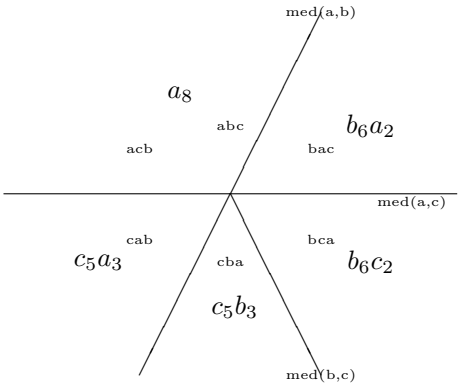
This is a continuous single facility location-allocation problem that I have been studying under several variants for $f(x)$, most of the time consisting of fixed weighted distances to demand point(s) [9, 17, 18, 15].

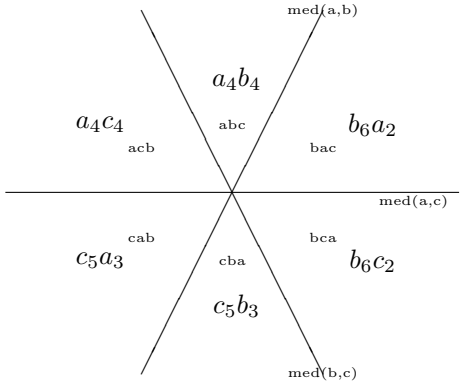
For any fixed location x finding the best allocation $W = (w_s)_{s\in S}$ for the supplies is a continuous knapsack problem, easily solved by taking sources $s \in S$ at their full capacity $w_s = c_s$ in non-increasing order of distance $d(s,x)$, until the demand D is met. Only the last chosen source may contribute below its capacity, and all further sources will not contribute at all ($w_s = 0$). There are only a finite number of allocations W of such type possible.

For any fixed allocation W finding the corresponding best site x amounts to solve a single facility minisum location problem (Fermat-Weber problem) which may be easily done by various methods of convex optimization. And this x should then yield W as optimal allocation.

It is therefore of interest to know the regions with fixed allocation.

The planar subdivision corresponding to different solutions to this knapsack problem is what I call a Knapsack Voronoi diagram. In case all capacities are equal c, we obtain a traditional k-th order Voronoi diagram with $k = \lceil D/c \rceil$, with additional splits of some cells as soon as demand is not a multiple of capacity. When capacities differ the order k is not fixed and we have new types of diagrams. In particular vertices of the diagram may have from 3 up to 6 edges. The following examples show how vertices with 5 or 6 edges may arise. Demand is always $D = 8$, but in the first case $c_a = 10, c_b = 6, c_c = 5$, while in the second $c_a = 4, c_b = 6, c_c = 5$.





As long as the coefficients of the distances $d(s,x)$ in (1) are simply equal to w_s the edges of the corresponding Knapsack Voronoi diagram remain linear segments. But in some models such as the location of an assembly station [18] additional factors appear and then we need diagrams similar to multiplicatively weighted Voronoi diagrams, where cells have circular arcs as boundaries and may be disconnected.

This shows the interest of studying the properties and efficient ways to calculate such diagrams, clearly a task for CG.

More general constructs, such as multi-facility versions of previous location models may be considered where the allocation problem for fixed location(s) of the central facility(ies) to be located consists of a linear program with coefficients either fixed or depending only on the distances between sources and facility site(s). Such an LP can generically have only a finite number of optimal solutions, each corresponding to certain linear inequalities between the distances, so corresponding to (often empty) cells of a Voronoi-like diagram. I do not know of any study of such structures.

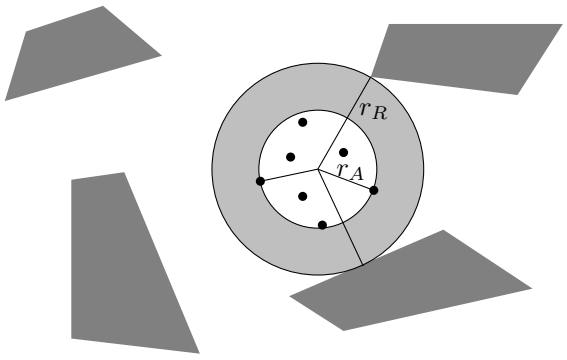
3 Push-pull Voronoi diagrams for points and polygons

Push-pull location problems try to find good sites within a given region for facilities that at the same time are far (pushed away) from some repelling points $r \in R$ and close (pulled) to some other attracting points $a \in A$.

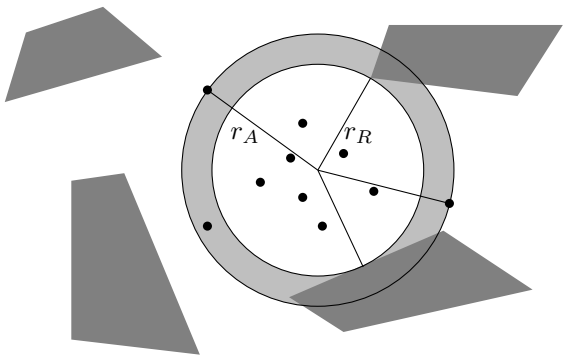
For euclidean distances Ohsawa [11] fully constructs the set of efficient points for this biobjective maximin-minimax problem. He shows that any such efficient point must lie on the boundary of cells obtained by intersecting the farthest-point Voronoi diagram w.r.t. R with the closest-point Voronoi diagram w.r.t. A within the given region. These line-segments may then be projected into two-dimensional value-space where an efficient CG boundary seeking technique finishes the job.

This work has been extended later first to rectangular distance ℓ_1 in [13], then to partial coverage problems in [12].

A few years ago together with José Gordillo and Emilio Carrizosa I studied [8] another similar push-pull location problem where the set R consists of extensive facilities described as polygonal regions. But instead of looking at the bi-objective problem that has a continuum of efficient solutions, we were looking for one particular solution that 'best' separates (if possible) R from A. Separation was measured in a support vector machine way (see e.g. [4]) by maximizing the difference $r_R^2 - r_A^2$ where r_R is the (euclidean) distance to the closest $r \in R$ and r_A the farthest distance to some $a \in A$. In the feasible case where A may be separated by a circle from all regions in R, i.e. when this objective may be positive, this means geometrically that we seek the largest area annulus enclosing all points of A and not meeting R. In the non feasible case the objective will always be negative and we look for the smallest area annulus that contains or overlaps all A and such that its 'hole' does not meet R. These two cases are illustrated below; the annulus is coloured as light gray.



(Feasible case)



(Unfeasible case)

For a site x we call 'active' any r closest to x (the actual closest point(s) of this r is also called active) and any a farthest from x . We showed that generically three cases may arise at an optimal solution:

(1) there are 4 active elements with at least one of each type, or there are 3 active elements with one a and two r (in the nonconvex case possible twice the same r with different active points) either (2) with colinear active points, or (3) one active r active at a vertex. Enumeration of all realizations of such cases leads to an $O(n^5)$ algorithm.

However, these conditions are directly related to the farthest point Voronoi diagram V_A w.r.t. A and the closest point Voronoi diagram V_R w.r.t. the polygons R. It is well known that edges of V_R are parts of bisectors between two polygons, and these consist of successive linear and parabolic pieces depending on whether the active points of both polygons are of the same type (a vertex or on an edge) or not. The points where these pieces touch are called breakpoints.

Now Case (1) may be realized in three ways: either as a vertex of V_A or as a vertex of V_R or as the point of intersection between an edge of V_A and an edge of V_R. Case (3) corresponds to a breakpoint of some edge of V_R and case (2) happens at a finite number of points easily constructed from V_R.

Therefore using the CG approach should lead to much lower complexity. But this CG approach to the problems remains to be done.

It should be noted that in an optimal solution to a multifacility version of this push-pull problem all sites will satisfy the same conditions, so the same set of candidate sites arises, but now several of such sites will have to be combined.

References

[1] J. Brimberg, H.T. Kakhki, G.O.Wesolowsky, Location among regions with varying norms, *Annals of Operations Research* **122** (2003), 87–102.

[2] J. Brimberg, H. Kakhki, G. Wesolowsky, Locating a single facility in the plane in the presence of bounded regions and different norms, *Journal of the Operational Research Society of Japan* **48** (2005) 135–147.

[3] S. Cheng, H. Na, A. Vigneron, Y. Wang, Approximate Shortest Paths in Anisotropic Regions, *SIAM Journal on Computing* (2008) **38** 802–824.

[4] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods.* Cambridge University Press, Cambridge, 2000.

[5] J. Fathali, M. Zaferanieh, Location problems in regions with l_p and block norms, *Iranian Journal of Operations Research* **2** (2011) 72–87.

[6] J. Fliege, *Efficient dimension reduction in multifacility location problems*, Shaker Verlag, Aachen, 1997.

[7] L. Franco, F. Velasco, L. Gonzalez-Abril, Gate points in continuous location between regions with different p norms, *European Journal of Operational Research* **218** (2012) 648–655.

[8] J. Gordillo, F. Plastria, E. Carrizosa, Locating a semi-obnoxious facility with repelling polygonal regions, working paper (2007) 30p. `http://www.optimization-online.org/DB_HTML/2007/04/1652.html`

[9] L. Kaufman, F.Plastria, The Weber problem with supply surplus, *Belgian Journal of Operations Research, Statistics and Computer Science* **28** (1988) 15–31.

[10] J.S.B. Mitchell, C.H. Papadimitriou, The weighted region problem: finding shortest paths through a weighted planar subdivision, *Journal of the ACM* **38** (1991) 18–73.

[11] Y. Ohsawa, Bicriteria Euclidean location associated with maximin and minimax criteria, *Naval Research Logistics* **47** (2000) 581–592

[12] Y. Ohsawa, F. Plastria, K. Tamura, Euclidean push-pull partial covering problems, *Computers & Operations Research* **33** (2006) 3566–3582

[13] Y. Ohsawa, K. Tamura, Efficient location for a semi-obnoxious facility *Annals of Operations Research* **123** (2003) 173–188

[14] M. Parlar, Single facility location problem with region-dependent distance metrics, *International Journal of Systems Sciences* **25** (1994) 513–525.

[15] F.Plastria, Up and downgrading the euclidean 1-median problem, in preparation (2013)

[16] F.Plastria, Shortest paths using varying distance functions, in preparation (2013)

[17] F.Plastria, M.Elosmani, On the convergence of the Weiszfeld algorithm for continuous single facility location-allocation problems, *TOP* **16** (2008) 388–406.

[18] F.Plastria, M.Elosmani, Continuous location of an assembly station, *TOP* First online (April 2011).

[19] M. Zaferanieh, H. Kakhki, J. Brimberg, G. Wesolowsky, A BSSS algorithm for the single facility location problem in two regions with different norms, *European Journal of Operational Research* **190** (2008) 79–89.

On the barrier-resilience of arrangements of ray-sensors

David Kirkpatrick*[1], Boting Yang†[2], and Sandra Zilles‡[2]

[1]Department of Computer Science, University of British Columbia, Vancouver, Canada.
[2]Department of Computer Science, University of Regina, Regina, Canada.

Abstract

Given an arrangement $\mathcal{A}$ of n sensors and two points s and t in the plane, the barrier resilience of $\mathcal{A}$ with respect to s and t is the minimum number of sensors whose removal permits a path from s to t such that the path does not intersect the coverage region of any sensor in $\mathcal{A}$. When the surveillance domain is the entire plane and sensor coverage regions are unit line segments, even with restricted orientations, the problem of determining the barrier resilience is known to be NP-hard [11, 12]. On the other hand, if sensor coverage regions are arbitrary lines, the problem has a trivial linear time solution. In this paper, we give an $O(n^2m)$ time algorithm for computing the barrier resilience when each sensor coverage region is an arbitrary ray, where m is the number of sensor intersections.

1 Introduction

Barrier coverage is an important coverage concept that arises in the analysis of wireless sensor networks [3]. Other notions of coverage try to quantify the effectiveness of a collection of sensors in detecting the presence of objects in a particular surveillance region. Barrier coverage, motivated by applications such as border control, measures the effectiveness of detecting the movement of objects *between*, but not necessarily *within*, critical regions. Given a sensor network, specified as an arrangement $\mathcal{A}$ of sensors with associated coverage regions in the plane, and two points s and t, we say that the sensor network provides *barrier coverage* if it guarantees that any object moving from point s to point t must be detected by at least one sensor.

If sensor regions are connected then determining barrier coverage amounts to asking if s and t belong to the same face of the arrangement, which is straightforward to check provided the sensor region boundaries are sufficiently simple. In order to measure robustness of barrier coverage, Kumar et al. [10] introduced *k-barrier coverage* that guarantees that any path from a point s to a point t must intersect at least k distinct sensor regions. They showed that for unit disk sensors (i.e., sensors whose coverage regions are unit disks) distributed in a strip separating s and t, k-barrier coverage can be determined efficiently by reduction to a maximum flow problem in the intersection graph of the disks.

Bereg and Kirkpatrick [2] introduced and studied the associated optimization problem, which they called *barrier resilience*. This specifies the minimum number of sensors whose removal permits a path from point s to point t such that the path does not penetrate any of the remaining sensor coverage regions. Bereg and Kirkpatrick showed that there is a 3-approximation (or, under mild restrictions concerning the separation of s and t, a 2-approximation) algorithm for computing the barrier resilience of unit disk sensors. When the sensor coverage regions are arbitrary line segments, Alt et al. [1] proved that the problem of determining barrier resilience is NP-hard and APX-hard. In fact, even if all sensor coverage regions are unit line segments in one of at most two orientations, the barrier resilience problem remains NP-hard [11, 12]. The reader is referred to the papers [3,5,8] for more information on barrier coverage and related problems.

It is straightforward to see that if sensor coverage regions are arbitrary lines, the barrier resilience problem has a linear time solution, since the resilience of a given arrangement of lines is just the number of lines that separate s and t.

*Email: kirk@cs.ubc.ca.
†Email: boting@cs.uregina.ca.
‡Email: zilles@cs.uregina.ca.
Research support for all three authors is provided by the Natural Sciences and Engineering Research Council of Canada.

In this paper, we address the case where sensor coverage regions are half-lines (or rays). We describe an $O(n^2m)$ time algorithm for computing the resilience of an arbitrary arrangement of n rays with m intersections. (Due to space constraints, proofs are omitted; full proofs will appear in an expanded version of the paper.)

2 Ray barriers and barrier graphs

Let $\vec{V}$ be a set of n rays, specified by an endpoint and a direction, in the plane. Suppose that we are given a sensor network consisting of n sensors, where the coverage regions of sensors correspond to the rays in $\vec{V}$, and two points s and t which are not intersected by any of the rays in $\vec{V}$. We consider the problem of finding a subset $\vec{U}$ of rays in $\vec{V}$ with the minimum cardinality such that there is a path from s to t which does not intersect any rays in $\vec{V}\backslash\vec{U}$. The cardinality of $\vec{U}$ is referred to as the *resilience* of the sensor configuration $(s,t,\vec{V})$, and $\vec{U}$ is a *resilience set* of $(s,t,\vec{V})$.[1]

We say that a set $\vec{V'} \subseteq \vec{V}$ forms a *barrier* for $(s,t,\vec{V})$ if any path from s to t intersects at least one of the rays in $\vec{V'}$. Thus a set $\vec{U} \subseteq \vec{V}$ is a resilience set of $(s,t,\vec{V})$ if and only if $\vec{U}$ is a smallest subset of $\vec{V}$ with the property that $\vec{V}\backslash\vec{U}$ does not form a barrier. Our algorithm for computing a resilience set uses a reformulation of the problem as a graph problem. This reformulation is based on the observation that if a set of rays forms a barrier it must contain a subset consisting of two rays that forms a barrier; we refer to such a subset as a 2-*barrier*.

In order to substantiate this observation, we introduce some helpful notation. For two points a and b in the plane, we use ab to denote the line segment with endpoints a and b, and use $\vec{a}$ to denote a ray with endpoint a.

For the remainder of this paper, suppose, without loss of generality, that the line containing st is horizontal. (Accordingly, we will say "above (resp., below) st" as an abbreviation for "above (resp., below) the horizontal line supporting st".) For any ray $\vec{a} \in \vec{V}$, we assign a unique color as follows: (i) if $\vec{a}$ intersects st and goes down we assign it the color *red* (drawn as a solid ray in figures); (ii) if $\vec{a}$ intersects st and goes up we assign it the color *blue* (drawn as a dashed ray in figures); and (iii) if $\vec{a}$ does not intersect st we assign it the color *black* (drawn as a dotted ray in figures).

[1] Note that a configuration $(s,t,\vec{V})$ does not necessarily have a unique resilience set.

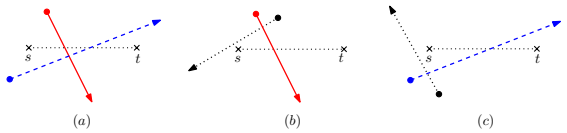


Figure 1: (a): a red-blue 2-barrier; (b): a red-black 2-barrier; (c): a blue-black 2-barrier

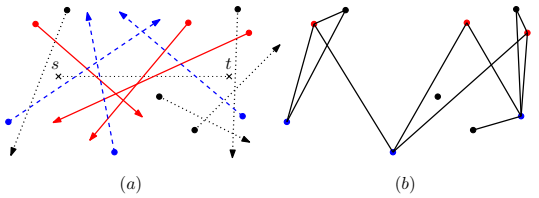


Figure 2: (a): a sensor configuration $(s,t,\vec{V})$; (b): its associated barrier graph

We first observe that certain pairs of intersecting rays are guaranteed to form a 2-barrier (see Figure 1).

Proposition 1 *Let $(s,t,\vec{V})$ be a sensor configuration. A pair of intersecting rays $\{\vec{a},\vec{b}\} \subseteq \vec{V}$ forms a 2-barrier if (i) one is red and the other is blue, (ii) one is red and the other is black and they intersect above st, or (iii) one is blue and the other is black and they intersect below st.*

Lemma 2 *Let $(s,t,\vec{V})$ be a sensor configuration and $\vec{V'} \subseteq \vec{V}$. The set $\vec{V'}$ forms a barrier for $(s,t,\vec{V})$ if and only if there are two rays $\vec{a},\vec{b} \in \vec{V'}$ such that $\{\vec{a},\vec{b}\}$ forms a 2-barrier of one of the types described in Proposition 1.*

Lemma 2 motivates the reformulation of the resilience problem as a graph problem (see Lemma 3 below.) We say that a graph $G = (V,E)$ is a *barrier graph* of $(s,t,\vec{V})$, denoted by $G(\vec{V})$, if V is the set of all endpoints in $\vec{V}$, and $\{a,b\} \in E$ iff the corresponding pair of rays $\{\vec{a},\vec{b}\}$ forms a 2-barrier (see Figure 2). It follows immediately from Lemma 2 that barrier graphs are tripartite.

Note that we use the same notation for a vertex in $G(\vec{V})$ and an endpoint in $\vec{V}$. When there is no ambiguity, a vertex of $G(\vec{V})$ is also referred to as an endpoint of a ray. We exploit the vertex-endpoint duality to view $G(\vec{V})$ as a vertex-coloured embedded graph. This allows us to say, for example, that any vertex that lies above st must be red or black, and any vertex that lies below st must be blue or black.

Lemma 3 *For any sensor configuration $(s,t,\vec{V})$, a vertex set V_c is a minimum size vertex cover of $G(\vec{V})$ if and only if the corresponding set of rays $\vec{V}_c$ is a resilience set of $(s,t,\vec{V})$.*

From Lemma 3, it is clear that we can find a resilience set efficiently if there is an efficient algorithm to compute a minimum size vertex cover of the graph $G(\vec{V})$. Unfortunately, the vertex cover problem on general tripartite graphs is NP-complete (there is a straightforward reduction from the vertex cover problem for cubic planar graphs, which is known to be NP-complete [6]). In fact, Clementi et al. [4] have shown that the minimum size vertex cover for tripartite graphs is not even approximable to within a factor of 34/33, unless P=NP.

Thus, we are motivated to look for additional structure in instances of the tripartite vertex cover problems that arise from barrier graphs. In some settings, for example if all rays are parallel to one of two different lines, the graph $G(\vec{V})$ is easily seen to be bipartite. It is well-known, by König's theorem [9], that, for bipartite graphs, constructing a minimum size vertex cover is equivalent to constructing a maximum size matching. Thus, we can use the Hopcroft-Karp algorithm [7] to find a minimum size vertex cover in such instances in $O(m\sqrt{n})$ time, where m is the number of edges in $G(\vec{V})$.

To exploit the structure inherent in less restricted instances, we first introduce some additional notation (see Figure 3). We denote by ℓ_{sv} the line passing through points s and v. Similarly ℓ_{tw} denotes the line passing through points t and w. A generic line through s (respectively, t) is denoted ℓ_{s-} (respectively, ℓ_{t-}). Similarly, a distinguished line through s (respectively, t) will be denoted ℓ_{s*} (respectively, ℓ_{t*}). With any line ℓ_{s-} (respectively, ℓ_{t-}) we associate the half-space, denoted ℓ^-_{s-} (respectively, ℓ^+_{t-}) consisting of all points to the left of ℓ_{s-}, or above ℓ_{s-} in case ℓ_{s-} is horizontal (respectively, all points to the right of ℓ_{t-}, or above ℓ_{t-} in case ℓ_{t-} is horizontal).

Armed with this notation, we can capture two additional properties of barrier graphs that can be exploited in the efficient construction of optimal vertex covers:

Lemma 4 *Let $G(\vec{V})$ be a barrier graph and suppose that V_c is any vertex cover of $G(\vec{V})$. Then there must exist lines ℓ_{s*} and ℓ_{t*}, through s and t respectively, such that V_c contains all of the red and blue vertices that lie in $\ell^-_{s*} \cup \ell^+_{t*}$.*

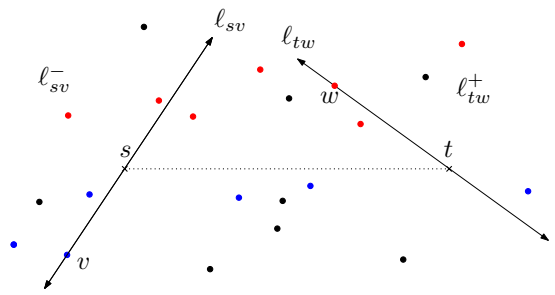


Figure 3: Lines (and associated half-spaces) through s and t.

Lemma 5 *Let ℓ_{s-} and ℓ_{t-} be arbitrary lines through s and t respectively, and let $\widetilde{RB}$ denote the set of red and blue vertices that lie in $\ell^-_{s-} \cup \ell^+_{t-}$. Then the subgraph of the barrier graph $G(\vec{V})$ that is induced by the vertices in $V \backslash \widetilde{RB}$, is bipartite.*

3 Algorithm

In this section, we present an algorithm for computing resilience sets. Its correctness follows immediately from Lemma 3.

Algorithm 1: RESILIENCE.

Input: Sensor configuration $(s,t,\vec{V})$.
Output: A resilience set for $(s,t,\vec{V})$.
1 build the vertex-coloured barrier graph $G(\vec{V})$ (described in Section 2)
2 **return** MIN-VERTEX-COVER$(G(\vec{V}))$

As we have already noted, a minimum size vertex cover of a bipartite graph can be found in polynomial time [7]. So the basic idea of our algorithm is to exploit this by forming a sequence of subsets $U_1, U_2, \ldots$ of V such that (i) for all i, $G|(V \backslash U_i)$, the subgraph of $G(\vec{V})$ induced on the vertex set $V \backslash U_i$, is bipartite, and (ii) for some i, the minimum size vertex cover of $G|(V \backslash U_i)$, together with the vertices in U_i, forms a minimum size vertex cover of $G(\vec{V})$.

We know, by Lemma 4, that for any minimum size vertex cover V_c of $G(\vec{V})$ there must exist lines ℓ_{s*} and ℓ_{t*}, through s and t respectively, such that V_c contains all of the vertices in $\widetilde{RB}$, the set of red and blue vertices that lie in $\ell^-_{s*} \cup \ell^+_{t*}$. Furthermore, the vertices of $V_c \backslash \widetilde{RB}$ must be a minimum size vertex cover of $G|(V \backslash \widetilde{RB})$; otherwise

V_c would not have minimum size. So our algorithm simply tries all possibilities for ℓ_{s*} and ℓ_{t*}, determines the associated set $\widetilde{RB}$, finds a minimum size vertex cover of $G|(V\backslash\widetilde{RB})$ (which, by Lemma 5, is bipartite), and chooses, among all of these possibilities, one that minimizes the size of this vertex cover together with the set $\widetilde{RB}$.

Algorithm 2: MIN-VERTEX-COVER of a barrier graph.

Input: A barrier graph $G(\vec{V})$.
Output: A minimum vertex cover of $G(\vec{V})$.

```
1  RB~ ← {red vertices in V}
2  VC_temp ←
   BIPARTITE-VERTEX-COVER(G|(V\RB~))
3  VC_best ← VC_temp ∪ RB~
4  for every red vertex v do
5      for every red vertex w do
6          RB~ ←
           {red and blue vertices in ℓ⁻_sv ∪ ℓ⁺_tw}
7          VC_temp ←
           BIPARTITE-VERTEX-COVER(G|(V\RB~))
8          if |VC_temp| + |RB~| < |VC_best| then
9              VC_best ← VC_temp ∪ RB~
10 return VC_best
```

As we have already noted, the correctness of Algorithm RESILIENCE follows immediately from Lemma 3. We now turn our attention to the correctness of our VERTEX COVER algorithm for barrier graphs.

Theorem 6 *The output of Algorithm 2 is a minimum size vertex cover of* $G(\vec{V})$.

It will be clear from the description of Algorithm 2 that the problem of constructing a minimum size vertex cover of a barrier graph with n vertices and m edges can be reduced to $O(n^2)$ minimum size vertex cover subproblems on induced subgraphs, each of which, by Lemma 5, is bipartite. As previously noted, König's theorem [9] states that, for bipartite graphs, constructing a minimum size vertex cover is equivalent to constructing a maximum size matching. Thus, we can use the Hopcroft-Karp algorithm to find a minimum size vertex cover in each subproblem in $O(m\sqrt{n})$ time [7], or $O(n^2m\sqrt{n})$ time in total. We note, however, that it is possible to implement Algorithm 2 to run in $O(n^2m)$ time, by ordering the successive subproblems in a way that they do not require independent solution; we leave the details to an expanded version of this paper.

References

[1] H. Alt, S. Cabello, P. Giannopoulos and C. Knauer, Minimum cell connection and separation in line segment arrangements, arXiv:1104.4618v2 [cs.CG], 2011.

[2] S. Bereg and D. Kirkpatrick, Approximating barrier resilience in wireless sensor networks, *Proceedings of the 5th International Workshop on Algorithmic Aspects of Wireless Sensor Networks, Lecture Notes in Computer Science*, Vol. 5804, pages 29–40, 2009.

[3] M. Cardei and J. Wu, Coverage in wireless sensor networks, In M. Ilyas and I. Mahgoub, editors, *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*, chapter 19, pages 432–446, CRC Press, 2005.

[4] A. Clementi, P. Crescenzi, G. Rossi, On the complexity of approximating colored-graph problems. *Proceedings of the 5th International Conference on Computing and Combinatorics, Lecture Notes in Computer Science*, Vol. 1627, pages 281–290, 1999.

[5] A. Chen, T. H. Lai and D. Xuan, Measuring and guaranteeing quality of barrier-coverage in wireless sensor networks, *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, pages 421–430, 2008.

[6] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.

[7] J. E. Hopcroft and R. M. Karp, An $n^{2.5}$ algorithm for maximum matchings in bipartite graphs, *SIAM Journal on Computing*, 2(4): 225–231, 1973.

[8] S. Kloder and S. Hutchinson, Barrier coverage for variable bounded-range line-ofsight guards, *Proceedings of IEEE International Conference on Robotics and Automation (ICRA'07)*, pages 391–396, 2007.

[9] D. König, Gráfok és mátrixok, *Matematikai és Fizikai Lapok*, 38:116–119, 1931.

[10] S. Kumar, T. H. Lai, and A. Arora, Barrier coverage with wireless sensors, *Wireless Networks*, 13(6): 817–834, 2007.

[11] K.-C. Tseng, Resilience of Wireless Sensor Networks, Master thesis, University of British Columbia, BC, Canada, 2011.

[12] K.-C. Tseng and D. Kirkpatrick, On Barrier Resilience of Sensor Networks, *Proceedings of the 7th International Workshop on Algorithmic Aspects of Wireless Sensor Networks, Lecture Notes in Computer Science*, Vol. 7111, pages 130-144, 2011.

Computing the stretch of an embedded graph

Sergio Cabello*[1], Markus Chimani†[2], and Petr Hliněný‡[3]

[1]Department of Mathematics, FMF, University of Ljubljana, Slovenia
[2]Theoretical Computer Science, Department of Maths/CS, Osnabrück University, Germany
[3]Faculty of Informatics, Masaryk University, Brno, Czech Republic

Abstract

Let G be a graph embedded in an orientable surface Σ, possibly with edge weights, and denote by $\text{len}(\gamma)$ the length (the number of edges or the sum of the edge weights) of a cycle γ in G. The stretch of a graph embedded on a surface is the minimum of $\text{len}(\alpha) \cdot \text{len}(\beta)$ over all pairs of cycles α and β that cross exactly once. We provide an algorithm to compute the stretch of an embedded graph in time $O(g^4 n \log n)$ with high probability, or in time $O(g^4 n \log^2 n)$ in the worst case, where g is the genus of the surface Σ and n is the number of vertices in G.

Introduction

Consider a graph G embedded on an orientable surface Σ of genus g. What can be said about the crossing number of G in the plane? Is it computable in polynomial time? If it is not, can we obtain a reasonable approximation in polynomial time? Unfortunately, Cabello and Mohar [3] show that the crossing number of such graphs is not computable in polynomial time, even when Σ is the torus. Djidjev and Vrt'o [4] show that the crossing number of G is upper bounded by $O(g\Delta n)$, where n is the number of vertices in G and Δ is the maximum degree of G. This is an improvement over the previous bound of $O(C^g \Delta n)$, for some constant C, by Böröczky, Pach and Tóth [1]. Under some mild assumptions about the density of the embedding of G, Hliněný and Chimani [5] give a $(3 \cdot 2^{3g+2}\Delta^2)$-approximation algorithm for the crossing number of G. This last work is the main motivation for our research.

Hliněný and Chimani [5] define the ***stretch*** of an embedded graph G as

$$\texttt{str}(G) \;=\; \min_{\alpha,\beta}\{\text{len}(\alpha) \cdot \text{len}(\beta)\},$$

where α, β ranges over all pairs of cycles in G that cross exactly once. Here, $\text{len}(\alpha)$ denotes the number of edges in α and a ***cycle*** is a closed walk in a graph without repeated vertices. A precise definition of what "crossing exactly once" means is given in Section 1.2. The stretch plays a fundamental role in their analysis of the algorithm. The concept of stretch can be generalized to the case of positive edge-weighted graphs in a natural way: take $\text{len}(\alpha)$ to be the sum of the edge-weights along the cycle α. Henceforth we will assume this more general definition of stretch.

It is worth noting that, if two cycles α and β are crossing once, then they must be both (surface-)non-separating. That is, cutting the surface along α or β does not disconnect the surface. This is so because any cycle crosses a (surface-)separating cycle an even number of times. Thus, when computing the stretch, we can restrict our attention to pairs (α, β) of non-separating cycles.

In this paper we provide an algorithm to compute the stretch of an embedded graph in time $O(g^4 n \log n)$ with high probability, or in time $O(g^4 n \log^2 n)$ in the worst case, where g is the genus of the surface Σ and n is the number of vertices in G.

Overview of the approach. Let us provide an informal overview of the main ideas. We show the following recursive property of the stretch: it is defined either by the shortest non-separating cycle α^* and one other cycle crossing α^* exactly once, or by the stretch of surface obtained by cutting along α^* and pasting disks. We do not prove this claim directly, but using a detour through another concept: odd-stretch.

The definition of odd-stretch resembles the definition of stretch, but we allow closed walks α and β, instead of just cycles, and allow an odd number of crossings, instead of exactly one crossing. It turns out that the stretch and odd-stretch of a graph is the same. However, working with the odd-stretch is easier because we only need to take care of the parity of crossings and, when constructing new closed walks

*Supported by the Slovenian Research Agency, program P1-0297, project J1-4106, and within the EUROCORES Programme EUROGIGA (project GReGAS) of the European Science Foundation.

†This research was conducted while being funded by a Carl-Zeiss-Foundation juniorprofessorship, and partially supported by EUROCORES Programme EUROGIGA (project GraDR) of the European Science Foundation.

‡Supported by the Czech Science Foundation, EUROCORES grant GIG/11/E023 (project GraDR).

via exchange arguments, we do not need to take care to construct cycles. Finally, we prove the aforementioned recursive property for the odd-stretch factor.

The eventual algorithm, given in Figure 1 is very simple. However, there is a fine point we have to take care of to obtain a polynomial-time algorithm. Repeatedly cutting along shortest non-separating cycles and pasting disks may give rise to an exponential growth in the size of the graphs: at each cut we make copies of the vertices along the cycle and thus the number of vertices may nearly double at each iteration. However, if at some iteration we get a shortest non-separating cycle with more vertices than the original graph, we can finish the recursive search. In this way we avoid the potentially exponential growth in the size of the graphs.

1 Odd-stretch

In this section we introduce the concept of odd-stretch, which is a generalization of stretch. We first discuss crossings for curves in general position and then crossings for closed walks in a graph. Finally, we define the odd-stretch, discuss some of its properties and, eventually, show that the odd-stretch is the same as the stretch.

1.1 Crossings of curves in general position

Two curves C and C' on a surface Σ are in ***general position*** if they have a finite number of intersections and, at each intersection, they cross transversally. For two curves $C = C(t)$ and $C' = C'(t)$ in general position, the ***set of crossings*** is

$$X(C, C') \;=\; \{x \in \Sigma \mid \exists t, t' \text{ s.t. } x = C(t) = C'(t')\}.$$

Two curves C and C' in general position ***cross*** k times if and only if $k = |X(C, C')|$. We will use the following (intuitive) fact: the number of crossings between two closed curves, modulo 2, is invariant under small perturbations of any of the two closed curves.

1.2 Crossings of closed walks

Two closed walks α and β in G cross k times if and only if: there are arbitrarily small perturbations of α and β to general position that cross k times, and any small enough perturbation of α and β has at least k intersecting points. Moreover, we can always assume that the crossings of the perturbations occur in a neighborhood of the vertices. We denote by $\texttt{cr}(\alpha, \beta)$ the number of crossings between α and β.

For any closed walk α, the set of closed walks in G that cross α an odd number of times satisfies the so-called 3-path condition. The next lemma states this in an equivalent way for easier use later on. This property was already noted in [5].

Lemma 1 *Let α be a closed walk and let γ be a closed walk crossing α an odd number of times. Let x and y be two vertices on γ and let π be some walk from x to y. Let γ' be the closed walk defined by concatenating $\gamma[y \to x]$ and π. Let γ'' be the closed walk defined by concatenating $\gamma[x \to y]$ and the reverse of π. Then either γ' or γ'' cross α an odd number of times.*

1.3 Odd-stretch of an embedded graph

The odd-stretch of an embedded graph G is

$$\texttt{oddstr}(G) \;=\; \min_{\alpha,\beta}\{\text{len}(\alpha) \cdot \text{len}(\beta)\},$$

where α, β ranges over all pairs of closed walks in G that cross an odd number of times. We next remark the two main differences with the previous stretch: α and β iterate over closed walks, instead of cycles, and the curves can cross an odd number of times, instead of exactly once. A priori, the stretch and the odd-stretch of an embedded graph are different. A posteriori we can see that they are the same; this was already noted in [5].

Lemma 2 *The odd-stretch of G and the stretch of G are the same.*

2 Algorithm for computing the stretch factor

We will use the following two properties:

Lemma 3 ([2]) *Let G be a graph with m vertices embedded in a surface of genus g.*

- *We can compute a shortest non-separating cycle in time $O(g^2 m \log m)$ with high probability, or in time $O(g^2 m \log^2 m)$ in the worst case.*
- *For any given non-separating cycle α, we can compute a shortest cycle of G that crosses α exactly once in time $O(gm \log m)$ with high probability, or in time $O(gm \log^2 m)$ in the worst case.*

Lemma 4 *Let α be a shortest non-separating cycle in G. For any two vertices x and y on α, α contains a shortest path from x to y or from y and x.*

The algorithm for computing the stretch of an embedded graph is given in Figure 1. We first discuss its time complexity and then its correctness.

Lemma 5 *Algorithm* ComputeStretch *has time complexity $O(g^4 n \log n)$ with high probability, or $O(g^4 n \log^2 n)$ in the worst case, where n is the number of vertices in G.*

Algorithm ComputeStretch
Input: graph G embedded in surface Σ
Output: stretch of G
1. $i \leftarrow 1$;
2. $(G_1, \Sigma_1) \leftarrow (G, \Sigma)$;
3. str $\leftarrow \infty$;
4. **while** Σ_i not the sphere and $|V(G_i)| \leq g \cdot |V(G)|$ **do**
5. $\quad \alpha_i \leftarrow$ shortest non-separating cycle in G_i;
6. $\quad \beta_i \leftarrow$ shortest cycle crossing α_i exactly once;
7. $\quad$ str $\leftarrow \min\{\text{str}, \text{len}(\alpha_i) \cdot \text{len}(\beta_i)\}$;
8. $\quad (G_{i+1}, \Sigma_{i+1}) \leftarrow$ cut (G_i, Σ_i) along α_i and attach disks to the boundaries;
9. $\quad i \leftarrow i + 1$;
10. **return** str

Figure 1: Algorithm ComputeStretch to compute the stretch of an embedded graph.

Proof. Because of Lemma 3, in each iteration of the while loop we need $O(g_i^2 n_i \log n_i)$ time whp, or $O(g_i^2 n_i \log^2 n_i)$ in the worst case, where n_i is the number of vertices in G_i and g_i is the genus of Σ_i. Since $n_i = O(gn)$ because of the condition for iterating the while loop and $g_i \leq g$, each iteration of the while loop takes $O(g^2(gn)\log(gn)) = O(g^3 n \log n)$ time whp, or $O(g^3 n \log^2 n)$ time in the worst case. There are at most g iterations of the while loop. □

Lemma 6 *Let α be a shortest non-separating cycle and let β be a shortest cycle crossing α exactly once. Let G' be the embedded graph obtained from G by cutting along α and attaching a disk to the boundaries. The stretch of G is the minimum between* $\text{len}(\alpha) \cdot \text{len}(\beta)$ *and the stretch of G'.*

Proof. Let Σ be the surface where G is embedded and let Σ' be the surface where G' is embedded. Since any two closed curves of G' that cross an odd number of times in Σ' also cross an odd number of times in Σ, it is clear that

$$\texttt{str}(G) \;\leq\; \min\{\texttt{str}(G'), \text{len}(\alpha) \cdot \text{len}(\beta)\}.$$

Thus, we have to argue the other inequality. If (α, β) define the stretch of G, then the other inequality is obvious.

Let us consider next the case where (α, β) do not define the stretch of G; it holds that $\texttt{str}(G) < \text{len}(\alpha) \cdot \text{len}(\beta)$. Let (γ^*, σ^*) be the pair of cycles that define the stretch of G. If there are several such pairs, we choose one such that $\texttt{cr}(\gamma^*, \alpha) + \texttt{cr}(\sigma^*, \alpha)$ is minimum. We distinguish 3 cases depending on the values of $\texttt{cr}(\gamma^*, \alpha)$ and $\texttt{cr}(\sigma^*, \alpha)$:

Case $\texttt{cr}(\gamma^*, \alpha) = \texttt{cr}(\sigma^*, \alpha) = 0$. In this case, γ^* and σ^* keep crossing once in Σ', and thus $\texttt{str}(G) = \texttt{str}(G')$.

Case $\texttt{cr}(\gamma^*, \alpha) = 1$ or $\texttt{cr}(\sigma^*, \alpha) = 1$. This case cannot actually happen. Let us assume that $\texttt{cr}(\gamma^*, \alpha) = 1$; the other case is symmetric. Since γ^* crosses α once and β is a shortest cycle crossing α once, we have $\text{len}(\beta) \leq \text{len}(\gamma^*)$. Using that α is a shortest non-separating cycle we have

$$\texttt{str}(G) \;=\; \text{len}(\gamma^*) \cdot \text{len}(\sigma^*) \;\geq\; \text{len}(\beta) \cdot \text{len}(\alpha).$$

This implies that (α, β) actually define the stretch of G.

Case $\texttt{cr}(\gamma^*, \alpha) \geq 2$ or $\texttt{cr}(\sigma^*, \alpha) \geq 2$. This case cannot actually happen. Let us assume that $\texttt{cr}(\gamma^*, \alpha) \geq 2$; the other case is symmetric. Let x and y be two crossings of γ^* and α that are consecutive along α. Because of Lemma 4, α contains a shortest path between x and y. Let π denote this shortest path. We can use π and γ^* to construct two cycles γ' and γ'' that are shorter and cross α fewer times than γ^*. Because of Lemma 1, some $\tilde{\gamma} \in \{\gamma', \gamma''\}$ crosses σ^* an odd number of times. The pair $(\tilde{\gamma}, \sigma^*)$ contradicts the choice of (γ^*, σ^*).

This finishes all cases. (Note that the second and third cases are not mutually exclusive.) □

Lemma 6 shows correctness of the algorithm if the condition $|V(G_i)| \leq g \cdot |V(G)|$ is true for each $i = 1, \ldots, g$. We next argue why we can finish the search if at some iteration $|V(G_i)| > g \cdot |V(G)|$.

Lemma 7 *If, for some i, α_i has more than $|V(G)|$ vertices, then for any $\ell \geq i$*

$$\texttt{str}(G) \;=\; \min\{\text{len}(\alpha_j) \cdot \text{len}(\beta_j) \mid j = 1, \ldots, \ell - 1\}.$$

Proof. Assume, for the sake of this proof, that in the algorithm ComputeStretch we drop testing the condition $|V(G_i)| \leq g \cdot |V(G)|$. The algorithm then makes exactly g iterations and computes cycles α_j, β_j for each $j = 1, \ldots, g$. Because of Lemma 6 it holds

$$\texttt{str}(G) \;=\; \min\{\text{len}(\alpha_j) \cdot \text{len}(\beta_j) \mid j = 1, \ldots, g\}.$$

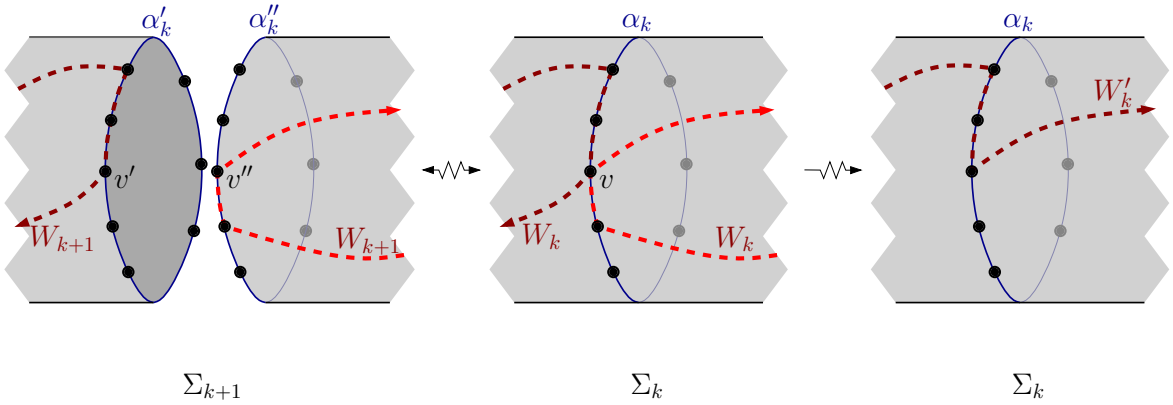


Figure 2: Figure for the proof of Lemma 7.

Assume that, at some iteration, the shortest non-separating cycle α_i in G_i, has more than $|V(G)|$ vertices. For each $k < i$, the cycle α_i corresponds to a closed walk W_k in the graph G_k. Moreover, the walk W_k does not cross the cycle α_k, for each $k < i$. Since α_i has more than $|V(G)|$ vertices, W_1 repeats some vertex of $G_1 = G$. This means that W_1 is not a cycle.

Let k be the maximum index, $1 \leq k < i$ such that W_k is *not a cycle* in G_k; thus W_{k+1} is a cycle in G_{k+1}. Since W_1 is not a cycle and W_i is a cycle, the index k is well defined. See Figure 2, left and center. Let v be a vertex of G_k that is repeated in W_k. Cutting G_k through α_k produces two copies α'_k and α''_k of α_k. Let v' and v'' be the corresponding copies of v. We can form a closed walk W'_k in G_k by taking the subwalk of W_k from the first appearance of v until the second. This closed walk W'_k crosses α_k once. Therefore $\mathrm{len}(\beta_k) \leq \mathrm{len}(W'_k) < \mathrm{len}(W_k) = \mathrm{len}(\alpha_i) \leq \mathrm{len}(\alpha_j)$ for each $j \geq i$. We conclude that, for each $j \geq i$,

$$\mathrm{len}(\alpha_j)\cdot\mathrm{len}(\beta_j) > \mathrm{len}(\beta_k)\cdot\mathrm{len}(\alpha_j) \;\geq\; \mathrm{len}(\beta_k)\cdot\mathrm{len}(\alpha_k).$$

Since $k < i \leq \ell$ we conclude that

$$\begin{aligned} \mathtt{str}(G) &= \min\{\mathrm{len}(\alpha_j)\cdot\mathrm{len}(\beta_j) \mid j = 1,\ldots,g\} \\ &= \min\{\mathrm{len}(\alpha_j)\cdot\mathrm{len}(\beta_j) \mid j = 1,\ldots,\ell-1\}. \end{aligned}$$

□

Theorem 8 *Let G be a graph with n vertices embedded in surface of genus g. We can compute the stretch of G in time $O(g^4 n \log n)$ with high probability, or in time $O(g^4 n \log^2 n)$ in the worst case.*

Proof. The time bound follows from Lemma 5. For the correctness, note that the while loop has the following invariant: at the start of iteration i, the variable *str* stores the value

$$\min(\{\mathrm{len}(\alpha_j)\cdot\mathrm{len}(\beta_j) \mid j = 1,\ldots,i-1\} \cup \{\infty\}).$$

If $|V(G_i)| \leq g \cdot |V(G)|$ for each iteration, then the algorithm finishes with $i = g+1$, the surface Σ_{g+1} is a sphere, and correctness follows from Lemma 6. If at some iteration ℓ we have $|V(G_\ell)| > g \cdot |V(G)|$, then there was some index $i < \ell$ such that the cycle α_i had more than $|V(G)|$ vertices. Correctness then follows from Lemma 7. □

Acknowledgments. We are grateful to Daniel Štefankovič for some early discussions.

References

[1] K. J. Börözky, J. Pach, and G. Tóth. Planar crossing numbers of graphs embeddable in another surface. *Int. J. Found. Comput. Sci.*, 17(5):1005–1016, 2006.

[2] S. Cabello, E. W. Chambers, and J. Erickson. Multiple-source shortest paths in embedded graphs, 2013. Accepted to SIAM J. Computing. Preprint available at `http://arxiv.org/abs/1202.0314`. Preliminary version at SODA 2007.

[3] S. Cabello and B. Mohar. Adding one edge to planar graphs makes crossing number hard. In *Proc. SoCG 2010*, pages 68–76, 2010. See `http://arxiv.org/abs/1203.5944` for the full version.

[4] H. Djidjev and I. Vrt'o. Planar crossing numbers of graphs of bounded genus. *Discrete & Computational Geometry*, 48(2):393–415, 2012.

[5] P. Hliněný and M. Chimani. Approximating the crossing number of graphs embeddable in any orientable surface. In *Proc. SODA 2010*, pages 918–927, 2010.

An algorithm that constructs irreducible triangulations of once-punctured surfaces

M. J. Chávez*[1], S. Lawrencenko†[2], J. R. Portillo‡[1], and M. T. Villar§[1]

[1]Universidad de Sevilla, Spain
[2]Russian State University of Tourism and Service, Lyubertsy, Moscow Region, Russia

Abstract

A triangulation of a surface is irreducible if there is no edge whose contraction produces another triangulation of the surface. In this work we propose an algorithm that constructs the set of irreducible triangulations of any surface with precisely one boundary component.

Introduction and terminology

We restrict our attention to the following objects:

- $S = S_g$ or N_k is the closed orientable surface S_g of genus g or closed nonorientable surface of nonorientable genus k. In particular, S_0 and S_1 are the sphere and torus, N_1 and N_2 are the projective plane and the Klein bottle (respectively).
- $S - D$ is S minus an open disk D (the hole). This compact surface is called the *once-punctured* surface. We assume that the boundary ∂S of S, which is equal to ∂D, is homeomorphic to a circle.

We use the notation Σ whenever we assume the general case - that is, $\Sigma \in \{S, S - D\}$.

If a (finite, undirected, simple) graph G is 2-cell embedded in Σ, the components of $\Sigma - G$ are called *faces*. A triangulation of Σ with graph G is a 2-cell embedding $T : G \to \Sigma$ in which each face is bounded by a 3-cycle (that is, a cycle of length 3) of G and any two faces are either disjoint, share a single vertex, or share a single edge. We denote by $V = V(T)$, $E = E(T)$, and $F = F(T)$ the sets of the vertices, the edges, and the faces of T, respectively. Equivalently, the triangulation T of a surface can be defined as a hypergraph of rank 3 or a 3-graph, with the vertex set $V(T)$ and the collection $F(T)$ of triplets of $V(T)$ (called 3-edges, or triangles, of T) (see [4]).

By $G(T)$ we denote the graph $(V(T), E(T))$ of triangulation T. Two triangulations T_1 and T_2 are called *isomorphic*, denoted $T_1 \cong T_2$, if there is a bijection $\alpha : V(T_1) \to V(T_2)$ such that $uvw \in F(T_1)$ if and only if $\alpha(u)\alpha(v)\alpha(w) \in F(T_2)$. Throughout this paper we distinguish triangulations only up to isomorphism.

In the case $\Sigma = S - D$ let ∂T, which is equal to ∂D, denote the boundary cycle of T. The vertices and edges of ∂T are called *boundary vertices* and *boundary edges* of T, respectively.

A triangulation of a punctured surface is *irreducible* (term which is more accurately defined in Section 1) if no edge can be shrunk without producing multiple edges or changing the topological type of the surface. The irreducible triangulations of Σ form a basis for the family of all triangulations of Σ, in the sense that any triangulation of Σ can be obtained from a member of the basis by applying the *splitting* operation (introduced in Section 1) a finite number of times. To have such a basis in hand can be very useful in practical application of triangulation generating; the papers [6] and [19] are worth mentioning.

It is known that for every surface Σ the basis of irreducible triangulations is finite (the case of closed surfaces is proved in [2], [13], [12], and [7] and the case of surfaces with boundary is proved in [3]). At present such bases are known only for seven closed surfaces and two once-punctured surfaces: the sphere ([14]), projective plane ([1]), torus ([8]), Klein bottle ([9] and [15]), S_2, N_3, and N_4 ([16, 17]), the disk and Möbius band ([5]).

In this paper we present an algorithm which is designed as an application of some recent advances on the study of irreducible triangulations of once-punctured surface collected in [5]. Specifically, Lemmas 1-3 (in Section 1) are the main supporting theoretical results for the mentioned algorithm. As a particular example, all the non-isomorphic combinatorial types (293 in number) of triangulations of the once-punctured torus are determined.

1 Previous results

Let T be a triangulation of Σ. An unordered pair of distinct adjacent edges vu and vw of T is called

*Email: mjchavez@us.es. Research supported by PAI FQM-189
†Email: lawrencenko@hotmail.com.
‡Email: josera@us.es. Research supported by PAI FQM-164
§Email: villar@us.es. Research supported by PAI FQM-164

a *corner* of T at vertex v, denoted by $\langle u,v,w\rangle$ $(=\langle w,v,u\rangle)$. The *splitting* of a corner $\langle u,v,w\rangle$, denoted by $\mathrm{sp}\langle u,v,w\rangle$, is the operation which consists of cutting T open along the edges vu and vw and then closing the resulting hole with two new triangular faces, $v'v''u$ and $v'v''w$, where v' and v'' denote the two images of vertex v appearing as a result of cutting. Under this operation, vertex v is extended to the edge $v'v''$ and the two faces having this edge in common are inserted into the triangulation; therefore the order increases by one and the number of edges increases by three.

If a corner $\langle u,v,w\rangle$ is composed of two edges vu and vw neighboring around vertex v, $\mathrm{sp}\langle u,v,w\rangle$ is equivalent to the stellar subdivision of the face uvw.

Especially in the case $\{\Sigma = S-D, uv \in E(T), \text{and } v \in V(T)\}$, the operation $\mathrm{sp}\langle u,v]$ of *splitting a truncated corner* $\langle u,v]$ produces a single triangular face $uv'v''$, where $v'v'' \in E(\partial(\mathrm{sp}\langle u,v](T)))$.

Under the inverse operation, *shrinking* the edge $v'v''$, denoted by $\mathrm{sh}\rangle v'v''\langle$, this edge collapses to a single vertex v, the faces $v'v''u$ and $v'v''w$ collapse to the edges vu and vw, respectively. Therefore $\mathrm{sp}\langle u,v,w\rangle \circ sh\rangle v'v''\langle(T) = T$. It should be noticed that in the case $\{\Sigma = S-D, v'v'' \in E(\partial T)\}$, there is only one face incident with $v'v''$, and only that single face collapses to an edge under $\mathrm{sh}\rangle v'v''\langle$. Clearly, the operation of splitting doesn't change the topological type of Σ if $\Sigma \in \{S, S-D\}$. We demand that the shrinking operation must preserve the topological type of Σ as well; moreover, multiple edges must not be created in a triangulation. A 3-cycle of T is called *nonfacial* if it doesn't bound a face of T. In the case in which an edge $e \in E(T)$ occurs in some nonfacial 3-cycle, if we still insist on shrinking e, multiple edges would be produced, which would expel $\mathrm{sh}\rangle e\langle(T)$ from the class of triangulations. An edge e is called *shrinkable* or a *cable* if $\mathrm{sh}\rangle e\langle(T)$ is still a triangulation of Σ; otherwise the edge is called *unshrinkable* or a *rod*. The subgraph of $G(T)$ made up of all cables is called the *cable-subgraph of* $G(T)$.

The impediments to edge shrinkability in a triangulation T of a punctured surface $S-D$ are identified in [2, 3, 1, 8]; an edge $e \in E(T)$ is a rod if and only if e satisfies one of the following conditions:

(1) e is in a nonfacial 3-cycle of $G(T)$. In particular, e is a boundary edge in the case in which the boundary cycle is a 3-cycle.

(2) e is a chord of D -that is, the end vertices of e are in $V(\partial D)$ but $e \notin E(\partial D)$.

From now on, we assume that $S \neq S_0$ and make an agreement that by "non-facial 3-cycle" we mean a non-null-homotopic 3-cycle whenever we refer to conditions (1) and (2). Therefore, an edge e is a rod provided e occurs in some non-null-homotopic 3-cycle, and e is a cable otherwise. Especially, the boundary edges of stellar subdivided faces are now regarded as cables unless they occur in some non-null-homotopic 3-cycles. The convenience of this agreement is that once a rod turns into a cable in the course of any splitting sequence, it always remains a cable under forthcoming splittings.

A triangulation is said to be *irreducible* if it is free of cables or equivalently, each edge is a rod. For instance, a single triangle is the only irreducible triangulation of the disk $S_0 - D$.

Let T be an irreducible triangulation of a punctured surface $S-D$ where $S \neq S_0$. Let us close the hole in T by restoring the disk D add a vertex, p, in D, and join p to the vertices in ∂D. We thus obtain a triangulation, T^*, of the closed surface S. In this setting we call D the *patch*, call p the *central vertex of the patch*, and say that T is obtained from the corresponding triangulation T^* of S by the *patch removal.*

Notice that T^* may be an irreducible triangulation of S but not necessarily. Using the assumption that T is irreducible and the fact that each cable of T^* fails to satisfy condition (1) (in the strong non-null-homotopic sense), it can be easily seen that in the case T^* is not irreducible, all cables of T^* have to be entirely in $D \cup \partial D$ and, moreover, there is no cable which is entirely in ∂D whenever the length of the boundary cycle ∂D is greater than or equal to 4. In particular, we observe that each chord of D (if any) is a rod in T because it meets condition (2), and is also a rod in T because it meets condition (1). We now come to a lemma which is to be stated shortly after some necessary definitions.

A vertex of a triangulation R of S is called a *pylonic vertex* if that vertex is incident with all cables of R. A triangulation which has at least one cable and at least one pylonic vertex is called a *pylonic triangulation.*

A triangulation may have a unique cable and therefore two pylonic vertices. However, if the number of cables in a pylonic triangulation R is at least 2, R has exactly one pylonic vertex.

Lemma 1 *Suppose an irreducible triangulation T of a punctured surface $S-D$ $(S \neq S_0)$ is obtained from the corresponding triangulation T^* of S by the patch removal. If T^* has at least two cables, then either the central vertex p of the patch is the only pylonic vertex of T^*, or else the length of ∂D is equal to 3.*

Let $\Xi_n = \Xi_n(S)$ denote the set of triangulations of a fixed closed surface S that can be obtained from an irreducible triangulation of S by a sequence of exactly n repeated splittings.

In the following result, by the "removal of a vertex v" we mean the removal of v together with the interiors of the edges and faces incident with v and by the "removal of a face" we mean the removal of the interior of that face.

Lemma 2 *Each irreducible triangulation T of $S - D$ ($S \neq S_0$) can be obtained either*

(i) by removing a vertex from a triangulation in $\Xi_0 = \Xi_0(S)$, or

(ii) by removing a pylonic vertex from a triangulation in $\Xi_1 \cup \Xi_2 \cup \cdots \cup \Xi_K$, where the constant K is provided by [3] (whenever a pylonic triangulation occurs), or

(iii) by removing either of the two faces containing a cable in their boundary 3-cycles provided that cable is unique in a triangulation in Ξ_1 (whenever such a situation occurs), or

(iv) by removing the face containing two, or three, cables in its boundary 3-cycle provided those two, or three, cables collectively form the whole cable-subgraph in a triangulation in $\Xi_1 \cup \Xi_2$ (if such a situation occurs).

Lemma 3 *If a triangulation of S has at least two cables but has no pylonic vertex, then no pylonic vertex can be created under further splitting of the triangulation.*

2 Sketch of the algorithm

In this section triangulations are considered to be hypergraphs of rank 3 or 3-graphs. Let T be a 3-graph with $V = V(T)$ and $F = F(T)$ the sets of the vertices and the triangles of T. This 3-graph can be uniquely represented as a bipartite graph $B_T = (V(B_T), E(B_T))$ in the following way: $V(B_T) = V(T) \cup F(T)$, $uv \in E(B_T)$ if and only if the vertex u lies in the triangle v in T.

The algorithm *input* is the set $\mathcal{I}$ of irreducible triangulations of a closed surface $S \neq S_0$. The *output* of the algorithm is the set of all non-isomorphic combinatorial types of irreducible triangulations of the once-punctured surface $S - D$.

The first step is the generation of the set $\Xi_1 \cup \Xi_2$ from the set $\mathcal{I}$. Next, every 3-graph $T \in \Xi_1 \cup \Xi_2$ is then represented by their corresponding bipartite graph B_T. This has been implemented with the computational package *Mathematica* ([18]).

The second step consists of discarding all duplicate bipartite graphs and then, all duplicate triangulations in $\Xi_1 \cup \Xi_2$ will be discarded. That is, the obtention of all non-isomorphic triangulations, denoted $\widetilde{\Xi_1} \cup \widetilde{\Xi_2}$ respectively, which is implemented with the computing packages *Nauty* (and *gtools*, [10], [11]).

Next, all pylonic vertices in $\widetilde{\Xi_1} \cup \widetilde{\Xi_2}$ are detected and operations *(i)–(iv)* described in Lemma 2 are applied to obtain irreducible triangulations (using *Mathematica*).

If Ξ_2 has no pylonic triangulation, immediately proceed to the final step: Discard all duplicate triangulations by using *Nauty* and *gtools*. Otherwise generate Ξ_3 and apply the preceding steps to Ξ_3. Repeat this procedure to process $\Xi_4, \Xi_5, \ldots$ until no pylonic triangulation is left in the current Ξ_n; then the process terminates and a required output is produced.

The validity of this procedure is justified by Lemmas 1 - 3 along with the results of [3]. In particular, the finiteness of the procedure is deduced from the upper bound [3] on the number of vertices in an irreducible triangulation of $S - D$. In particular, that upper bound implies (along with Lemma 3) that Ξ_n does not have a pylonic triangulation for any $n \geq K + 1$, where $K = 945$ for S_1 and $K = 376$ for N_1. In reality K is much less than these values. By computer verification (and also by hand) we have checked that in fact $K = 1$ for S_1, and $K = 2$ for N_1.

Let us now mention two examples.

Firstly, this algorithm has been implemented for the set of two irreducible triangulations of N_1 ([1]). The algorithm gives a set of 6 irreducible triangulations of the Möbius band, $N_1 - D$, which is precisely the same as that obtained by some of the authors of this work in [5], although by using no computational package.

Secondly, we introduce the details of the torus case, S_1.

Example: the once-punctured torus

Input: The set of 21 irreducible triangulations of S_1 (as they are labelled in [8]).

- $\Xi_1(S_1)$ has 433 non-isomorphic triangulations: 232 of them have no pylonic vertex, 193 have an only pylonic vertex and 8 have two pylonic vertices.
- $\Xi_2(S_1)$ has 11612 non-isomorphic triangulations, none of them is a pylonic triangulation.
- Operations described in Lemma 2 provide:
  - (i) 184 triangulations; only 80 of them are non-isomorphic.
  - (ii) 209 triangulations; only 203 of them are non-isomorphic.
  - (iii) 16 triangulations, 10 of them are non-isomorphic.
  - (iv) 0 triangulations.

Output: 293 non-isomorphic combinatorial types of irreducible triangulations of the once-punctured torus $S_1 - D$.

3 Final conclusion

It is clear that this algorithm can be implemented for any closed surface whenever its basis of irreducible

triangulations is known. In a future contribution we hope to present the set of irreducible triangulations of the once-punctured Klein bottle, $N_2 - D$.

References

[1] D. Barnette (D. W. Barnette). Generating the triangulations of the projective plane. *J. Comb. Theory, Ser. B*, **33**, (1982), 222–230.

[2] D. W. Barnette, A. L. Edelson. All 2-manifolds have finitely many minimal triangulations. *Isr. J. Math.*, **67**, (1989), No. 1:123–128.

[3] A. Boulch, É. Colin de Verdière, A. Nakamoto. Irreducible triangulations of surfaces with boundary. *Graphs and Combinatorics* DOI 10.1007/s00373-012-1244-1, 2012.

[4] J. Bracho, R. Strausz, Nonisomorphic complete triangulations of a surface. *Discrete Math.* **232**, (2001), 11–18.

[5] M. J. Chávez, S. A. Lawrencenko, A. Quintero, M. T. Villar. Irreducible triangulations of the Möbius band. Preprint 2013.

[6] L. Giomi, M. J. Bowick. Elastic theory of defects in toroidal crystals. *Eur. Phys. J. E.*, **27**, (2008), 275–296.

[7] G. Joret, D. R. Wood. Irreducible triangulations are small. *J. Comb. Theory, Ser. B*, **100**, (2010), No. 5:446–455.

[8] S. A. Lavrenchenko (S. Lawrencenko). Irreducible triangulations of the torus. *J. Sov. Math.*, **51**, No. 5 (1990), 2537–2543; translation from Ukr. Geom. Sb. **30**, (1987), 52–62.

[9] S. Lawrencenko, S. Negami. Irreducible triangulations of the Klein bottle. *J. Comb. Theory, Ser. B*, **70**, No. 2 (1997), 265–291.

[10] B. McKay, Practical Graph Isomorphism. *Congressus Numerantium*, **30**, (1981), 45–87.

[11] B. McKay, *nauty User's Guide (Version 2.4)* http://pallini.di.uniroma1.it/

[12] A. Nakamoto, K. Ota. Note on irreducible triangulations of surfaces. *J. Graph Theory*, **20**, (1995), No. 2:227–233.

[13] S. Negami. Diagonal flips in pseudo-triangulations on closed surfaces. *Discrete Math.*, **240**, (2001), No. 1–3:187–196. 2001.

[14] E. Steinitz, H. Rademacher, *Vorlesungen über die Theorie der Polyeder unter Einschluss der Elemente der Topologie.* Berlin: Springer, 1976. [Reprint of the original 1934 edition.]

[15] T. Sulanke. Note on the irreducible triangulations of the Klein bottle. *J. Comb. Theory, Ser. B*, **96**, No. 6, (2006), 964–972.

[16] T. Sulanke. Generating irreducible triangulations of surfaces, preprint, 2006, `arXiv:math/0606687v1 [math.CO]`.

[17] T. Sulanke. Irreducible triangulations of low genus surfaces, preprint, 2006, `arXiv:math/0606690v1 [math.CO]`.

[18] Wolfram Research, Inc. Mathematica Version 8.0 (Wolfram Research, Inc.) Champaign, Illinois 2010.

[19] Y. Zhongwey, J. Shouwei. STL file generation from digitised data points based on triangulation of 3D parametric surfaces.*Int. J. Adv. Manuf. Technol.* **23**, (2004), 882–888.

On the enumeration of permutominoes

Ana Paula Tomás*

DCC & CMUP, Faculdade de Ciências
Universidade do Porto, Portugal

Abstract

Although the exact counting and enumeration of polyominoes remain challenging open problems, several positive results were achieved for special classes of polyominoes. We give an algorithm for direct enumeration of *permutominoes* [12] by size, or, equivalently, for the enumeration of *grid orthogonal polygons* [19]. We show how the construction technique allows us to derive a simple characterization of the class of convex permutominoes, which has been extensively investigated recently [4]. The approach extends to some of its subclasses, namely to the row convex and the directed convex permutominoes.

Introduction

The generation of geometric objects has applications to the experimental evaluation and testing of geometric algorithms. No polynomial time algorithm is known for generating polygons uniformly on a given set of vertices. Some generators employ heuristics [1, 6] or restrict to certain classes of polygons, e.g., monotone, convex or star-shaped polygons [18, 20]. Numerous related problems have also been extensively investigated, as the exact counting or enumeration of polyominoes [9]. These remain challenging open problems in computational geometry and enumerative combinatorics. A *polyomino* is an edge-connected set of unit squares on a regular square lattice (grid). Polyominoes are defined up to translations. In this paper, we give an algorithm for the enumeration of *permutominoes* by *size*, or, equivalently, for the enumeration of *grid orthogonal polygons* (see Fig. 1). Polyominoes are usually enumerated by area (i.e., number of cells). The direct enumeration of polyominoes is a computational problem of exponential complexity. An overview of the main developments concerning direct and indirect approaches is given in [3]. Jensen's transfer-matrix algorithm [13] – an indirect method – is currently the most powerful algorithm for counting fixed polyominoes. Exact counts are known for polyominoes that have up to 56 cells [3, 14]. As far as we can see, Jensen's algorithm cannot be easily adapted for counting permutominoes.

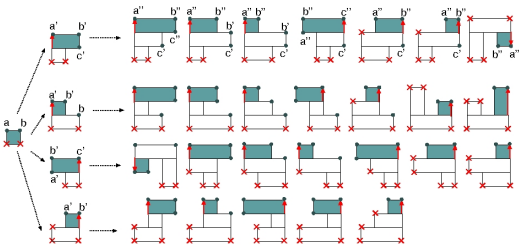


Figure 1: Permutominoes of size 1, 2, and 3 (i.e., with 4, 6, and 8 vertices) and their horizontal partitions.

1 Background

A polygon is called orthogonal if its edges meet at right angles. If r is the number of reflex vertices of an n-vertex orthogonal polygon, then $n = 2r + 4$ (e.g. [16]). Grid orthogonal polygons (*grid ogons*) were introduced in [19] as a relevant class for generation. A *grid ogon* is an orthogonal polygon without collinear edges, embedded in a regular square grid, and having exactly one edge in each line of its minimal bounding square. They were addressed more recently under the name of *permutominoes* [4, 8, 12], because they correspond to polyominoes that are determined by a suitable pair of permutations having the same size. All polyominoes we will consider are simply-connected and, similarly, all polygons are simple and without holes. A permutomino that is given by permutations of $\{1, 2, \ldots, r+2\}$, for $r \geq 0$, is said to have size $r + 1$. The size is the width of its minimal bounding square. The topological border of a permutomino of size $r+1$ is a grid ogon with r reflex vertices, and so, it has $n = 2r + 4$ vertices in total.

In this paper, we give an algorithm for the enumeration of all permutominoes by size, based on the INFLATE-PASTE[1] technique introduced in [19]. Every

*Email: apt@dcc.fc.up.pt. Research partially supported by the European Regional Development Fund through the programme COMPETE and by the Portuguese Government through the FCT – Fundação para a Ciência e Tecnologia under the project PEst-C/MAT/UI0144/2011, and project JEDI (PTDC/EIA/66924/2006).

[1] Demos at `http://www.dcc.fc.up.pt/~apt/genpoly`

grid n-ogon P results from a unit square by applying InflatePaste r times, where $r = (n-4)/2$ is the number of reflex vertices of P. InflatePaste glues a new rectangle to a grid ogon to obtain a new one with 1 more reflex vertex. The rectangle is glued by Paste to an horizontal edge incident to a convex vertex, say v, must be contained in a region that we called *the free neighbourhood of* v (Fig. 2), and is fixed to v. This region, denoted by $FSN(v)$, consists of the

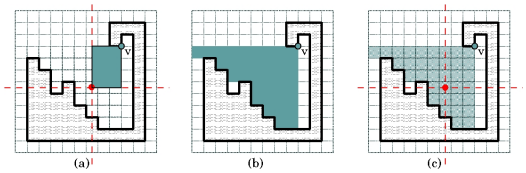


Figure 2: InflatePaste: (a) gluing a rectangle to v (b) $FSN(v)$ is the shaded region (c) the rectangle is defined by v and the center of a cell of $FSN(v)$.

external points that are rectangularly visible from v and belong to the quadrant with origin v that contains the horizontal edge incident to v (two points a and b are *rectangularly visible* if the axis-aligned rectangle that has a and b as opposite corners does not intersect the interior of the polygon). The Inflate operation keeps the grid regular: the grid lines are shifted, if needed, to insert two new horizontal and vertical lines for the new edges (Fig. 3).

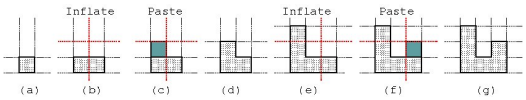


Figure 3: Two applications of InflatePaste: the center of the *inflated cell* becomes a convex vertex of the polygon created at each step.

In [19], $FSN(v)$ was called the free staircase neighborhood of v. Actually, $FSN(v)$ is a Ferrers diagram, with origin at v, and hence it can be given by a sequence of integers, each integer representing the number of cells that form each row of the diagram. Any cell in $FSN(v)$ could be used for growing the polygon using v. Therefore, for the example given in Fig. 2, we could produce $9+7+6+4+4+3+2 = 35$ distinct grid ogons using the selected vertex.

2 Direct Enumeration

ECO was introduced in [2] as a construction paradigm for the *enumeration of combinatorial objects* of a given class, by performing local transformations that increase a certain parameter (the size) of the objects.

In this section we propose a direct enumeration procedure for grid ogons (i.e., permutominoes) using InflatePaste. It is based on the existence of a unique depth-first generating tree for each n-ogon, once we fix the order for visiting the horizontal partition. One possibility is to define it as the order induced by a clockwise walk around the polygon, starting at its southwest vertex, as in Fig. 4. The bottom

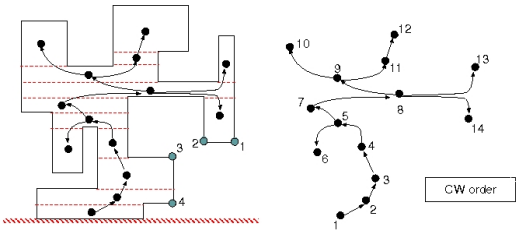


Figure 4: The unique generating tree for the represented grid ogon. Vertices 1, 2, 3 and 4 in the polygon are the ones that could still be expanded in our enumeration method, and should be visited in this order.

line (i.e., the horizontal line that contains the SW-vertex) is never moved upwards, but polygons can move freely along it. Fig. 1 shows how permutominoes can be generated by our method. The vertices marked with a cross would be no longer available for expansion.

```
PermutominoEnum(P,S,G,n0,r)
  if r = 0 then return fi
  MakeEmpty(TrS)
  while not IsEmpty(S) do
    v := Pop(S)   /* v is (vx, vy) */
    eH(v) := the horizontal edge of P that contains v
    if IsConvex(v,P) then
      for C in FreeNeighbourhood(v, P, G) do
        (p, q) := the southwest corner of C
        InflateGrid(p,q,G)
        w1 := (p+1,vy); w2 := (p+1,q+1);
        w3 := (vx,q+1)
        PasteRectangle(v, [w1, w2, w3], P)
        if w1 ∈ Interior(eH(v)) then
          Push({w2, w3},S,P)
        else Push({w1, w2, w3},S,P) fi
        OutputPolygon(P,n0+2)
        PermutominoEnum(P,S,G,n0+2,r−1)
        CutRectangle(v, [w1, w2, w3], P)
        DeflateGrid(p+1,q+1,G)
      done
    fi
    Push(v,TrS);
  done
  while not IsEmpty(TrS) do
    Push(Pop(TrS),S)
  done
```

Here, P is the initial polygon, G a representation of the grid lines, S a stack that contains the convex vertices of P that are available for expansion, n_0 the number of vertices of P and r the maximum number of reflex vertices of the polygons. PermutominoEnum enumerates recursively all descendants

of P that have up to $n_0 + 2r$ vertices. If initially $P := \{(1,1),(2,1),(2,2),(1,2)\}$ (w.r.t. the standard 2D cartesian coordinate system and given in CCW-order), $S := \{(1,2),(2,2)\}$, and $n_0 = 4$, then the algorithm will enumerate and count all grid ogons that have up to $2r+4$ vertices.

In contrast to other existing methods for the enumeration of polyominoes, PermutominoEnum, for permutominoes, does not need to keep an exponential number of state configurations in order to count them correctly. Each permutomino is generated exactly once and, hence, there is no need to check for repetitions. Nevertheless, the running time of the algorithm is dominated by the number of permutominoes generated (and thus it is exponential).

An algorithm for enumerating the *convex* permutominoes by size was published recently [10]. Its running cost is proportional to the number of permutominoes generated. It is quite easy to design a specialized version of our algorithm for enumerating convex permutominoes with identical complexity. Actually, as we will see, for convex permutominoes the free neighbourhoods are linear (rectangles of width 1) and only the two topmost convex vertices can be active.

3 Convex Permutominoes

Although the exact counting and enumeration of polyominoes remain challenging open problems, several positive results were achieved for special classes of polyominoes [5, 7, 15], namely for the class of convex polyominoes and some of its subfamilies (e.g., directed-convex polyominoes, parallelogram polyominoes, stack polyominoes, and Ferrers diagrams). The larger class of row-convex (resp. column-convex) polyominoes was considered also [11]. A polyomino is said to be *row-convex* (resp. *column-convex*) if all its rows (resp. *columns*) are connected, i.e., the associated orthogonal polygon is y-monotone (x-monotone). A polyomino is said to be *convex* if it is both row-convex and column-convex. These classes, which satisfy convexity and/or directness conditions, have been studied using different approaches and are fairly well characterized, for some parameters, e.g., area and perimeter [5]. The corresponding classes of permutominoes have been addressed recently [4, 8].

The analysis of the transformations performed by Inflate-Paste during the application of PermutominoEnum allow us to derive simple characterizations and exact countings for such classes of permutominoes. Fig. 5 shows all n-vertex convex permutominoes for $n = 4, 6, 8$, each one embedded on a grid. Only the two topmost convex vertices may be active for Inflate-Paste (so, L and R stand for left or right). Crossed vertices are inactive in the following transformation steps: "u" means that the vertex

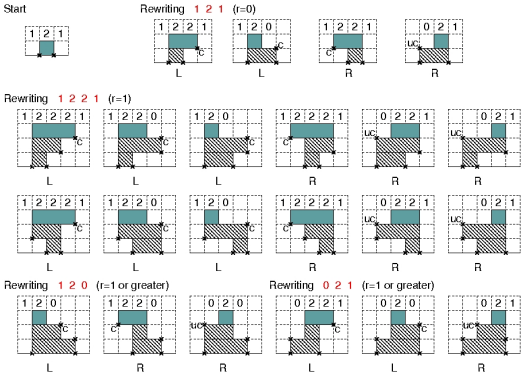


Figure 5: Enumerating convex permutominoes by size.

would be discarded in PermutominoEnum as well (due to uniqueness conditions) and "c" means that the resulting permutomino would not be convex. The sequence of $\{0,1,2\}^\star$ displayed on the grid top row is the *expansion key* of the corresponding permutomino. Each element of the key gives the number of active convex vertices that see a certain grid cell (in Fig. 5, each counter is in its cell). Here, *see* means that the cell belongs to the free neighborhood of the vertex. For all the remaining empty cells, the counter is 0 and, thus, we omitted it. If we add up the elements of the expansion key of a given convex permutomino, we get the number of convex permutominoes that it yields immediately in PermutaminoEnum. In this way, the expansion keys provide an exact encoding of the structural features that are relevant for counting convex permutominoes according to the number of vertices. Actually, it is the key as a whole that matters but not the particular cells associated to each counter. By analysing Inflate-Paste in the scope of PermutaminoEnum, we may conclude that the *expansion key* of any convex permutomino with $r \geq 0$ reflex vertices must be of one of the following forms:

$$\begin{array}{ll} 1\,2^{r+1}\,1 & \\ 1\,2^{j}\,0, & \text{for } 1 \leq j \leq r-1, \\ 0\,2^{j}\,1, & \text{for } 1 \leq j \leq r-1, \text{ and} \\ 0\,2^{j}\,0, & \text{for } 1 \leq j \leq r-2. \end{array}$$

Inflate-Paste operations acting on convex permutominoes can be seen as rewrite rules. Each rule rewrites the key of a convex permutomino with $r-1$ reflex vertices to the key of one of the convex permutominoes derived from it, having one more reflex vertex, for $r \geq 1$. The rewrite rules are:

$$\begin{array}{lcll} 12^r1 & \rightarrow^{L,R} & 12^{r+1}1 & \\ 12^r1 & \rightarrow^{L} & 12^{j}0, & \text{for } 1 \leq j \leq r \\ 12^r1 & \rightarrow^{R} & 02^{j}1, & \text{for } 1 \leq j \leq r \\ 12^{j'}0 & \rightarrow^{L} & 12^{j}0, & \text{for } 1 \leq j \leq j' \leq r-1 \\ 12^{j'}0 & \rightarrow^{R} & 12^{j'+1}0, & \text{for } 1 \leq j' \leq r-1 \\ 12^{j'}0 & \rightarrow^{R} & 02^{j}0, & \text{for } 1 \leq j \leq j' \leq r-1 \end{array}$$

$$
\begin{array}{llll}
02^{j'}1 & \rightarrow^{R} & 02^{j}1, & \text{for } 1 \leq j \leq j' \leq r-1 \\
02^{j'}1 & \rightarrow^{L} & 02^{j'+1}1, & \text{for } 1 \leq j' \leq r-1 \\
02^{j'}1 & \rightarrow^{L} & 02^{j}0, & \text{for } 1 \leq j \leq j' \leq r-1 \\
02^{j'}0 & \rightarrow^{L,R} & 02^{j}0, & \text{for } 1 \leq j \leq j' \leq r-2
\end{array}
$$

where L (left) and R (right) identify the topmost vertex selected. For rules with annotation "L, R", both vertices can be selected, one at a time. Figs. 6–8 illustrate the idea underlying these rules.

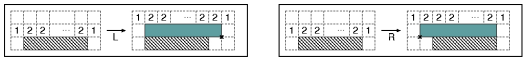


Figure 6: Rewriting $12^{r}1$ using the rewrite rules $12^{r}1 \rightarrow^{L} 12^{r+1}1$ and $12^{r}1 \rightarrow^{R} 12^{r+1}1$.

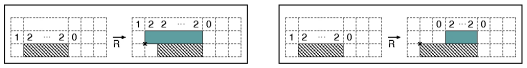


Figure 7: Rewriting $12^{j'}0$ using (a) $12^{j'}0 \rightarrow^{R} 12^{j'+1}0$ and (b) a rule $12^{j'}0 \rightarrow^{R} 02^{j}0$, for $1 \leq j \leq j'$.

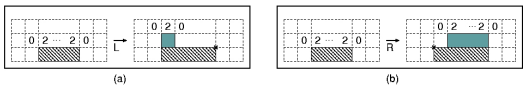


Figure 8: Rewriting $02^{j'}0$ using (a) $02^{j'}0 \rightarrow^{L} 02^{j}0$ and (b) $02^{j'}0 \rightarrow^{R} 02^{j}0$, for some $1 \leq j \leq j'$.

The correctness and completeness of this rewrite system can be checked easily by case-analysis, taking into account the conditions on convexity.

Proposition 1 *Let $A^{(r)}_{\alpha,j,\beta}$ be the number of convex permutominoes of the class $\alpha 2^{j} \beta$ with r reflex vertices, for $\alpha, \beta \in \{0,1\}$, $1 \leq j \leq r+1$ and $r \geq 0$. Then, $A^{(r)}_{0,j,1} = A^{(r)}_{1,j,0}$, for all r and j (symmetry by reflection w.r.t. V-axis) and $A^{(r)}_{\alpha,j,\beta}$ is inductively defined as follows.*

$$
\begin{aligned}
A^{(0)}_{1,1,1} &= 1 \\
A^{(r)}_{1,r+1,1} &= 2A^{(r-1)}_{1,r,1}, \text{ for } r \geq 1 \\
A^{(r)}_{1,j,0} &= A^{(r-1)}_{1,r,1} + \sum_{j'=\max(1,j-1)}^{r-1} A^{(r-1)}_{1,j',0}, \text{ for } 1 \leq j \leq r \\
A^{(r)}_{0,j,0} &= 2\sum_{j'=j}^{r-1} A^{(r-1)}_{1,j',0} + 2\sum_{j'=j}^{r-2} A^{(r-1)}_{0,j',0}, \text{ for } 1 \leq j \leq r-1
\end{aligned}
$$

The number of convex permutominoes of size n is given by sequence A126020, in [17]. A closed formula for this number is given in [8]. In a similar way, we may deduce the recurrences for row-convex permutominoes and the simpler classes of bargraphs, stacks, and Ferrers diagrams.

References

[1] T. Auer and M. Held, Heuristics for the generation of random polygons, in: *Proc. CCCG'96*, 1996, 38–43.

[2] E. Barcucci, A. Del Lungo, E. Pergola, and R. Pinzani, ECO: a methodology for the enumeration of combinatorial objects, *J. Differ. Equ. Appl.* **5** (1999), 435–490.

[3] G. Barequet and M. Moffie, On the complexity of Jensen's algorithm for counting fixed polyominoes, *J. Discrete Algorithms* **5** (2007), 348–355.

[4] P. Boldi, V. Lonati, R. Radicioni, and M. Santini, The number of convex permutominoes, *Information and Computation* **206** (2008), 1074–1083.

[5] M. Bousquet-Mélou, Bijection of convex polyominoes and equations for enumerating them according to area, *Discrete Appl. Math.* **48** (1994), 21–43.

[6] M. Damian, R. Flatland, J. O'Rourke, and S. Ramaswami, Connecting polygonizations via stretches and twangs, *Theory of Computing Systems* **47** (2010), 674–695.

[7] E. Deutsch, Enumerating symmetric directed convex polyominoes, *Discrete Math.* **280** (2004), 225–231.

[8] F. Disanto, A. Frosini, R. Pinzani, and S. Rinaldi, A closed formula for the number of convex permutominoes, *Electron. J. Combin.* **14** (2007), #R57.

[9] S. Golomb, *Polyominoes*, Princeton U. Press, 1994.

[10] E. Grazzini, E. Pergola, and M.Poneti, On the exhaustive generation of convex permutominoes, *Pure Mathematics and Applications* **19** (2008), 93–104.

[11] D. Hickerson, Counting horizontally convex polyominoes, *J. Integer Sequences* **2** (1999), Article 99.1.8.

[12] F. Insitti, Permutation diagrams, fixed points and Kazhdan-Lusztig R-polynomials, *Annals of Combinatorics* **10** (2006), 369–387.

[13] I. Jensen, Enumerations of lattice animals and trees, *J. Stat. Phys.* **102** (2001), 865–881.

[14] I. Jensen, Counting polyominoes: a parallel implementation for cluster computing, in: *Proc. ICCS'03*, LNCS, vol. 2659, Springer, 2003, 203–212.

[15] A. Del Lungo, E. Duchi, A. Frosini, and S. Rinaldi, On the generation and enumeration of some classes of convex polyominoes, *Electron. J. Combin.* **11** (2004), #R60.

[16] J. O'Rourke, An alternate proof of the rectilinear art gallery theorem, *J. Geometry* **21** (1983), 118–130.

[17] N. J. A. Sloane, The On-Line encyclopedia of integer sequences, OEIS Foundation, `http://oeis.org/`.

[18] C. Sohler, Generating random star-shaped polygons, in: *Proc. CCCG'99*, 1999.

[19] A. P. Tomás and A. Bajuelos, Quadratic-time linear-space algorithms for generating orthogonal polygons with a given number of vertices, in: *Proc. CGA'04–ICCSA'04*, LNCS, vol. 3045, Springer, 2004, 117–126.

[20] C. Zhu, G. Sundaram, J. Snoeyink, and J. S. B. Mitchell, Generating random polygons with given vertices, *Comput. Geom.* **6** (1996), 277–290.

Distance domination, guarding and vertex cover for maximal outerplanar graphs

Santiago Canales[1], Gregorio Hernández*[2], Mafalda Martins‡†[3], and Inês Matos‡[3]

[1]Universidad Pontificia Comillas de Madrid, Spain
[2]Universidad Politécnica de Madrid, Spain
[3]Universidade Aveiro & CIDMA, Portugal

Abstract

In this paper we define a distance guarding concept on plane graphs and associate this concept with distance domination and distance vertex cover concepts on triangulation graphs. Furthermore, for any n-vertex maximal outerplanar graph, we provide tight upper bounds for $g_{2d}(n)$ ($2d$-guarding number), $\gamma_{2d}(n)$ ($2d$-domination number) and $\beta_{2d}(n)$ ($2d$-vertex cover number).

Introduction

Domination, covering and guarding are widely studied subjects in graph theory. Given a graph $G = (V, E)$ a *dominating set* is a subset D of V such that every vertex not in D is adjacent to a vertex in D. A subset C of V is a *vertex cover* if each edge of the graph is incident to at least one vertex of the set. Thus, a dominating set guards the *vertices* of a graph while a vertex cover guards its *edges*. In plane graphs these concepts differ from the *guarding set* concept that guards the *faces* of the graph. Let $G = (V, E)$ be a plane graph, a guarding set is a subset S of V such that every face has a vertex in S. There are many papers and books about domination and its many variants in graphs, e.g. [4, 8, 9, 10]. Domination was extended to *distance domination* by Meir and Moon [11]: given a graph G, a subset D of V is said to be a *distance k-dominating set* if for each vertex $u \in V - D$, $dist_G(u, v) \leq k$ for some $v \in D$. The minimum cardinality of a distance k-dominating set is said to be the *distance k-domination number* of G and is denoted by $\gamma_{kd}(G)$. In the case of distance domination, there are also some known results concerning bounds for $\gamma_{kd}(G)$, e.g., [13, 14]. However, if graphs are restricted to triangulations, as far as we know, there are no known bounds for $\gamma_{kd}(G)$. The distance domination was generalized to *broadcast domination* when the power of each vertex may vary [6]. Given a graph $G = (V, E)$, a *broadcast* is a function $f : V \to \mathbb{N}_0$. A broadcast is *dominating* if for every vertex v, there is a vertex u such that $f(u) > 0$ and $d(u, v) \leq f(u)$. A dominating broadcast f is *optimal* if $\sum_{v \in V} f(v)$ is minimum over all choices of broadcast dominating functions for G. The *broadcast domination problem* consists in building this optimal function. Note that, if $f(V) = \{0, k\}$, then the broadcast domination problem is the distance k-dominating problem. If a broadcast f provides coverage to the edges of G instead of covering its vertices, we have a generalization of the vertex cover concept [2]. A broadcast f is *covering* if for every edge $(x, y) \in E$, there is a path P in G that includes the edge (x, y) and one end of P must be a vertex u, where $f(u)$ is at least the length of P. A covering broadcast f is *optimal* if $\sum_{v \in V} f(v)$ is minimum over all choices of broadcast covering functions for G. Note that, if $f(V) = \{0, 1\}$, then the broadcast cover problem coincides with the problem of finding a minimum vertex cover. Regarding the broadcast cover problem when all vertices have the same power (i.e., when $f(V) = \{0, k\}$, for a fixed $k \neq 1$), as far as we know, there are no published results besides [5] where the authors propose a centralized and distributed approximation algorithm to solve it. Concerning the guarding concept there are also known bounds on the guarding number $g(G)$. For example, $g(G) \leq \lfloor \frac{n}{2} \rfloor$, for any n-vertex plane graph [3], and if G is a maximal outerplanar graph this bound is $\lfloor \frac{n}{3} \rfloor$ [7]. Contrary to the notions of domination and vertex cover on plane graphs that were extended to include its distance versions, the guarding concept was not generalized to its distance version.

In this paper we generalize the guarding concept on plane graphs to its *distance guarding* version. We also formalize the broadcast cover problem when all vertices have the same power, which we call *distance*

*Research supported by ESF EUROCORES programme EuroGIGA - ComPoSe IP04 - MICINN Project EUI-EURC-2011-4306.

†Email: mafalda.martins@ua.pt. Research also supported by FCT grant SFRH/BPD/66431/2009.

‡Research supported by *FEDER* funds through *COMPETE*–Operational Programme Factors of Competitiveness, CIDMA and FCT within project PEst-C/MAT/UI4106/2011 with COMPETE number FCOMP-01-0124-FEDER-022690.

k-vertex cover. Since there are no combinatorial results for the concepts of domination, vertex cover and guarding on its distance versions on triangulation graphs (*triangulations*, for short), we also address this problem. In the next section we describe some of the terminology that will be used throughout this paper.

1 Preliminaries

Given a triangulation of a point set $T = (V, E)$, we say that a bounded face T_i of T (i.e., a triangle) is *kd-visible* from a vertex $p \in V$, if there is a vertex $x \in T_i$ such that $dist_T(x, p) \leq k - 1$. The *kd-visibility region* of a vertex $p \in V$ comprises the triangles of T that are kd-visible from p (see Fig. 1).

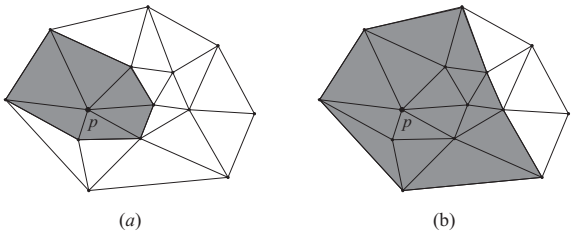


Figure 1: The kd-visible region of p for: (a) $k = 1$; (b) $k = 2$.

A *kd-guarding set* for T is a subset $F \subseteq V$ such that every triangle of T is kd-visible from an element of F. We designate the elements of F by *kd-guards*. The *kd-guarding number* $g_{kd}(T)$ is the number of vertices in a smallest kd-guarding set for T. Note that, to avoid confusion with *multiple guarding* [1] - where the typical notation is k-guarding - we will use kd-guarding, with an extra "d". Given a set S and a positive integer n, we define $g_{kd}(S) = max\{g_{kd}(T) : T = (V, E)$ is triangulation with $V = S\}$ and $g_{kd}(n) = max\{g_{kd}(S) : |S| = n\}$. A *kd-vertex cover* for T, or *distance k-vertex cover* for T, is a subset $C \subseteq V$ such that for each edge $e \in E$ there is a path of length at most k, which contains e and a vertex of C. The *kd-vertex cover number* $\beta_{kd}(T)$ is the number of vertices in a smallest kd-vertex cover set for T. Given a set S and a positive integer n, we define $\beta_{kd}(S) = max\{\beta_{kd}(S) : T = (V, E)$ is triangulation with $V = S\}$ and $\beta_{kd}(n) = max\{\beta_{kd}(S) : |S| = n\}$. Finally, as already defined by other authors, a *kd-dominating set* for T, or *distance k-dominating set* for T, is a subset $D \subseteq V$ such that each vertex $u \in V - D$, $dist_T(u, v) \leq k$ for some $v \in D$. The kd-domination number $\gamma_{kd}(T)$ is the number of vertices in a smallest kd-dominating set for T. Given a set S and a positive integer n, we define $\gamma_{kd}(S) = max\{\gamma_{kd}(T) : T = (V, E)$ is triangulation with $V = S\}$ and $\gamma_{kd}(n) = max\{\gamma_{kd}(S) : |S| = n\}$. Our main goal is to obtain bounds on $\gamma_{kd}(T)$, $g_{kd}(T)$ and $\beta_{kd}(T)$. We start by establishing a tight upper bound for $g_{2d}(n)$ for a special class of triangulations, namely the *maximal outerplanar graphs*. A graph is a maximal outerplanar graph if it is a triangulation of a simple polygon without holes [12]. Edges on the exterior face are called *exterior edges*, and *interior edges* otherwise.

In the next section we show that there is a relationship between $2d$-guarding, $2d$-dominating and $2d$-vertex cover sets on triangulations. In sections 3 and 4 we provide upper bounds for $g_{2d}(n)$, $\gamma_{2d}(n)$ and $\beta_{2d}(n)$ on maximal outerplanar graphs and show that these bounds are tight.

2 Relationship between distance vertex cover, guarding and domination on triangulations

We start by showing that the three concepts are different. Fig. 2 depicts $2d$-dominating and $2d$-guarding sets for a given triangulation. Note that in Fig. 2(a) the set $\{u, v\}$ is $2d$-dominating since the remaining vertices are at distance 1 or 2. However, it is not a $2d$-guarding set because the shaded triangle is not guarded, as its vertices are at distance 2 from $\{u, v\}$. In 2(b) $\{w, z\}$ is a $2d$-guarding set, however it is not a $2d$-vertex cover since any path between the bold edge and w or z has length at least 3. Therefore, the bold edge is not covered.

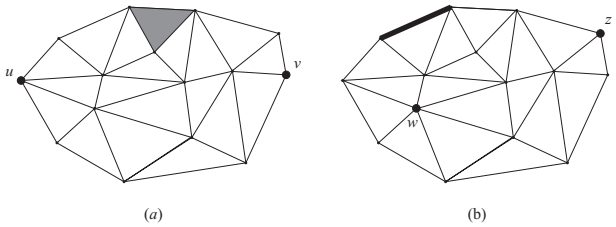


Figure 2: (a) $2d$-dominating set for a triangulation T; (b) $2d$-guarding set for T.

Now we are going to establish a relation between $g_{2d}(T)$, $\gamma_{2d}(T)$ and $\beta_{2d}(T)$. The following results can be easily generalized to $g_{kd}(T)$, $\gamma_{kd}(T)$ and $\beta_{kd}(T)$.

Lemma 1 *If C is a $2d$-vertex cover of a triangulation T, then C is a $2d$-guarding set and a $2d$-dominating set of T.*

Lemma 2 *If F is a $2d$-guarding set of a triangulation T, then F is a $2d$-dominating set of T.*

The previous lemmas prove the following result.

Theorem 3 *Given a triangulation T the minimum cardinality $g_{2d}(T)$ of any $2d$-guarding set verifies $\gamma_{2d}(T) \leq g_{2d}(T) \leq \beta_{2d}(T)$.*

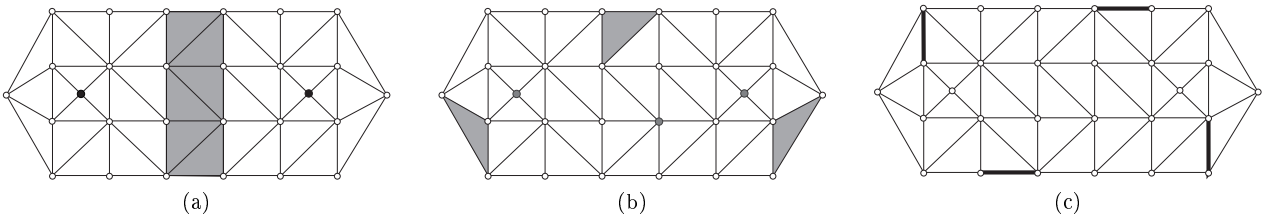


Figure 3: (a) A $2d$-dominating set for a triangulation T (black vertices); (b) a $2d$-guarding set for T (gray vertices); (c) each of the bold edges needs a different vertex to be $2d$-covered.

Note that the inequalities above can be strict, as we can see in the triangulation T presented in Fig. 3, where $\gamma_{2d}(T) = 2$, $g_{2d}(T) = 3$, and $\beta_{2d}(T) \geq 4$.

3 $2d$-domination and $2d$-guarding of maximal outerplanar graphs

In this section we establish upper bounds for $g_{2d}(n)$ and $\gamma_{2d}(n)$ on maximal outerplanar graphs. In order to do this, and following the ideas of O'Rourke [12], we first need to introduce some lemmas.

Lemma 4 *Suppose that $f(m)$ guards are always sufficient to $2d$-guard any outerplanar maximal graph G with m vertices. Then if G has two guards placed at any two adjacent vertices or one guard placed at any one of its vertices, then $f(m-2)$ or $f(m-1)$ additional guards are sufficient to $2d$-guard G, respectively.*

Lemma 5 *Let G be an outerplanar maximal graph with $n \geq 2k$ vertices. There is an interior edge e of G that partitions G into two pieces, one of which contains $m = k, k+1, \ldots, 2k-1$ or $2k-2$ exterior edges of G.*

Theorem 6 *Every n-vertex maximal outerplanar graph, with $n \geq 5$, can be $2d$-guarded by $\lfloor \frac{n}{5} \rfloor$ $2d$-guards. That is, $g_{2d}(n) \leq \lfloor \frac{n}{5} \rfloor$ for all $n \geq 5$.*

Proof. For $5 \leq n \leq 11$, the truth of the theorem can be easily established. It should be noted that for $n = 5$ the $2d$-guard can be placed randomly and for $n = 6$ it can be placed at any vertex of degree greater than 2. Assume that $n \geq 12$ and that the theorem holds for all $n' < n$. Lemma 5 guarantees the existence of an interior edge e that divides G into two maximal outerplanar graphs G_1 and G_2, such that G_1 has m exterior edges of G with $6 \leq m \leq 10$. The vertices of G are labeled with $0, 1, \ldots, n-1$ such that e is $(0, m)$. Each value of m is considered separately. We present the cases $m = 6$ and $m = 9$.

Case $m = 6$. G_1 has $m + 1 = 7$ exterior edges, thus G_1 can be $2d$-guarded with one guard. G_2 has $n - 5$ exterior edges including e, and by induction hypothesis, it can be $2d$-guarded with $\lfloor \frac{n-5}{5} \rfloor = \lfloor \frac{n}{5} \rfloor - 1$ guards. Thus G_1 and G_2 together can be $2d$-guarded by $\lfloor \frac{n}{5} \rfloor$ guards.

Case $m = 9$. The presence of any of the internal edges (0,8), (0,7), (0,6), (9,1), (9,2) and (9,3) would violate the minimality of m. Thus, the triangle T in G_1 that is bounded by e is either (0,5,9) or (0,9,4). Since these are equivalent cases, suppose that T is (0,5,9) (see Fig. 4(a)).

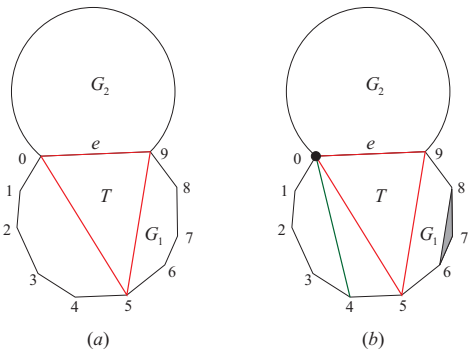


Figure 4: (a) The triangle T is (0,5,9); (b) the internal edge (0,4) and the triangle (6,7,8) are present.

The pentagon (5,6,7,8,9) can be $2d$-guarded by placing one guard at a chosen vertex. However, to $2d$-guard the hexagon (0,1,2,3,4,5) we cannot place a guard randomly.We will consider two separate cases: (i) The internal edge (0,4) is not present: if a guard is placed at vertex 5, then the hexagon (0,1,2,3,4,5) is $2d$-guarded, thus G_1 is $2d$-guarded. Since G_2 has $n-8$ edges it can be $2d$-guarded by $\lfloor \frac{n-8}{5} \rfloor \leq \lfloor \frac{n}{5} \rfloor - 1$ guards applying the induction hypothesis. This yields a $2d$-guarding of G by $\lfloor \frac{n}{5} \rfloor$ guards; (ii) The internal edge (0,4) is present: if a guard is placed at vertex 0, then G_1 is $2d$-guarded unless the triangle (6,7,8) is present in the triangulation (see Fig. 4(b)). In any case, two $2d$-guards placed at vertices 0 and 9 guard G_1. G_2 has $n - 8$ exterior edges, including e. By lemma 4 the two guards placed at vertices 0 and 9 allow the remainder of G_2 to be guarded by $f(n-8-2) = f(n-10)$ additional $2d$-guards. Recall that $f(n')$ is the number of $2d$-guards that are always sufficient to guard a maximal outerplanar graph with

n' vertices. By the induction hypothesis $f(n') = \lfloor \frac{n'}{5} \rfloor$. Thus, $\lfloor \frac{n-10}{5} \rfloor = \lfloor \frac{n}{5} \rfloor - 2$ guards suffices to $2d$-guard G_2. Together with the guards placed at vertices 0 and 9 that $2d$-guard G_1, all G is guarded by $\lfloor \frac{n}{5} \rfloor$ $2d$-guards.

□

To prove that this upper bound is tight we need to construct a maximal outerplanar graph G of order n such that $g_{2d}(G) = \lfloor \frac{n}{5} \rfloor$. Fig. 5 shows a maximal outerplanar graph G for which $\gamma_{2d}(G) = \frac{n}{5}$, since the black vertices can only be $2d$-dominated by different vertices. Thus, $\gamma_{2d}(n) \geq \frac{n}{5}$. Note that this example can be generalized to kd-domination to obtain $\gamma_{kd}(n) \geq \frac{n}{(2k+1)}$.

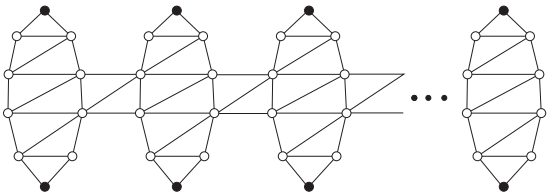


Figure 5: A maximal outerplanar graph G for which $\gamma_{2d}(G) = \frac{n}{5}$.

According to theorem 3, $\gamma_{2d}(G) \leq g_{2d}(G)$, so $\lfloor \frac{n}{5} \rfloor \leq g_{2d}(G)$. In conclusion, $\lfloor \frac{n}{5} \rfloor$ $2d$-guards are occasionally necessary and always sufficient to guard a n-vertex maximal outerplanar graph G. On the other hand, we can also establish that $\gamma_{2d} = \lfloor \frac{n}{5} \rfloor$, since $\lfloor \frac{n}{5} \rfloor \leq \gamma_{2d}(n)$ and $\gamma_{2d}(n) \leq g_{2d}(n)$, for all n. Thus, it follows:

Theorem 7 *Every n-vertex maximal outerplanar graph with $n \geq 5$ can be $2d$-guarded (and $2d$-dominated) by $\lfloor \frac{n}{5} \rfloor$ guards. This bound is tight.*

4 $2d$-covering of maximal outerplanar graphs

In this section we determine an upper bound for $2d$-vertex cover on maximal outerplanar graphs which is also tight.

Lemma 8 *Suppose that $f(m)$ vertices are always sufficient to $2d$-cover any outerplanar maximal graph G with m vertices. If we randomly choose a vertex of G to be a $2d$-covering vertex, then $f(m-1)$ additional vertices are sufficient $2d$-cover G.*

Theorem 9 *Every n-vertex maximal outerplanar graph, with $n \geq 4$, can be $2d$-covered with $\lfloor \frac{n}{4} \rfloor$ vertices. That is, $\beta_{2d}(n) \leq \lfloor \frac{n}{4} \rfloor$ for all $n \geq 4$.*

Now, we will prove that this upper bound is tight. The bold edges of the maximal outerplanar graph illustrated in Fig. 6 can only be $2d$-covered from different vertices, and therefore $\beta_{2d}(n) \geq \frac{n}{4}$. To conclude:

Theorem 10 *Every n-vertex maximal outerplanar graph with $n \geq 5$ can be $2d$-covered by $\lfloor \frac{n}{4} \rfloor$ vertices. This bound is tight.*

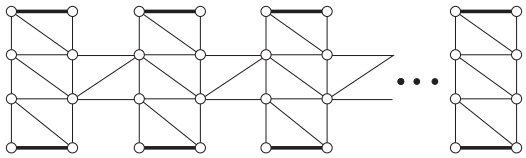


Figure 6: A maximal outerplanar graph G for which $\beta_{2d}(G) = \frac{n}{4}$.

References

[1] P. Belleville, P. Bose, J. Czyzowicz, J. Urrutia, J. Zaks, K-vertex guarding simple polygons, *Comput. Geom. Theory Appl.* , **42(4)** (2009), 352–361.

[2] J.R.S. Blair and S.B. Horton, Broadcast covers in graphs, *Congr. Num.* **173** (2005), 109–115.

[3] P. Bose, T. Shermer, G. Toussaint, and B. Zhu, Guarding Polyhedral Terrains, *Computational Geometry: Theory and Applications*, **6(3)**(1997), 173–185.

[4] C.N. Campos and Y. Wakabayashi, On dominating sets of maximal outerplanar graphs, *Discrete Appl.Math*, **161(3)** (2013), 330–335.

[5] Q. Chen, L. Zhao, Approximation algorithms for the L-distance vertex cover problem, *Proc. ICTMF* (2012), Bali, Indonesia, 100–104.

[6] D. Erwin, Dominating broadcasts in graphs, *Bull. Inst. Combin. Appl.* **42** (2004), 89–105.

[7] S. Fisk. A short proof of Chvatal's watchman theorem, *J. Combin. Theory Ser. B*, **(24)** (1978), 374.

[8] T.W. Haynes, S.T. Hedetniemi, P.J. Slater, *Fundamentals of Domination in Graphs*, M. Dekker, 1998.

[9] E. King, J. Pelsmajer, Dominating sets in plane triangulations, *Discret Math.* **310(17-18)** (2010), 2221–2230.

[10] L.R. Matheson, R.E. Tarjan, Dominating sets in planar graphs, *Eur. J. Combin.* **17(6)** (1996), 565–568.

[11] A. Meir, J.W. Moon, Relations between packing and covering number of a tree, *Pacific J. Math* 61(1) (1975), 225–233.

[12] J. O'Rourke, *Art Gallery Theorems and Algorithms*, Oxford University Press, Inc., New York, USA, 1987.

[13] F. Tian, J. Xu, Bounds for distance domination numbers of graphs, *Journal of University of Science and Technology of China* **34(5)** (2004), 529–534.

[14] F. Tian, J. Xu, A note on distance domination numbers of graphs, *Aust. J. of Combin* **43** (2009), 181–190.

Abstract Voronoi diagrams

Rolf Klein*

Universität Bonn, Institut für Informatik I

Abstract

Abstract Voronoi diagrams are a unifying framework that covers many types of concrete Voronoi diagrams. This talk reports on the state of the art, including recent progress.

Introduction

Concrete Voronoi diagrams [1] are mostly defined in terms of sites and distance, and both concepts can vary greatly. Abstract Voronoi diagrams [6] are built on what most concrete diagrams have in common: a system of simple bisecting curves $J(p,q) = J(q,p)$, where p, q are just indices from a set S of n elements. Each curve $J(p,q)$ divides the plane into two domains, $D(p,q)$ and $D(q,p)$. The *abstract Voronoi region* of p with respect to S is defined by

$$\mathrm{VR}(p,S) := \bigcap_{q \in S \setminus \{p\}} D(p,q)$$

and the *abstract Voronoi diagram* of S is just the plane minus all Voronoi regions.

An interesting question is what properties to require of the curves $J(p,q)$. They should be as weak as possible for generality, but strong enough to ensure that useful "Voronoi" structures result from the above definitions. It turns out [7] that the following are sufficient.

(A1) Each curve $J(p,q)$, where $p \neq q$, is unbounded. After stereographic projection to the sphere, it can be completed to a closed Jordan curve through the north pole.

For any three indices p, q, r in S, and $S' := \{p,q,r\}$,

(A2) each Voronoi region $\mathrm{VR}(p,S')$ is path-wise connected,

(A3) each point of the plane belongs to the closure of a Voronoi region $\mathrm{VR}(p,S')$.

*Email: rolf.klein@uni-bonn.de. This work was supported by the European Science Foundation (ESF) in the EUROCORES collaborative research project EuroGIGA/VORONOI.

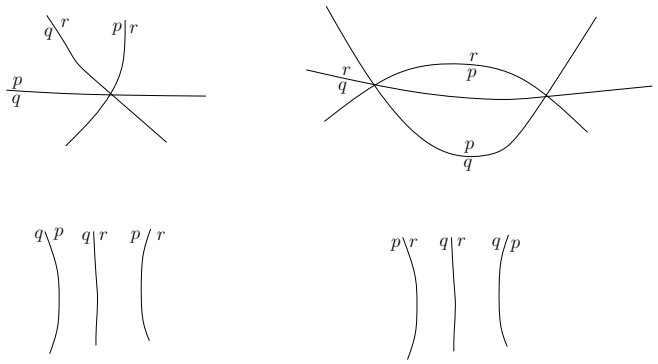


Figure 1: Admissible curve systems

Informally, if the bisecting curves are unbounded and behave decently, and if any triplet $J(p,q), J(p,r), J(q,r)$ is situated as shown in Figure 1, the AVD theory applies.

1 Results

This means that structural results and efficient algorithms become available without further effort [7].

Theorem 1 *$V(S)$ is a planar graph of complexity $O(n)$. It can be constructed in an expected number of $O(n \log n)$ many steps.*

If we replace Axiom A2 by the more general requirement

(A2') Each Voronoi region $\mathrm{VR}(p,S')$ has at most s connected components

(and assume that any two curves intersect only finitely often), the above result can be generalized as follows [3].

Theorem 2 *Abstract Voronoi diagrams with disconnected regions can be computed in an expected number of*

$$O\left(s^2 n \sum_{j=3}^{n} \frac{m_j}{j}\right)$$

steps, where m_j denotes the average number of faces per region in all AVDs of j sites from S.

One can extend the definition of abstract Voronoi diagrams to orders $k > 1$ by defining

$$\mathrm{VR}^k(P,S) := \bigcap_{p\in P,\ q\in S\setminus P} D(p,q).$$

For order $k = n-1$, the resulting AVDs are trees [9] of linear size. In the general case the following complexity result holds. Here we assume that all curves are in general position, and that the standard Voronoi-regions are non-empty.

Theorem 3 *The abstract order-k Voronoi diagram $V^k(S)$ has at most $2k(n-k)$ many faces. This bound can be achieved.*

2 Conclusion

Open is the case of closed bisecting curves.

References

[1] F. Aurenhammer, R. Klein, and D.-T. Lee, *Voronoi Diagrams and Delaunay Triangulations*, World Scientific Publishing Company, to appear August 2013.

[2] C. Bohler, P. Cheilaris, R. Klein, C.H. Liu, E. Papadopoulou, and M. Zavershynskyi, On the complexity of higher order abstract Voronoi diagrams, to be presented at ICALP' 13.

[3] C. Bohler and R. Klein, Abstract Voronoi diagrams with disconnected regions, Bonn 2013, manuscript.

[4] C. Bohler and R. Klein, Point sites with individual distance functions, Bonn 2012, manuscript.

[5] A. G. Corbalan, M. Mazon, and T. Recio, Geometry of bisectors for strictly convex distance functions, *International Journal of Computational Geometry and Applications* **6(1)** (1996), 45–58.

[6] R. Klein, *Concrete and abstract Voronoi diagrams*, Lecture Notes in Computer Science, 400, Springer-Verlag, 1987.

[7] R. Klein, E. Langetepe, and Z. Nilforoushan, Abstract Voronoi diagrams revisited, *Computational Geometry: Theory and Applications* **42(9)** (2009), 885–902.

[8] R. Klein, K. Mehlhorn, and S. Meiser, Randomized incremental construction of abstract Voronoi diagrams, *Computational Geometry: Theory and Applications* **3** (1993), 157–184.

[9] K. Mehlhorn, S. Meiser, and R. Rasch, Furthest site abstract Voronoi diagrams, *International Journal of Computational Geometry and Applications* **11(6)** (2001), 583–616.

Equipartitioning triangles

Pedro Ramos*[1] and William Steiger†[2]

[1]Department of Physics and Mathematics, University of Alcalá, Alcalá de Henares, Spain.
[2]Department of Computer Science, Rutgers University, 110 Frelinghuysen Road, Piscataway, New Jersey 08854-8004.

Abstract

An intriguing conjecture of Nandakumar and Ramana Rao is that for every convex body $K \subseteq R^2$, and for any positive integer n, K can be expressed as the union of n convex sets with disjoint interiors and each having the same area and perimeter. The first difficult case - $n = 3$ - was settled by Bárány, Blagojević, and Szucs using powerful tools from algebra and equivariant topology. Here we give an elementary proof of this result in case K is a triangle, and show how to extend the approach to prove that the conjecture is true for triangles.

Introduction

Let K be a convex body in the plane. Nandakumar and Ramana Rao [7] noticed that if a ham-sandwich cut for K were rotated through π radians - always maintaining a bisection of K - then at some point in this process, K is partitioned into two convex parts with disjoint interiors, and each having the same area and perimeter. A slightly more careful argument using this fact, along with induction, was given to show that for $n = 2^k$, K can always be partitioned into n convex subsets, each with the same area and perimeter. They made the intriguing

Conjecture 1 *For every $n \in N$ and all convex bodies $K \subseteq R^2$, K is the disjoint union of n convex pieces, each with the same area and perimeter.*

The conjecture describes an $n-$*equipartition* of K (because of the n equal areas) which is in addition *fair*, by virtue of the equal perimeters. Observe that, in the related problems of *cake partitioning* [1] only the perimeter of ∂K is taken into account.

*Email: pedro.ramos@uah.es. Partially supported by MEC grant MTM2011-22792 and by the ESF EUROCORES programme EuroGIGA, CRP ComPoSe, under grant EUI-EURC-2011-4306.

†Email: steiger@cs.rutgers.edu. Work supported in part by grant 0944081, National Science Foundation, USA. The author thanks The University of Alcala for supporting a visit to their Department of Mathematics. He is grateful to that department for their congenial hospitality.

Bárány et. al. [3], using heavy-duty tools from algebra and equivariant topology settled the case $n = 3$: A 3−*fan* is a point P in the plane with three rays emanating from it. It is convex if all angles are at most π. It equipartitions K if the three rays divide K into three regions of equal area, and it is fair, if these regions also have equal perimeter. Bárány et. al. showed that there is a convex 3−fan that makes a *fair equi-partition* of K. Subsequently, Aronov and Hubard [2] and then Karasev [6], showed that the conjecture was true for $n = p^k$, a prime power, and also in dimension $d \geq 2$, with "area" replaced by "volume" and "perimeter", by "surface area". Blagojević and Ziegler found some problems with the proofs in these two papers, so they established the results - and more - using different tools. In the present paper, in an attempt to understand some of the geometric features of this problem and why - or why not - it may be difficult, we use (only) elementary methods to study the conjecture for R^2, and when K is a triangle. We call a 3−fan *interior for* K if the apex P is interior to K; otherwise it is *exterior*. In the first case, all three rays play a role in the partition. In the second case, the partition is simply via two chords (which might meet on the boundary of K, but not in its interior). Because our results concern only exterior 3-fans, we will state them using chords. A main result of this paper is

Theorem 1 *Every triangle has a fair equi-partition defined by two disjoint chords.*

More importantly, the ideas used to prove this result can be extended to show that every triangle has a fair n-partition defined by $n - 1$ disjoint chords.

Corollary 2 *For every $n \geq 1$, every triangle has a fair, n-equipartition using $n - 1$ disjoint chords.*

In [7] it was observed that when K is a triangle, it can be covered by $n = k^2$ disjoint triangles similar to K, and all with the same area (and perimeter). But Theorem 1 is a new step toward resolving Conjecture 1, if only for triangles.

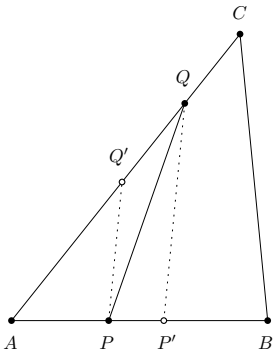


Figure 1: Illustration for Lemma 2.

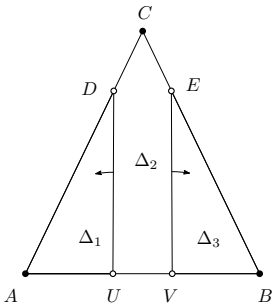


Figure 2: Rotation for an isosceles triangle.

1 Some details

In this section we describe some ideas behind the proofs for our results. The full proofs will appear in the actual paper.

We are able to understand this problem in the triangle case partly due to a simple tool that describes how perimeters change when a chord in a partition is moved slightly while still preserving the areas of all regions.

We use a Cartesian coordinate system in the Euclidean plane, and denote by AB and $|AB|$, respectively, the segment defined by points A and B and its length. Vector $\overrightarrow{OP}$ and point P will be identified if the context is clear enough. The list of points $AB\cdots D$ will be used to denote the corresponding polygon and, finally, $\pi(\cdot)$ will be the perimeter of the polygon.

Let A, B, C be non collinear points, and fix A as the origin of the coordinate system.

Lemma 3 *Consider points $P \in AB$ and $Q \in AC$ such that $|AP| \leq |AQ|$. Let $P' = tP$ and let $Q' = \frac{1}{t}Q$ (then the area of APQ equals the area of $AP'Q'$).*

1. *$\pi(AP'Q')$ and $\pi(BCQ'P')$ are convex functions of t, achieving their minima when $|AP'| = |AQ'|$.*
2. *If we write $\Delta\pi_1 = \pi(AP'Q') - \pi(APQ)$ and $\Delta\pi_2 = \pi(BCQ'P') - \pi(BCQ'P')$ then $|\Delta\pi_1| \geq |\Delta\pi_2|$.*

Consider a unit area triangle $\Delta = ABC$. Without loss of generality, we can assume that the smallest side is AB and that the coordinates of its vertices are $A = (0,0), B = (b,0)$, and $C = (c, 2/b)$, with $b > 0$ and $c \geq b/2$, as in the figure above. If we take points $U = (b/3, 0)$ and $V = (2b/3, 0)$, Δ is partitioned into $\Delta_1 = CAU, \Delta_2 = CUV$, and $\Delta_3 = CVB$, *all with the same area.* The goal is to rotate the chords CU and CV maintaining equality of all three areas, but in such a way as to force all three perimeters to coincide. To describe this process unambiguously, we place points D and E on the boundary of Δ. Initially both points are placed on C. We then manipulate the chords DU and EV by moving the endpoints along the boundary of Δ, thus altering the three sets in the partition. We use the notation π_i to denote the perimeter of region i.

The case where $x = b/2$, and Δ is isosceles, is easiest. Here, we move chord DU counter-clockwise (maintaining the area of $\Delta_1 = DAU$), and chord EV clockwise (maintaining the area of $\Delta_3 = EVB$) until U and V coincide at $F = (b/2, 0)$ (see Figure 2). During this process the middle region is a pentagon $\Delta_2 = CDUVE$, ending at $\Delta_2 = CDFE$. If we also keep $U+V = (b,0)$, there will be a position where the partition is fair, by the intermediate value theorem, since Δ_1 and Δ_3 initially have equal perimeters, but larger than that of Δ_2, and at the end, $\pi_1 = \pi_3 \leq \pi_2$, by virtue of AB being the smallest side of Δ.

When Δ is not isosceles, we rotate DU, again maintaining the area, till the perimeters of two regions are equal. Observe that, during the rotation, both π_1 and π_2 decrease. If $b/2 < c < 2b/3$, because we start with perimeters $\pi_1 > \pi_3 > \pi_2$, we must reach case 1 in Figure 3, where $\pi_1 = \pi_3 > \pi_2$. If $c \geq 2b/3$, we start with perimeters $\pi_1 > \pi_2 \geq \pi_3$. When rotating DU we can get $\pi_2 = \pi_3 < \pi_1$ (case 2) or $\pi_1 = \pi_2 > \pi_3$ (case 3).

Using Lemma 3 we know how to proceed in each case. For case 1, we rotate the chord EV clockwise, and the chord DU counterclockwise, preserving equality for the areas of regions 1 and 3. For case 2, we rotate the chords in the same way, maintaining now equal areas for regions 2 and 3. Finally, for case 3 both chords are rotated counterclockwise, keeping areas of regions 1 and 2 equal. Theorem 1 will be proven if we show that during this rotation, equality of the three perimeters is achieved.

For cases 1 and 2, the key is to observe that the foot of the left chord reaches the midpoint of AB first. If U is the midpoint of AB, we consider the triangle UBE', congruent with AUD (see Figure 4,left). From this, it is not hard to see that Δ_3 has to be VBE, with $V \in UB$ and that $\pi_1 < \pi_2$.

For case 3, consider a partition into three pieces of

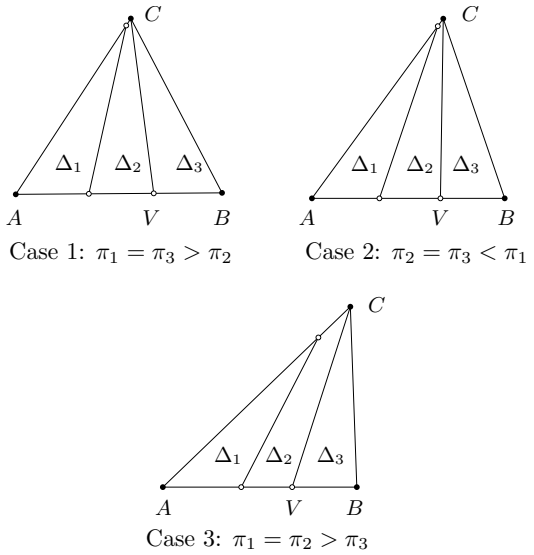


Figure 3: After the first rotation, two perimeters are equal.

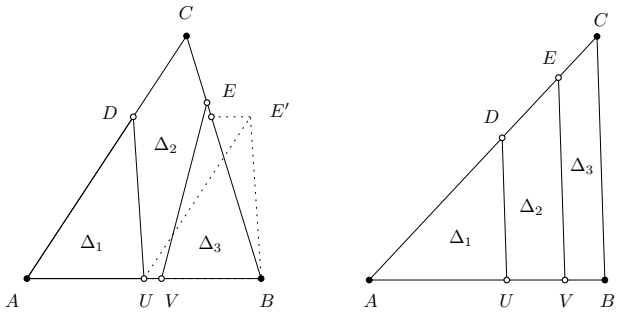


Figure 4: Situation at the end of the rotation.

equal area (but different perimeters) using two chords parallel to BC (see Figure 4,right). In this setting, it can be shown that $\pi_1 + \pi_2 < 2\pi_3$. Therefore, if we consider now the chord that divides AVE into two pieces of equal area and perimeter, from Lemma 3 it follows that perimeters of regions 1 and 2 are smaller and in that situation we must have $\pi_1 = \pi_2 < \pi_3$.

The approach of the arguments above can be applied to prove the existence of a fair n-equipartition of every triangle. Take points $V_i = (ib/n, 0)$ and points $U_i, i = 0, \ldots, n$; initially all $U_i = C$. The $n-1$ chords $U_iV_i, i = 1, \ldots, n-1$ partition Δ into n triangles ($\Delta_i = U_{i-1}V_{i-1}V_i, i = 1, \ldots, n$), of equal area. We order the chords left to right, and so the chord with foot at V_i is the i-th chord. The chords for which $V_i < c$ are called *left chords*, and the rest are the *right chords*.

We start by rotating the first chord counterclockwise, until the perimeter of the first region is equal, either to the perimeter of the second region, or to the perimeter of the last one. In the first case, we continue by rotating the first and the second chord (both counterclockwise), maintaining equality of the perimeters of the first two regions. In the second case, we continue by rotating the first chord counterclockwise, and the last chord clockwise, again maintaining equal perimeters for the involved regions. This process can be iterated, and when we reach the "central region" (the one in between the last left chord and the first right chord), we will have cases analogous to the ones in Figure 3. We proceed in the same way here, and it can be shown than equality of all perimeters is obtained before reaching a critical situation. The proof of this fact is more involved and will appear in the full version of this paper.

2 Discussion

It is clear that an equilateral triangle has two distinct fair 3-equipartitions that are interior (as well as three exterior ones - for each vertex, the process outlined in the previous section produces an equi-partition with an exterior 3-fan). It is easy to see that skinny triangles do not have fair interior 3-equipartitions. It would be interesting to characterize which triangles have equipartitions both exterior and interior, and which have only exterior ones.

It is easy to see that fair equipartitions of disks have to be radial, with the vertex at the center. Therefore, we cannot expect to extend the approach of this paper to the general case. Nevertheless, some convex bodies may have fair n-equipartitions produced by $n-1$ disjoint chords. It would be interesting to try to characterize this family.

References

[1] J. Akiyama and A. Kaneko and M. Kano and G. Nakamura and E. Rivera-Campo and S. Tokunaga and J. Urrutia. "Radial perfect partitions of convex sets in the plane". *Discrete and computational geometry (Tokyo, 1998)*, 1-13, Lecture Notes in Comput. Sci., 1763, Springer, Berlin, 2000.

[2] B. Aronov and A. Hubard. "Convex equipartitions of volume and surface area". arXiv.org/abs/1010.4611, 2010.

[3] I. Bárány, I., P. Blagojević, and A. Szucs. "Equipartitioning by a convex 3-fan". *Advances in Mathematics, 223 (2)*, 579-593, 2010.

[4] I. Bárány, I., P. Blagojević, and A. Blagojević "Functions, measures, and equipartitioning k-fans". *Discrete and Computational Geometry*, 2013, to appear.

[5] P. Blagojević and G. Ziegler. "Convex equipartitions by equivariant obstruction theory". arXiv:1202.5504v2, 2012.

[6] R. Karasev. "Equipartition of several measures". arXiv:1011.4762, 2010.

[7] R. Nandakumar and N. Ramana Rao "'Fair' partitions of polygons' - an introduction" *Proc. Indian Academy of Sciences - Mathematical Sciences (to appear); http://arxiv.org/abs/0812*, 2010.

On the nonexistence of k-reptile simplices in $\mathbb{R}^3$ and $\mathbb{R}^4$ *

Jan Kynčl[†1] and Zuzana Safernová[‡1]

[1] Department of Applied Mathematics and Institute for Theoretical Computer Science, Charles University, Faculty of Mathematics and Physics, Malostranské nám. 25, 118 00 Praha 1, Czech Republic

Abstract

A d-dimensional simplex S is called a *k-reptile* (or a *k-reptile simplex*) if it can be tiled without overlaps by k simplices with disjoint interiors that are all mutually congruent and similar to S. For $d = 2$, triangular k-reptiles exist for many values of k and they have been completely characterized by Snover, Waiveris, and Williams. On the other hand, the only k-reptile simplices that are known for $d \geq 3$, have $k = m^d$, where m is a positive integer. We substantially simplify the proof by Matoušek and the second author that for $d = 3$, k-reptile tetrahedra can exist only for $k = m^3$. We also prove a weaker analogue of this result for $d = 4$ by showing that four-dimensional k-reptile simplices can exist only for $k = m^2$.

1 Introduction

A closed set $X \subset \mathbb{R}^d$ with nonempty interior is called a *k-reptile* (or a *k-reptile set*) if there are sets $X_1, X_2, \ldots, X_k$ with disjoint interiors and with $X = X_1 \cup X_2 \cup \cdots \cup X_k$ that are all mutually congruent and similar to X. Such sets have been studied in connection with fractals and also with crystallography and tilings of $\mathbb{R}^d$ [4, 8, 9, 11].

It easy to see that whenever S is a d-dimensional k-reptile simplex, then all of $\mathbb{R}^d$ can be tiled by congruent copies of S: indeed, using the tiling of S by its smaller copies $S_1, \ldots, S_k$ as a pattern, one can inductively tile larger and larger similar copies of S. On the other hand, not all space-filling simplices must be k-reptiles for some $k \geq 2$.

Clearly, every triangle tiles $\mathbb{R}^2$. Moreover, every triangle T is a k-reptile for $k = m^2$, since T can be tiled in a regular way with m^2 congruent tiles, each positively or negatively homothetic to T. See e. g. Snover et al. [19] for an illustration.

The question of characterizing the tetrahedra that tile $\mathbb{R}^3$ is still open and apparently rather difficult. The first systematic study of space-filling tetrahedra was made by Sommerville. Sommerville [20] discovered a list of exactly four tilings (up to isometry and rescaling), but he assumed that all tiles are *properly congruent* (that is, congruent by an orientation-preserving isometry) and meet face-to-face. Edmonds [6] noticed a gap in Sommerville's proof and by completing the analysis, he confirmed that Sommerville's classification of proper, face-to-face tilings is complete. In the non-proper and non face-to-face situations there are infinite families of non-similar tetrahedral tilers. Goldberg [10] described three such families, obtained by partitioning a triangular prism. In fact, Goldberg's first family was found by Sommerville [20] before, but he selected only special cases with a certain symmetry. Goldberg [10] noticed that even the general case admits a proper tiling of $\mathbb{R}^3$. Goldberg's first family also coincides with the family of simplices found by Hill [14], whose aim was to classify *rectifiable* simplices, that is, simplices that can be cut by straight cuts into finitely many pieces that can be rearranged to form a cube. The simplices in Goldberg's second and third families are obtained from the simplices in the first family by splitting into two congruent halves. According to Senechal's survey [17], no other space-filling tetrahedra than those described by Sommerville and Goldberg are known.

For general d, Debrunner [5] constructed $\lfloor d/2 \rfloor + 2$ one-parameter families and a finite number of additional special types of d-dimensional simplices that tile $\mathbb{R}^d$. Smith [18] generalized Goldberg's construction and using Debrunner's ideas, he obtained $(\lfloor d/2 \rfloor + 2)\phi(d)/2$ one-parameter families of space-filling d-dimensional simplices; here $\phi(d)$ is the Euler's totient function. It is not known whether for some d there is an acute space-filling simplex or a two-parameter family of space-filling simplices [18].

In recent years the subject of tilings has received a certain impulse from computer graphics and other computer applications. In fact, our original motivation for studying simplices that are k-reptiles comes from a problem of probabilistic marking of Internet

*The authors were supported by the project CE-ITI (GACR P202/12/G061) of the Czech Science Foundation, by the grant SVV-2013-267313 (Discrete Models and Algorithms) and by project GAUK 52410. The research was partly conducted during the Special Semester on Discrete and Computational Geometry at École Polytechnique Féderale de Lausanne, organized and supported by the CIB (Centre Interfacultaire Bernoulli) and the SNSF (Swiss National Science Foundation).

[†]Email: kyncl@kam.mff.cuni.cz

[‡]Email: zuzka@kam.mff.cuni.cz

packets for IP traceback [1, 2]. See [15] for a brief summary of the ideas of this method. For this application, it would be interesting to find a d-dimensional simplex that is a k-reptile with k as small as possible.

For dimension 2 there are several possible types of k-reptile triangles, and they have been completely classified by Snover et al. [19]. In particular, k-reptile triangles exist for all k of the form a^2+b^2, a^2 or $3a^2$ for arbitrary integers a, b. In contrast, for $d \geq 3$, reptile simplices seem to be much more rare. The only known constructions of higher-dimensional k-reptile simplices have $k = m^d$. The best known examples are the *Hill simplices* (or the *Hadwiger–Hill simplices*) [5, 12, 14]. A d-dimensional Hill simplex is the convex hull of vectors 0, b_1, b_1+b_2, $\ldots$, $b_1+\cdots+b_d$, where $b_1, b_2, \ldots, b_d$ are vectors of equal length such that the angle between every two of them is the same and lies in the interval $(0, \frac{\pi}{2} + \arcsin\frac{1}{d-1})$.

Concerning nonexistence of k-reptile simplices in dimension $d \geq 3$, Hertel [13] proved that a 3-dimensional simplex is an m^3-reptile using a "standard" way of dissection (which we will not define here) if and only if it is a Hill simplex. He conjectured that Hill simplices are the only 3-dimensional reptile simplices. Herman Haverkort recently pointed us to an example of a k-reptile tetrahedron which is not Hill, which contradicts Hertel's conjecture. In fact, except for the one-parameter family of Hill tetrahedra, three other space-filling tetrahedra described by Sommerville [20] and Goldberg [10] are also k-reptiles for every $k = m^3$. The simplices and their tiling are based on the barycentric subdivision of the cube. The construction can be naturally extended to find similar examples of d-dimensional k-reptile simplices for $d \geq 4$ and $k = m^d$. Matoušek [15] showed that there are no 2-reptile simplices of dimension 3 or larger. For dimension $d = 3$ Matoušek and the second author [16] proved the following theorem.

Theorem 1 [16] *In $\mathbb{R}^3$, k-reptile simplices (tetrahedra) exist only for k of the form m^3, where m is a positive integer.*

We give a new, simple proof of Theorem 1 in Section 3.

Matoušek and the second author [16] conjectured that a d-dimensional k-reptile simplex can exist only for k of the form m^d for some positive integer m. We prove a weaker version of this conjecture for four-dimensional simplices.

Theorem 2 *Four-dimensional k-reptile simplices can exist only for k of the form m^2, where m is a positive integer.*

Four-dimensional Hill simplices are examples of k-reptile simplices for $k = m^4$. Whether there exists a four-dimensional m^2-reptile simplex for m non-square remains an open question.

2 Angles in simplices and Coxeter diagrams

Given a d-dimensional simplex S with vertices $v_0, \ldots, v_d$, let F_i be the facet opposite to v_i. A *dihedral angle* $\beta_{i,j}$ of S is the internal angle of the facets F_i and F_j, that meet at the $(d-2)$-face $F_i \cap F_j$.

An *edge-angle* of S is the internal $(d-1)$-dimensional angle incident to an edge and can be represented by a $(d-2)$-dimensional spherical simplex.

The *Coxeter diagram* of S is a graph $c(S)$ with labeled edges such that the vertices of $c(S)$ represent the facets of S and for every pair of facets F_i and F_j, there is an edge $e_{i,j}$ labeled by the dihedral angle $\beta_{i,j}$.

Observation 3 *The edge-angles of a four-dimensional simplex S can be represented by spherical triangles, whose angles are dihedral angles in S. Therefore, an edge-angle in S represented by a spherical triangle with angles α, β, γ corresponds to a triangle in the Coxeter diagram with edges labeled by α, β, γ.* □

The most important tool we use is Debrunner's lemma [5, Lemma 1], which connects the symmetries of a d-simplex with the symmetries of its Coxeter diagram (which represents the "arrangement" of the dihedral angles). This lemma allows us to substantially simplify the proof of Theorem 1 and enables us to step up by one dimension and prove Theorem 2, which seemed unmanageable before.

Lemma 4 (Debrunner's lemma [5]) *Let S be a d-dimensional simplex. The symmetries of S are in one-to-one correspondence with the symmetries of its Coxeter diagram $c(S)$ in the following sense: each symmetry φ of S induces a symmetry Φ of $c(S)$ so that $\varphi(v_i) = v_j \Leftrightarrow \Phi(F_i) = F_j$, and vice versa.*

3 A simple proof of Theorem 1

We proceed as in the original proof, but instead of using the theory of scissor congruence, Jahnel's theorem about values of rational angles and Fiedler's theorem, we only use Debrunner's lemma (Lemma 4).

Assume for contradiction that S is a k-reptile tetrahedron where k is not a third power of a positive integer. A dihedral angle α is called *indivisible* if it cannot be written as a linear combination of other dihedral angles in S with nonnegative integer coefficients.

The following lemmas are proved in [16].

Lemma 5 [16, Lemma 3.1] *If α is an indivisible dihedral angle in S, then the edges of S with dihedral angle α have at least three different lengths.*

Lemma 6 [16, Lemma 3.3] *One of the following two possibilities occur:*

(i) *All the dihedral angles of S are integer multiples of the minimal dihedral angle α, which has the form $\frac{\pi}{n}$ for an integer $n \geq 3$.*

(ii) *There are exactly two distinct dihedral angles β_1 and β_2, each of them occurring three times in S.*

First we exclude case (ii) of Lemma 6. If S has two distinct dihedral angles $\beta_1 \neq \beta_2$, each occurring at three edges, then they can be placed in S in two essentially different ways; see Fig. 1. In both cases, for each $i \in \{1, 2\}$, the Coxeter diagram of S has at least one nontrivial symmetry which swaps two distinct edges with label β_i. By Debrunner's lemma, the corresponding symmetry of S swaps two distinct edges with dihedral angle β_i, which thus have the same length. But then the edges with dihedral angle β_i have at most 2 different lengths and this contradicts Lemma 5, since the smaller of the two angles β_1, β_2 is indivisible.

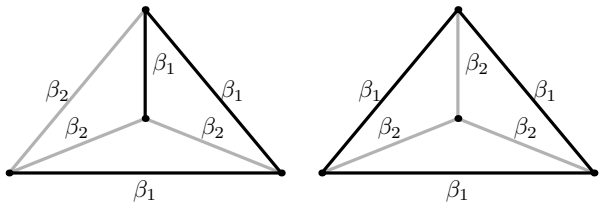


Figure 1: Two possible configurations of two dihedral angles.

Now we exclude case (i) of Lemma 6. Call the edges of S (and of $c(S)$) with dihedral angle α the *α-edges*. Since there are at least three α-edges in S, there is a vertex v of S where two α-edges meet. Let β be the dihedral angle of the third edge incident to v (possibly β can be equal to α).

We have $2\alpha + \beta > \pi$, using a well-known fact that the sum of the three dihedral angles occurring at a vertex of S exceeds π. Writing $\beta = m\alpha = \frac{m}{n} \cdot \pi$, we have $2 \cdot \frac{\pi}{n} + \frac{m}{n} \cdot \pi > \pi$, which implies $m > n - 2$. Since $m < n$, we have $m = n - 1$ and hence $\beta = \pi - \alpha$.

Now we distinguish several cases depending on the subgraph H_α of $c(S)$ formed by the α-edges.

- H_α contains three edges incident to a common vertex (which correspond to a triangle in S). Then all the other edges must have the angle β and we get the configuration as in Fig. 1 (right), which we excluded earlier.

- H_α contains a triangle (the corresponding edges in S meet at a single vertex). Then $\beta = \alpha$, and thus $\alpha = \frac{\pi}{2}$, which contradicts the condition $n \geq 3$ from Lemma 6 (i).

- H_α is a path of length three (this corresponds to a path in S, too). Then two edges have the angle $\beta > \alpha$ and the remaining edge has some angle $\gamma \neq \alpha$. See Figure. 2 (left). The resulting Coxeter diagram has a nontrivial involution swapping two α-edges. By Debrunner's lemma, this contradicts Lemma 5.

- It remains to deal with the case where H_α is a four-cycle (which corresponds to a four-cycle in S). In this case the remaining two edges have dihedral angle β, so the Coxeter diagram has a dihedral symmetry group D_4 acting transitively on the α-edges. By Debrunner's lemma, all the α-edges have the same length. This again contradicts Lemma 5.

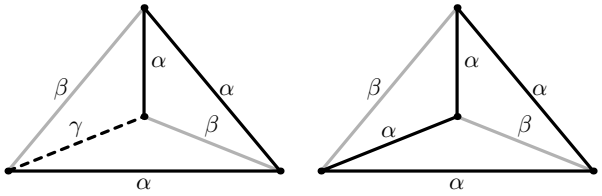


Figure 2: The α-edges form a path (left) or a four-cycle (right) in $c(S)$

We obtained a contradiction in each of the cases, hence the proof of Theorem 1 is finished.

4 The proof of Theorem 2

The method of the proof is similar to the three-dimensional case.

Assume for contradiction that S is a four-dimensional k-reptile simplex where k is not a square of a positive integer. Let $S_1, \ldots, S_k$ be mutually congruent simplices similar to S that tile S. Then each S_i has volume k-times smaller than S, and thus S_i is scaled by the ratio $\rho := k^{-1/4}$ compared to S. For k non-square, ρ is an irrational number of algebraic degree 4 over $\mathbb{Q}$.

Similarly to [16] we define an *indivisible* edge-angle (spherical triangle) as a spherical triangle which cannot be tiled with smaller spherical triangles representing the other edge-angles of S or their mirror images. Clearly, the edge-angle with the smallest surface area is indivisible. We consider a spherical triangle and its mirror image as the same spherical triangle.

We obtain a result similar to Lemma 5: if $\mathcal{T}_0$ is an indivisible edge-angle in S, then the edges of S with edge-angle $\mathcal{T}_0$ have at least four different lengths (and in particular, there are at least four such edges).

The strategy of the proof is now the following. First we exclude the case of two indivisible edge-angles, using only elementary combinatorial arguments and Debrunner's lemma.

Then we consider the case of one indivisible edge-angle. Here we need more involved arguments. We study the problem of tiling spherical triangles by congruent triangular tiles, which might be of independent interest. We give a partial classification of such tilings, which might be possible to extend to a full classification using a reasonable amount of effort. A related question, a classification of edge-to-edge tilings of the sphere by congruent triangles, has been completely solved by Agaoka and Ueno [3].

To rule out several cases, we use a characterization of $\binom{d+1}{2}$-tuples of dihedral angles by Fiedler [7]. The characterization implies, in particular, that if $\beta_{i,j}, i, j = 1, 2, \ldots, d+1$, are the dihedral angles of some d-dimensional simplex, then the matrix $(a_{i,j}; i, j = 1, 2, \ldots, n+1)$, where $a_{i,i} = 0$ and $a_{i,j} = \cos \beta_{i,j}$ for $i \neq j$, is singular.

References

[1] M. Adler, Tradeoffs in probabilistic packet marking for IP traceback, *Proc. 34th Annu. ACM Symposium on Theory of Computing* (2002) 407–418.

[2] M. Adler, J. Edmonds and J. Matoušek, Towards asymptotic optimality in probabilistic packet marking, *Proc. 37th Annu. ACM Symposium on Theory of Computing* (2005) 450–459.

[3] Y. Agaoka and Y. Ueno, Classification of tilings of the 2-dimensional sphere by congruent triangles, *Hiroshima Math. J.* **32**(3) (2002), 463–540.

[4] C. Bandt, Self-similar sets. V. Integer matrices and fractal tilings of $\mathbf{R}^n$, *Proc. Amer. Math. Soc.* **112**(2) (1991), 549–562.

[5] H. E. Debrunner, Tiling Euclidean d-space with congruent simplexes, *Discrete geometry and convexity (New York, 1982)*, vol. 440 of *Ann. New York Acad. Sci.*, New York Acad. Sci., New York (1985) 230–261.

[6] A. L. Edmonds, Sommerville's missing tetrahedra, *Discrete Comput. Geom.* **37**(2) (2007), 287–296.

[7] M. Fiedler, Geometry of the simplex in E_n (in Czech, with English and Russian summary), *Časopis pro pěstování matematiky* **79** (1954), 297–320.

[8] M. Gardner, *The unexpected hanging and other mathematical diversions*, The University of Chicago Press (1991).

[9] G. Gelbrich, Crystallographic reptiles, *Geom. Dedicata* **51**(3) (1994), 235–256.

[10] M. Goldberg, Three infinite families of tetrahedral space-fillers, *J. Comb. Theory, Ser. A* **16** (1974), 348–354.

[11] S. W. Golomb, Replicating figures in the plane, *Mathematical Gazette* **48** (1964), 403–412.

[12] H. Hadwiger, Hillsche Hypertetraeder (in German), *Gaz. Mat. (Lisboa)* **12**(50) (1951), 47–48.

[13] E. Hertel, Self-similar simplices, *Beiträge Algebra Geom.* **41**(2) (2000), 589–595.

[14] M. J. M. Hill, Determination of the volumes of certain species of tetrahedra without employment of the method of limits, *Proc. London Math. Soc.* **27** (1896), 39–52.

[15] J. Matoušek, Nonexistence of 2-reptile simplices, *Discrete and Computational Geometry: Japanese Conference, JCDCG 2004*, Lecture Notes in Computer Science 3742, Springer, Berlin (2005) 151–160, erratum at `http://kam.mff.cuni.cz/~matousek/no2r-err.pdf`.

[16] J. Matoušek and Z. Safernová, On the nonexistence of k-reptile tetrahedra, *Discrete Comput. Geom.* **46**(3) (2011), 599–609.

[17] M. Senechal, Which tetrahedra fill space?, *Math. Magazine* **54**(5) (1981), 227–243.

[18] W. D. Smith, Pythagorean triples, rational angles, and space-filling simplices (2003), manuscript, `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.124.6579`.

[19] S. L. Snover, C. Waiveris and J. K. Williams, Rep-tiling for triangles, *Discrete Math.* **91**(2) (1991), 193–200.

[20] D. M. Y. Sommerville, Division of space by congruent triangles and tetrahedra, *Proc. Roy. Soc. Edinburgh* **4** (1923), 85–116.

Drawing the double circle on a grid of minimum size

S. Bereg*[1], R. Fabila-Monroy†[2], D. Flores-Peñaloza‡[3], M.A. Lopez§[4], and P. Pérez-Lantero¶[5]

[1]Department of Computer Science, University of Texas at Dallas, USA.
[2]CINVESTAV, Instituto Politécnico Nacional, Mexico.
[3]Departamento de Matemáticas, Facultad de Ciencias, UNAM, Mexico.
[4]Department of Computer Science, University of Denver, USA.
[5]Escuela de Ingeniería Civil en Informática, Universidad de Valparaíso, Chile.

Abstract

In 1926, Jarník introduced the problem of drawing a convex n-gon with vertices having integer coordinates. He constructed such a drawing in the grid $[1, c \cdot n^{3/2}]^2$ for some constant $c > 0$, and showed that this grid size is optimal up to a constant factor. We consider the analogous problem of drawing the double circle, and prove that it can be done within the same grid size. Moreover, we give an $O(n \log n)$-time algorithm to construct such a point set.

1 Introduction

Given $n \geq 3$, a *double circle* is a set $P = \{p_0, p_1, \ldots, p_{n-1}, p'_0, p'_1, \ldots, p'_{n-1}\}$ of $2n$ planar points in general position such that: (1) $p_0, p_1, \ldots, p_{n-1}$ are precisely the vertices of the convex hull of P labelled in counterclokwise order around the boundary; (2) point p'_i is close to the segment joining p_i with p_{i+1}; (3) the line passing through p_i and p'_i separates p_{i+1} from P; and (4) the line passing through p'_i and p_{i+1} separates p_i from P (see Figure 1). Subindices are taken modulo n. The double circle has been considered in combinatorial geometry and it is conjectured to have the least number of triangulations [1, 2].

Drawing an n-vertex convex polygon with integer vertices can be easily done by considering the n points $(1, 1), (2, 4), (3, 9), \ldots, (n, n^2)$ as the vertices of the polygon. In this case the *size* of the integer point set is equal to $n^2 - 1 = \Theta(n^2)$, where size refers to the smallest N such that the point set can be translated to lie in the grid $[0, N]^2$. In 1926, Jarník [5] showed how to draw an n-vertex convex polygon with

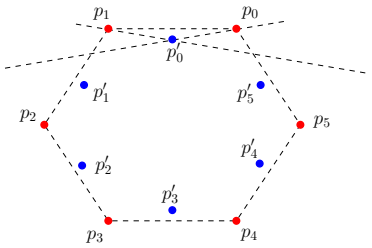


Figure 1: A double circle of twelve points.

size $N = O(n^{3/2})$ and proved that this bound is optimal. In recent years the so-called Jarník polygons and extensions of them have been studied [3, 6].

Given any integer point (i, j), we say that (i, j) is *visible* (from the origin) if the interior of the line segment joining the origin and (i, j) contains no lattice points. Observe that (i, j) is visible if and only if $\gcd(i, j) = 1$, where $\gcd(i, j)$ denotes the greatest common divisor of i and j. We consider points as vectors as well, and vice versa. A Jarník polygon is formed by choosing a natural number Q, and taking the set V_Q of visible vectors (i, j) such that $\max\{|i|, |j|\} \leq Q$ [4, 5, 6]. The polygon is then the unique (up to translation) convex polygon whose edges, viewed as vectors, are precisely the elements of V_Q, that is, the vertices can be obtained by starting from an arbitrary point and adding the vectors of V_Q, one by one, in counterclockwise order, to the previously computed vertex (see Figure 2).

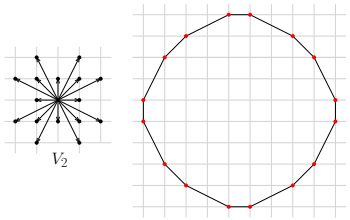


Figure 2: A Jarník polygon (right) and its generating vectors V_2 (left).

We study how to draw a $2n$-point double circle with integer points using the smallest size N. We present

*Email: besp@utdallas.edu.
†Email: ruyfabila@math.cinvestav.edu.mx. Partially supported by grant 153984 (CONACyT, Mexico).
‡Email: dflorespenaloza@gmail.com. Partially supported by grants 168277 (CONACyT, Mexico) and IA102513 (PAPIIT, UNAM, Mexico).
§Email: mlopez@du.edu.
¶Email: pablo.perez@uv.cl. Partially supported by grant CONICYT, FONDECYT/Iniciación 11110069 (Chile).

an $O(n \log n)$-time algorithm that correctly constructs the double circle with size within $O(n^{3/2})$, where that bound is also optimal. In Section 2 we show our algorithm, and in Section 3 its correctness is proved. Finally, in Section 4, we state future work.

2 Double circle construction

Observe that a simple construction with quadratic size is as follows: Consider the function $f(x) = x^2+x$. For $i = 1, \ldots, 2n-1$, add the point $(i, f(i))$ if i is odd, and the point $(i, f(i)+2)$ otherwise. The final point is $(n, \frac{f(2n-1)+f(1)}{2} - 1) = (n, 2n^2 - n)$, i.e., the point just below the midpoint of the segment connecting $(1, f(1))$ and $(2n-1, f(2n-1))$. The size of the resulting point set is $N = f(2n-1) - f(1) = (2n-1)^2 + (2n-1) - 2 = 4n^2 - 2n - 2 = \Theta(n^2)$.

We say that a sequence V of vectors is *symmetric* if V contains an even number of vectors sorted counterclockwise around the origin, and for every vector a in V its opposite vector $-a$ is also in V. Observe that any sequence of vectors defining a Jarník polygon is symmetric. For any sequence $V = [v_1, v_2, \ldots, v_{2t}]$ of $2t$ vectors let the point set $\mathcal{P}(V) := \{p_1, p_2, \ldots, p_{2t}\}$, where $p_1 = v_1$ and $p_i = p_{i-1} + v_i$ for $i = 2, \ldots, 2t$. Note that if we sort the elements of V_Q around the origin then the elements of $\mathcal{P}(V_Q)$ are the vertices of the Jarník polygon. Furthermore, if V is symmetric then the elements of $\mathcal{P}(V)$ are in convex position. Let sequence $\text{alt}(V) := [v_2, v_1, v_4, v_3, \ldots, v_{2t}, v_{2t-1}]$ (see Figure 3 for an example with $t = 8$). For any scalar λ let the sequence $\lambda V := [\lambda v_1, \lambda v_2, \ldots, \lambda v_{2t}]$.

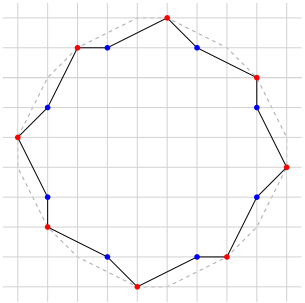


Figure 3: $\mathcal{P}(\text{alt}(V_4))$.

The idea is to generate a suitable symmetric sequence V of $2n$ vectors and then build the point set $\mathcal{P}(\text{alt}(V))$ as the double circle point set, up to some transformation of the elements of $\text{alt}(V)$. A (not optimal) example is $V = [(1,1), (1,2), \ldots, (1,n), (-1,-1), (-2,-2), \ldots, (-1,n)]$ for even $n \geq 4$. The point set $\mathcal{P}(\text{alt}(V))$ is in fact a double circle but its size is equal to $1 + 2 + \ldots + n = \Theta(n^2)$ (see Figure 4).

The construction in which the resulting point set is a double circle of size $O(n^{3/2})$ is based on the next

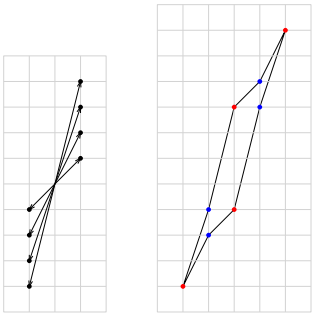


Figure 4: A naive construction for $n = 4$ showing both vectors (left) and the resulting point set (right).

two algorithms:

VisibleVectors(n): With input $n \geq 3$, the symmetric sequence V of $2n$ visible vectors, sorted counterclockwise around the origin, is generated so as to satisfy the next two invariants. Let $\text{B}_t := \{p \in \mathbb{Z}^2 : \|p\|_1 \leq t\}$, $k := \max_{v \in V} \|v\|_1$, and (even) s be the number of visible vectors of B_{k-1}: (i) all visible vectors of B_{k-1} are in V, and (ii) the other elements of V are generated as follows, until $2n-s$ elements are obtained: for $i = 1, \ldots, k-1$ generate vectors $(i, k-i)$, $(-i, -(k-i))$, $(-i, k-i)$, $(i, -(k-i))$ in this order, if and only if $\gcd(i, k-i) = 1$. Refer to Algorithm 2.1 for a pseudo-code.

BuildDoubleCircle(n): With input $n \geq 3$, build a $2n$-point double circle. First, set sequences V :=VisibleVectors(n) and $[v'_1, v'_2, \ldots, v'_{2n}] := \text{alt}(V)$. Then, the sequence $W = [w_1, w_2, \ldots, w_{2n}]$ of $2n$ vectors is created as follows: for $i = 1, 3, \ldots, 2n-1$ set $w_i = (1-\lambda)v'_i + \lambda v'_{i+1}$ and $w_{i+1} = \lambda v'_i + (1-\lambda)v'_{i+1}$, where $\lambda = 1/3$. Finally, build the $2n$-point set $\mathcal{P}((1/\lambda)W)$ as the double circle.

Algorithm 2.1: VisibleVectors(n)

$k \leftarrow 1, V \leftarrow [(1,0), (-1,0), (0,1), (0,-1)]$
repeat
  $k \leftarrow k+1$
  for $i \leftarrow 1$ **to** $k-1$
    do $j \leftarrow k-i$
      if gcd$(i,j) = 1$
        then **if** length$(V) < 2n$
          then $V \leftarrow V + [(i,j), (-i,-j)]$
        if length$(V) < 2n$
          then $V \leftarrow V + [(-i,j), (i,-j)]$
until length$(V) = 2n$
Sort V counterclockwise around origin
return (V)

3 Construction correctness

Let $V = [v_1, v_2, \ldots, v_{2n}]$ be the (circular) sequence of vectors obtained by executing VisibleVectors(n),

for $n \geq 3$. For every $i = 1, 3, 5, \ldots, 2n-1$ we say that the pair of vectors v_i, v_{i+1} is a pair of alt(V).

Lemma 1 (Chapter 2 of [4]) *Given a natural number Q, the number $|V_Q|$ of vertices of the Jarník polygon is equal to*

$$4 + 4 \sum_{\substack{i=1 \\ gcd(i,j)=1}}^{Q} \sum_{j=1}^{Q} 1 = \frac{24Q^2}{\pi^2} + O(Q \log Q).$$

The size $S(Q)$ of the Jarník polygon is equal to

$$\begin{aligned} 1 + 2 \sum_{\substack{i=1 \\ gcd(i,j)=1}}^{Q} \sum_{j=1}^{Q} i &= 1 + 2 \sum_{\substack{i=1 \\ gcd(i,j)=1}}^{Q} \sum_{j=1}^{Q} j \\ &= \frac{6Q^3}{\pi^2} + O(Q^2 \log Q). \end{aligned}$$

Lemma 2 *V is symmetric and point set $\mathcal{P}(V)$ has size $O(n^{3/2})$.*

Proof. Observe that for every vector a in V, $-a$ is also in V since in algorithm VisibleVectors the vectors are added to sequence V in pairs, and each pair consists of two opposite vectors. Then V becomes symmetric once the elements of V are sorted counterclockwise around the origin. On the other hand $V_{\lfloor \frac{k-1}{2} \rfloor} \subset V \subset V_k$, where $k = \max_{v \in V} \|v\|_1$. Then we have $|V_{\lfloor \frac{k-1}{2} \rfloor}| \leq 2n \leq |V_k|$, which implies $k = \Theta(\sqrt{n})$ by Lemma 1. By the same lemma we obtain:

$$\begin{aligned} \sum_{i=1}^{n} x(v_i), \sum_{i=1}^{n} y(v_i) &< 1 + 2 \sum_{\substack{i=1 \\ gcd(i,j)=1}}^{k} \sum_{j=1}^{k} i \\ &= S(k) \\ &= \Theta(k^3) = \Theta(n^{3/2}) \end{aligned}$$

Hence, the size of $\mathcal{P}(V)$ is $O(n^{3/2})$. □

Let o denote the origin of coordinates. Given two points p, q let $\ell(p, q)$ denote the line passing through p and q and *directed* from p to q, and pq denote the segment joining p and q. Given three points $p = (x_p, y_p)$, $q = (x_q, y_q)$, and $r = (x_r, y_r)$, let $\Delta(p, q, r)$ denote the triangle with vertices at p, q, and r; $A(p, q, r)$ denote de area of $\Delta(p, q, r)$; and turn(p, q, r) denote the so-called *geometric turn* (going from p to r passing through q) where

$$\text{turn}(p, q, r) = \begin{vmatrix} x_p & y_p & 1 \\ x_q & y_q & 1 \\ x_r & y_r & 1 \end{vmatrix}$$

and $A(p, q, r) = \frac{1}{2} |\text{turn}(p, q, r)|$. Extending this notation, let $\Delta(p, q) := \Delta(o, p, q)$, $A(p, q) := A(o, p, q)$, and turn$(p, q) :=$ turn(o, p, q). We use the so-called Pick's theorem:

Theorem 3 (Pick's theorem [7]) *The area of any simple polygon H with lattice vertices is equal to $i + b/2 - 1$, where i and b are the numbers of lattice points in the interior and the boundary of H, respectively.*

Lemma 4 *For every two consecutive vectors a_1, a_2 of V we have $A(a_1, a_2) = 1/2$.*

Proof. Suppose $\Delta(a_1, a_2)$ contains a lattice point p different from o, a_1, and a_2. Then p cannot belong to segments oa_1 and oa_2, and segment op contains a visible point q (possibly equal to p). If $\|q\|_1 < \max\{\|a_1\|_1, \|a_2\|_1\}$ then q must belong to V by invariant (i) of algorithm VisibleVectors. Otherwise, we have $\|q\|_1 = \max\{\|a_1\|_1, \|a_2\|_1\}$. Suppose w.l.o.g. that $\|a_1\|_1 < \|a_2\|_1$, and let the point q' denote the intersection of $\ell(o, q)$ with the segment s connecting a_1 to a_2. Observe that $q' = \delta a_1 + (1 - \delta)a_2$ for some $\delta \in (0, 1)$, and further that $\|q\|_1 \leq \|q'\|_1 = \|\delta a_1 + (1 - \delta)a_2\|_1 \leq \delta \|a_1\|_1 + (1 - \delta)\|a_2\|_2 < \|a_2\|_1$, which is a contradiction. Then we must have that $\|q\|_1 = \|a_1\|_1 = \|a_2\|_1$, which implies that q, a_1, a_2 belong to a same quadrant since in this case q is at the interior of the segment s. Therefore, q must belong to V by invariant (ii) of algorithm VisibleVectors. In both cases, the fact that q belongs to V contradicts the fact that a_1 and a_2 are consecutive vectors of V. Hence $A(a_1, a_2) = 1/2$ by Pick's theorem. □

Let $\lambda \in (0, 1/2)$. Given a pair a, b of vectors let $h(\lambda, a, b) := (1 - \lambda)a + \lambda b$ (see Figure 5).

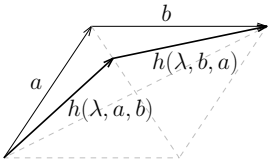


Figure 5: Two vectors a and b, and the vectors $h(\lambda, a, b)$ and $h(\lambda, b, a)$.

Lemma 5 *Let a_1, a_2, a_3, a_4 be four consecutive vectors of V such that a_1, a_2 and a_3, a_4 are pairs of* alt(V). *Let $\lambda \in (0, 1/2)$, $q_1 = h(\lambda, a_2, a_1)$, $q_2 = q_1 + h(\lambda, a_1, a_2)$, $q_3 = q_2 + h(\lambda, a_4, a_3)$, and $q_4 = q_3 + h(\lambda, a_3, a_4)$. Then q_2 is to the right of $\ell(o, q_1)$ and both q_3 and q_4 are to the left of $\ell(o, q_1)$.*

Proof. (Refer to Figure 6.) We have turn$(q_1, q_2) =$ turn$(q_1, q_1 + h(\lambda, a_1, a_2)) =$ turn$((1 - \lambda)a_2 + \lambda a_1, a_1 + a_2) = (1 - \lambda)$ turn$(a_2, a_1) + \lambda$ turn$(a_1, a_2) = 2(2\lambda - 1)A(a_1, a_2) < 0$, which implies that q_2 is to the right of the line $\ell(o, q_1)$. On the other hand:

$$\begin{aligned} & \text{turn}(q_1, q_3) \\ = \; & \text{turn}(h(\lambda, a_2, a_1), h(\lambda, a_2, a_1) + h(\lambda, a_1, a_2) + \\ & h(\lambda, a_4, a_3)) \\ = \; & \text{turn}(h(\lambda, a_2, a_1), h(\lambda, a_1, a_2)) + \\ & \text{turn}(h(\lambda, a_2, a_1), h(\lambda, a_4, a_3)) \end{aligned}$$

$$
\begin{aligned}
&= \operatorname{turn}((1-\lambda)a_2+\lambda a_1,(1-\lambda)a_1+\lambda a_2))+\\
&\quad \operatorname{turn}((1-\lambda)a_2+\lambda a_1,(1-\lambda)a_4+\lambda a_3))\\
&= (1-\lambda)^2\operatorname{turn}(a_2,a_1)+\lambda^2\operatorname{turn}(a_1,a_2)+\\
&\quad (1-\lambda)^2\operatorname{turn}(a_2,a_4)+\lambda(1-\lambda)\operatorname{turn}(a_2,a_3)+\\
&\quad \lambda(1-\lambda)\operatorname{turn}(a_1,a_4)+\lambda^2\operatorname{turn}(a_1,a_3)\\
&= 2\Big((2\lambda-1)A(a_1,a_2)+(1-\lambda)^2A(a_2,a_4)+\\
&\quad \lambda(1-\lambda)A(a_2,a_3)+\lambda(1-\lambda)A(a_1,a_4)+\\
&\quad \lambda^2A(a_1,a_3)\Big)\\
&= 2\Big(\frac{1}{2}(2\lambda-1)+(1-\lambda)^2A(a_2,a_4)+ \qquad (1)\\
&\quad \lambda(1-\lambda)A(a_2,a_3)+\lambda(1-\lambda)A(a_1,a_4)+\\
&\quad \lambda^2A(a_1,a_3)\Big)\\
&\geq (2\lambda-1)+(1-\lambda)^2+\lambda(1-\lambda)+ \qquad (2)\\
&\quad \lambda(1-\lambda)+\lambda^2\\
&= 2\lambda>0
\end{aligned}
$$

where equation (1) follows from Lemma 4 and equation (2) follows from the fact that by Pick's theorem the area of any non-empty triangle with lattice vertices is at least $1/2$. Therefore, q_3 is to the left of $\ell(o,q_1)$. Similarly, since we have that $\operatorname{turn}(a_i,a_j)>0$ $(i=1,2;\ j=3,4)$ then $\operatorname{turn}(h(\lambda,a_2,a_1),h(\lambda,a_3,a_4))>0$, which implies that q_4 is to the left of $\ell(o,q_1)$ given that q_3 is to the left of $\ell(o,q_1)$. By symmetry, it can be proved that $\operatorname{turn}(q_4,q_3,q_1)<0$ and $\operatorname{turn}(q_4,q_3,o)<0$, implying that both q_1 and o are to the right of $\ell(q_4,q_3)$. □

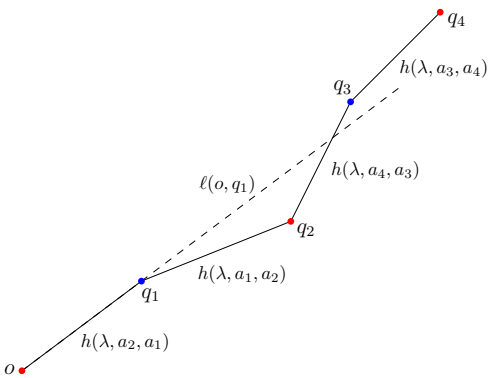


Figure 6: Proof of Lemma 5.

Theorem 6 *There is an $O(n\log n)$-time algorithm that for all $n\geq 3$ builds a double circle of $2n$ points in the grid $[0,N]^2$ where $N=O(n^{3/2})$.*

Proof. Execute the algorithm BuildDoubleCircle with input n, being V the result of calling VisiblePoints(n), building the point set P of $2n$ points. Observe that $\lambda=1/3$ implies that point $w_i/\lambda=3w_i$ is integer for $i=1\ldots 2n$, and then all elements of P are integer points. By Lemma 5 point set P is a double circle. The size of $\mathcal{P}(V)$ is $O(n^{3/2})$ by Lemma 2, and since all elements of P belong to the polygon with vertices $\mathcal{P}(3V)$ the size N of P is also $O(n^{3/2})$. Finally, translate P to lie in the grid $[0,N]^2$. In algorithm VisiblePoints the time complexity is dominated by: (1) computing $\gcd(i,j)$ for $O((\sqrt{n})^2)=O(n)$ pairs i,j; and (2) sorting vectors V counterclockwise around the origin. In Case (1) the time complexity is $O(n\log n)$ since $\gcd(i,j)$ consumes $O(\log(\min\{i,j\}))=O(\log\sqrt{n})=O(\log n)$ time. Case (2) consumes $O(n\log n)$ time as well. Since the time complexity of VisiblePoints dominates the time complexity of the main algorithm BuildDoubleCircle, the result follows. □

4 Future work

We are working on extending these results to build other known point sets in integer points of small size, such as the double convex chain, the Horton set, and others. We plan to eventually release a software library supporting many of these constructions.

Acknowledgements

The problem studied here were introduced and partially solved during a visit to University of Valparaiso funded by project CONICYT Fondecyt/Iniciación 11110069 (Chile). The authors would like to thank anonymous referees for their valuable comments.

References

[1] O. Aichholzer, F. Hurtado, and M. Noy. A lower bound on the number of triangulations of planar point sets. *Computational Geometry*, 29(2):135–145, 2004.

[2] O. Aichholzer, D. Orden, F. Santos, and B. Speckmann. On the number of pseudo-triangulations of certain point sets. *Journal of Combinatorial Theory, Series A*, 115(2):254–278, 2008.

[3] I. Bárány and N. Enriquez. Jarník's convex lattice n-gon for non-symmetric norms. *Mathematische Zeitschrift*, 270:627–643, 2012.

[4] M. N. Huxley. *Area, lattice points, and exponential sums.* Oxford Science Publications. The Clarendon Press Oxford University Press, New York, 1996.

[5] V. Jarník. Über die Gitterpunkte auf konvexen Kurven. *Mathematische Zeitschrift*, 24:500–518, 1926.

[6] G. Martin. The Limiting Curve of Jarník's Polygons. *Transactions of the American Mathematical Society*, 355(12):4865–4880, 2003.

[7] G. Pick. Geometrisches zur Zahlenlehre. *Sitzungsberichte des Deutschen Naturwissenschaftlich-Medicinischen Vereines für Böhmen "Lotos" in Prag.*, 19:311–319, 1899.

SensoGraph: Using proximity graphs for sensory analysis

David N. de Miguel*[1], David Orden†[2], Encarnación Fernández-Fernández‡[3], José M. Rodríguez-Nogales §[3], and Josefina Vila-Crespo ¶[4]

[1]Universidad de Alcalá, Spain.
[2]Departamento de Física y Matemáticas, Universidad de Alcalá, Spain.
[3]Departamento de Ingeniería Agrícola y Forestal, Universidad de Valladolid, Spain.
[4]Departamento de Anatomía Patológica, Medicina Preventiva y Salud Pública, Medicina Legal y Forense, Universidad de Valladolid, Spain.

Abstract

Sensory evaluation of foods is as important as chemical, physical or microbiological examinations, being specially relevant in food industries. Classical methods can be long and costly, making them less suitable for certain industries like the wine industry. Some alternatives have arisen recently, including Napping®, where the tasters represent the sensory distances between products by positioning them on a tablecloth; the more similar they perceive the products, the closer they position them on the tablecloth. This method uses multiple factor analysis (MFA) to process the data collected. The present paper introduces the software `SensoGraph`, which makes use of proximity graphs to analyze those data. The application is described and experimental results are presented in order to compare the performances of `SensoGraph` and Napping®, using eight wines from the Toro region and two groups of twelve tasters with different expertise.

1 Sensory analysis

The goal of sensory evaluation of foods is the study of the sensations they produce. When consuming a food, stimulus of several classes are perceived; visual (color, shape, brightness), tactile (at fingertips or mouth epithelium), odorous (at nose epithelium), gustatory (at taste buds), and even auditory (e.g., for crunchy food). The norm ISO 5492:2008 defines sensory analysis as the science related to the evaluation of organoleptic attributes of a product by the senses, and other ISO norms unify the tools and methods used for that evaluation. The sensory examination turns out to be as important as chemical, physical or microbiological examinations, being specially relevant in food industries.

The main tool in sensory analysis is a panel of tasters, either experts or consumers, who evaluate the products from an analytic and/or hedonic point of view. As any other instrument depends on its calibration, such a panel depends on human beings. Their perceptions are translated into quantifiable data, which is then treated by means of different methods.

The classical method is descriptive analysis, which aims to describe the sensory characteristics of a product and use them to quantify the sensory differences between products [11]. Different implementations of this method provide a quantitative description of the sensory attributes perceived by a group of expert tasters, chosen because of their sensory abilities, who are trained to describe and evaluate sensory differences among products. Such a training is a critical step in the creation of an expert panel of tasters, when tasters agree on the definitions of descriptors and the use of scales, in order to provide reliable and consistent results.

However, this training can be long and costly, making it less suitable for certain industries like the wine industry. There, sensory characterization is usually performed by the oenologist in charge of the winery, for whom it is difficult to enrol in a panel requiring a regular activity during a long time. Thus, in the last years several alternative methods have been proposed, aiming to provide a fast sensory positioning of a set of products, in order to avoid the most time-consuming steps in classical methods. A prominent one among these alternatives is Napping® [12]. In a single session, all the products are provided simultaneously to the tasters, who represent the sensory distances between products by positioning them on a tablecloth. Products which are perceived as similar should be positioned close to each other, while products perceived as different should be positioned far enough. Each

*Email: david.n.demiguel@gmail.com.
†Email: david.orden@uah.es. Research partially supported by MICINN Project MTM2011-22792, ESF EUROCORES programme EuroGIGA - ComPoSe IP04 - MICINN Project EUI-EURC-2011-4306 and Junta de Castilla y León Project VA172A12-2.
‡Email: effernan@iaf.uva.es.
§Email: rjosem@iaf.uva.es.
¶Email: jvila@pat.uva.es.

taster chooses the criteria and the relative importance given. These data are later processed using multiple factor analysis (MFA) [4], in order to take into account the criteria and relative importance of all the tasters. Despite having both advantages and disadvantages, Napping® has become a useful tool when some accuracy can be sacrificed for the sake of a faster study [13].

2 Proximity graphs

Given a set of points in the plane, a (geometric) *proximity graph* connects two of them according to a chosen proximity criterion. These graphs have been widely used in order to analyze the relative position of points, for instance looking for clusters or spanning structures. See [1, 7] and the references therein. Thus, it seems natural to use them in order to analyze the data collected by a tablecloth method for sensory analysis, like Napping®. Among the many different types of proximity graphs, we have chosen the following:

Nearest Neighbor Graph (NNG): Each point is joined to the closest among the remaining points [14].

k-Nearest Neighbor Graph (k-NNG): In this generalization of the NNG, each point is joined to the k closest among the remaining points.

Minimum Spanning Tree (MST): Among the trees passing through all the given points, the MST is the one which minimizes the sum of edge lenghts [10].

Relative Neighborhood Graph (RNG): This graph, introduced by Toussaint [15], joins two points P, Q if there is no point whose distances to both P and Q are smaller than the distance $d(P, Q)$.

k-Relative Neighborhood Graph (k-RNG): The generalization of the RNG which allows up to k points with distances to both P and Q smaller than $d(P, Q)$.

Gabriel Graph (GG): In this graph two points P, Q are joined if there is no other point inside the closed disk which has the segment $\overline{PQ}$ as diameter [5].

k-Gabriel Graph (k-GG): Generalization of the GG allowing up to k points to lie inside the closed disk.

Delaunay Triangulation (DT): Three points P, Q, R form a triangle precisely if their circumcircle does not contain any other point [3].

k-Delaunay Triangulation (k-DT): Generalization of the DT allowing up to k points to lie inside the closed disk.

β-skeleton (β-SK): A family of proximity graphs, one for each value of $\beta \geq 0$, see [9] for more details. For $\beta = 1$ we get the GG. For $\beta = 2$ we get the RNG.

Unit Disk Graph (UDG): In this graph two points P, Q are joined if the distance $d(P, Q)$ between them is no greater than a fixed threshold [2].

3 The SensoGraph application

SensoGraph is an application, still under development, which aims to use proximity graphs for the analysis of data collected from tablecloth sensory methods like Napping®. The interface intends to be intuitive and easy to use, so that no special knowledge is needed.

Tasting data are stored in the form of an $m \times 2n$ matrix, in which there is a row per product and a pair of columns per taster, storing the two coordinates assigned to the corresponding product. In a first screen, this matrix can be manually created or inserted from a CSV file, according to the IETF RFC 4180 standard. After insertion, the matrix can be modified by adding or removing rows or columns, as well as by editing an individual entry. See Figure 1.

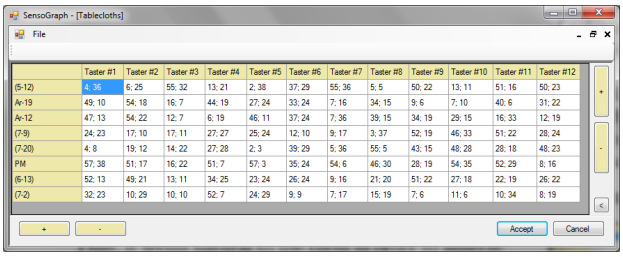


Figure 1: Matrix of tasting data.

After accepting the data matrix, a new screen is shown. There, the user can choose a type of proximity graph among the ones specified in Section 2. For those proximity graphs depending on a parameter, k-NNG, k-RNG, k-GG, k-DT, β-SK, and UDG, the user can change the value of the parameter. Furthermore, the application allows to intersect any of the graphs considered with the UDG, in case the user wants to avoid too long edges.

For the given choice of a type of proximity graph, the application generates the graph for each of the taster's tablecloths. The user can choose a taster and check its tablecloth and the resulting graph. When visualizing a tablecloth, it is also possible to change the type of proximity graph, in order to check the differences between them. See Figure 2.

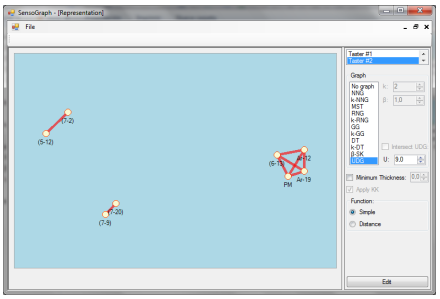


Figure 2: Tablecloth for taster number 2 with the UDG for radius 10.

Furthermore, chosen a type of proximity graph, the user can also merge all the tablecloths and their corresponding graphs into a single, global, picture. There, every edge is shown with a thickness, computed according to a *simple* function: The thickness corresponds to the number of tasters for which that edge appears in the proximity graph. This representation encodes the global opinion of the panel of tasters, so that those products perceived as similar are joined by thicker edges, while products considered different are joined by thinner edges (or even not joined at all).

In order to position the vertices of this global picture, `SensoGraph` considers the edges like springs which try to approach their endpoints, with a strength proportional to the edge thickness. Thus, vertices joined by thicker edges become closer than those joined by thinner edges. In this new picture, the products considered similar by the panel of tasters can be recognized not only by the thickness of the edge between them, but also by their mutual distance, see Figures 4, 6, and 8. Such a positioning is performed by a slight adaptation of the algorithm by Kamada and Kawai [8].

Since some types of proximity graphs insist in joining vertices which are far apart, `SensoGraph` offers a second way of computing the thickness of an edge. The *distance* function also looks only at the edges appearing in the proximity graph chosen but, in addition, it takes into account their length, decreasing the contribution of long edges to the total thickness. As mentioned above, the user can also choose to discard too long edges, by intersecting any of the proximity graphs with the UDG.

Furthermore, the user can also peel the graph in the global picture, by removing edges below any chosen thickness, in order to keep only the most relevant ones.

4 Experimental results

Eight wines from the Toro region, elaborated using different yeasts during the alcoholic fermentation, were considered. Two panels, of twelve non-trained tasters each, were selected. The *experts* panel was composed by people, mainly young, with some knowledge of the techniques for sensory analysis of wines. The *non-experts* panel was composed by plain consumers, with different ages and levels of knowledge. Each of the panels performed a session of Napping®, and the experts panel repeated for a second session, in order to check for improvements due to such a slight training.

The data from those three sessions was then processed both by multiple factor analysis (MFA) [6], as usual in Napping®, and by `SensoGraph`. Figures 3 to 8 show the results obtained, with `SensoGraph` using GG and the *simple* thickness function.

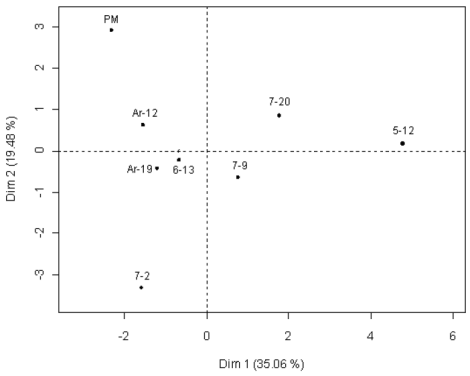


Figure 3: Experts panel, Napping®.

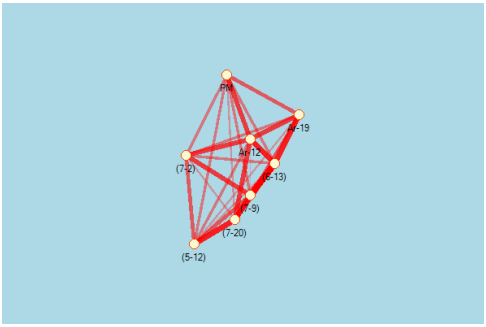


Figure 4: Experts panel, `SensoGraph`.

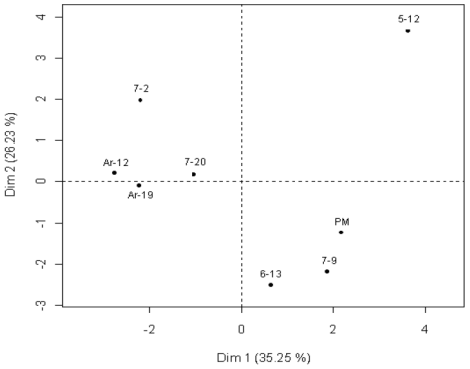


Figure 5: Repetition of experts panel, Napping®.

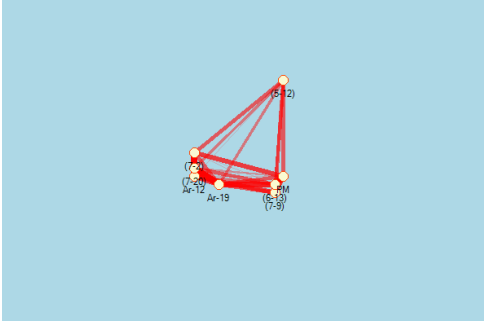


Figure 6: Repetition of experts panel, `SensoGraph`.

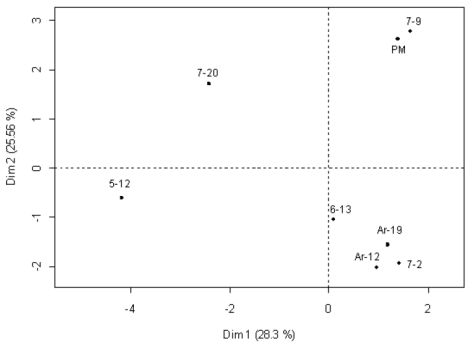


Figure 7: Non-experts panel, Napping®.

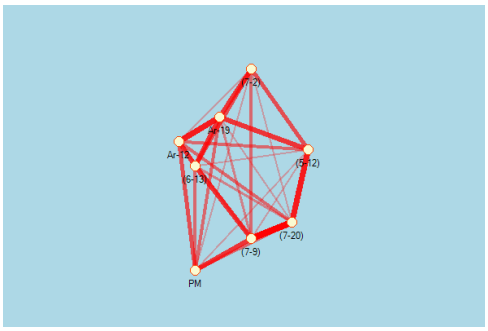


Figure 8: Non-experts panel, `SensoGraph`.

Among the three sessions performed, the most representative results of Napping® were those from the repetition of the experts panel. Figure 5 shows a 35.25% of the variation explained by the first dimension and a 26.23% of the remaining variation explained by the second dimension, for a total inertia of 61.48%. Being this the session for which Napping® is most reliable, it is the one chosen to compare with the results provided by `SensoGraph`.

For that session, using `SensoGraph` with the *simple* thickness function provides the same clusters as Napping® for all the graphs considered except for NNG, which has too few edges, and for k-DT, which has too many edges. Using the *distance* thickness function does not change the elements in the clusters, although the whole picture appears expanded and highlights only the strongest connections between different clusters. Furthermore, it improves the results for the extreme cases above, leading to the same clusters as Napping® for k-DT and palliating the differences for NNG.

A possible advantage of `SensoGraph` is giving other kind of information than Napping®. Apart from providing several types of proximity graphs and parameters to test with, `SensoGraph` shows how the different clusters are connected. For an example, Figures 5 and 6 lead to the same three clusters, but only the one from `SensoGraph` shows that they are actually quite connected, reflecting the fact that all the wines considered were actually quite similar [6].

5 Acknowledgements

The authors want to gratefully thank Ferran Hurtado for proposing them the use of proximity graphs in order to analyze data from tablecloth methods.

References

[1] J. Cardinal, S. Collette, and S. Langerman, Empty region graphs, *Computational Geometry: Theory and Applications*, **42** (2009), 183–195.

[2] B. N. Clark, C. J. Colbourn, and D. S. Johnson, Unit Disk Graphs, *Discrete Mathematics*, **86** (1990), 165–177.

[3] B. N. Delaunay, Sur la sphere vide, *Bulletin of the Academy of Sciences of the USSR*, **VII** (1934), 793–800.

[4] B. Escofier and J. Pagès, *Analyses factorielles simples et multiples*, Dunod, Paris, 1998.

[5] K. R. Gabriel and R. R. Sokal, A new statistical approach to geographic variation analysis, *Systematic Biology*, **18:3** (1969), 259–278.

[6] L. Gallego-Expósito, Nuevas técnicas de análisis sensorial de alimentos: Métodos espaciales, Degree thesis, 2011.

[7] J.W. Jaromczyk and G.T. Toussaint, Relative neighborhood graphs and their relatives, *Proceedings of the IEEE*, **80:9** (1992), 1502–1517.

[8] T. Kamada and S. Kawai, An algorithm for drawing general undirected graphs, *Information Processing Letters*, **31** (1989), 7–15.

[9] D. Kirkpatrick and J. Radke, *A framework for computational morphology*, in: Computational Geometry, Machine Intelligence and Pattern Recognition, 2, North-Holland, Amsterdam, 1985, 217—248.

[10] J. B. Kruskal, On the shortest spanning subtree of a graph and the traveling salesman problem, *Problems of the American Mathematical Society*, **7** (1956), 48–50.

[11] H. T. Lawless and H. Heymann, *Principles of Sensory Evaluation*, Aspen Publishers, Gaithersburg, 1999.

[12] J. Pagès, Collection and analysis of perceived product inter-distances using multiple factor analysis: Application to the study of 10 white wines from the Loire Valley, *Food Quality and Preference*, **16** (2005), 642–649.

[13] L. Perrin , R. Symoneaux, I. Maître, C. Asselin, F. Jourjon , and J. Pagès, Comparison of three sensory methods for use with the Napping procedure: Case of ten wines from Loire valley, *Food Quality and Preference*, **19** (2008), 1–11.

[14] M. I. Shamos and D. Hoey, Closest-point problems, in *Proceedings of FOCS*, 1975, 151–162.

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

Simulated Annealing applied to the MWPT problem

E.O. Gagliardi*[1], M. G. Leguizamón †[1], and G. Hernández ‡[2]

[1]Universidad Nacional de San Luis, Argentina
[2]Universidad Politécnica de Madrid, España

Abstract

The Minimum Weight Pseudo-Triangulation (MWPT) problem is suspected to be NP-hard. We show here how Simulated Annealing (SA) can be applied for obtaining approximate solutions to the optimal ones. To do that, we applied two SA algorithms, the basic version and our extended hybrid version of SA. Through the experimental evaluation and statistical study we assess the applicability and performance of the SA algorithms. The obtained results show the benefits of using the hybrid version of SA to achieve improved and higher quality solutions for the MWPT problem.

Introduction

A pseudo-triangle is a simple polygon with three convex vertices, and a pseudo-triangulation is a partition of a planar region into pseudo-triangles. See [9] and [10] for surveys about theory and applications, with several interesting results that include combinatorial properties and counting of special classes, rigidity theoretical results, and representations as polytopes, among others.

Optimization problems related to pseudo-triangulations are interesting under several optimality criteria. In this work, we consider the optimality criterion for pseudo-triangulations refereed as Minimum Weight. The weight of a pseudo-triangulation is the total length of their edges. Minimizing the total length is one of the main optimality criteria that provides a quality measure. This formulation is known as the Minimum Weight Pseudo-Triangulation (MWPT) problem. The complexity of the MWPT problem is unknown an it is assumed to be in NP-hard class [6].

Indeed, since no polynomial algorithm is known, approximate solutions of high quality are difficult to obtain by deterministic methods. Thus, we consider approximation algorithms. These algorithms are capable of obtaining approximate solutions to the optimal ones and they can be easily implemented for finding good solutions in NP-hard optimization problems [8].

In this work, we propose the use of *Simulated Annealing* (SA) for finding high quality pseudo-triangulations of minimum weight. For this, we investigate its application through an experimental study and an extended statistical analysis of the results. We have implemented the algorithms involved and additionally, generated our set of instances. Non parametric statistical tests were applied for assessing the performance of the algorithms implemented.

Our recent work on this research, [4] and [5], summarize successive stages of this research using a different metaheuristic. The preliminary results obtained at the initial phase guided to apply a more methodological approach in this research.

This paper is organized as follows. Section 1 describes a general overview of the SA metaheuristic and the main algorithms. Section 2 describes the experimental design, and Section 3 presents the experimental evaluation and statistical analysis. Lastly, Section 4 is devoted to the conclusions.

1 Simulated Annealing

Simulated Annealing applied to optimization problems emerges from the work of S. Kirkpatrick et al. [7] and V. Černý [2]. SA is based on the principles of statistical mechanics whereby the annealing process requires heating and then slowly cooling a substance to obtain a strong crystalline structure. This case is based in an extension of local search (a trajectory-based approach) to solve combinatorial optimization problems. Without loss of generality, the strategy is good for optimization problems, since SA has an explicit strategy to escape from locally optimal solutions. SA introduces a control parameter, T, called *temperature* or *cooling schedule*, whose initial value should be high and should decrease during the search process. The search process is done according to the

*Email: oli@unsl.edu.ar Research supported by Project Tecnologías Avanzadas de Bases de Datos (22/F014), Universidad Nacional de San Luis, Argentina

†Email: legui@unsl.edu.ar - Research supported by LIDIC, Universidad Nacional de San Luis, Argentina

‡Email: gregorio@fi.upm.es - Research supported by ESF EUROCORES programme EuroGICA. ComPoSe IP04-MINCINN Project EUI-EURC-2011-4306

execution of several iterations of the algorithm until a termination condition is achieved.

In order to apply SA to the MWPT problem it is necessary to define some components of a SA by specifying the following parameters: solution space S, objective function f, neighborhood of a solution $\mathcal{N}(S_i)$, initial solution S_0, initial temperature T_0, temperature decrement rule $\mathcal{R}$, number of moves at each temperature $M(T_k)$ (*length of the Markov chain*), acceptance function, and termination condition. The objective of this study is assessing throughout a rigorous experimental study the applicability and respective performances of MWPT-SA and MWPT-SA-2P for the MWPT problem. Next, we describe the common components taken into account in the experimentation.

1.1 The MWPT-SA Algorithm

This algorithm is the result of two combined strategies: random walk and iterative improvement, commonly named diversification and intensification. The search has two phases. The first phase consists of the exploration of the search space; however, this behavior is slowly decreased, leading the search to converge to a local minimum, i.e., phase of iterative improvement. At each iteration, a neighbor of the neighborhood is randomly chosen. The neighborhood of a pseudo-triangulation is obtained by application of edge flips on two adjacent pseudo-triangles [1]. The moves that improve the cost function are always accepted. Otherwise, the neighbor is selected with a given probability that depends on the current temperature. The basic outline is illustrated in Algorithm 1 (MWPT-SA).

Algorithm 1 MWPT-SA

```
Generate an initial solution S_i ∈ S
Set the initial temperature to T_0
k ← 0
while termination condition not met do
  c ← 1
  while c < M(T_k) do
    Choose S_j ∈ N(S_i) ⊂ S
    Evaluate δ = f(S_j) − f(S_i)
    if δ < 0 then
      S_i ← S_j
      SaveBestSoFarSolution
    else
      S_i ← S_j with probability p(T_k, S_i, S_j)
    end if
    c ← c + 1
  end while
  k ← k + 1
  Decrease temperature T_k
end while
```

Algorithm 1 (MWPT-SA) starts with an initial solution $S_i \in S$, which can be randomly or heuristically constructed. Then, it initializes the temperature with value T_0. $M(T_k)$ is the number of iterations for temperature T_k. At each inner iteration, a new solution $S_j \in \mathcal{N}(S_i)$ is randomly generated. If S_j is better than S_i, then S_j is accepted as the current solution. Otherwise, the move from S_i to S_j is an uphill move, and S_j is accepted with a probability computed according to the acceptance function. Finally, the value of T_k is decreased at each outer iteration, controlled by variable k. The algorithm continues in this way until the termination condition is met.

1.2 The MWPT-SA-2P Algorithm

This is an extend scheme, called MWPT-SA-2P algorithm, which was designed considering the cooling schedule of MWPT-SA. The cooling schedule is used for balancing between diversification and intensification, by allowing to return to previous stages. In this manner, the performance of MWPT-SA is improved. Indeed, escaping from the area of low quality in the phase of diversification was sometimes almost impossible for the basic algorithm. Then it is necessary to have the possibility of exploring other regions of the search space. The basic outline is illustrated in Algorithm 2. The algorithm introduces an additional variable (named *previous*) in order to incorporate a control over the trajectory. This variable allows to know which is the parameter setting of the algorithm running where the best so far solution has been found.

Algorithm 2 MWPT-SA-2P

```
Generate an initial solution S_i ∈ S
Set the initial temperature to T_0
k ← 0
while termination condition not met do
  c ← 1
  while c < M(T_k) do
    Choose S_j ∈ N(S_i) ⊂ S
    Evaluate δ = f(S_j) − f(S_i)
    if δ < 0 then
      S_i ← S_j
      SaveBestSoFarSolution
      T_previuos ← T_k
    else
      S_i ← S_j with probability p(T_k, S_i, S_j)
    end if
    c ← c + 1
  end while
  k ← k + 1
  if the best so far solution was updated then
    Decrease temperature T_k
  else
    if it is the first pass on T_k then
      Return to previous temperature T_previuos and
      Do the second pass
    end if
  end if
end while
```

Algorithm 2 (MWPT-SA-2P) controls, before decreasing the temperature, whether during the current temperature T_k the algorithm has found a better solution than the best so far solution. If not, the process returns to a state refereed as the previous tem-

perature, named $T_{previuos}$, where the best so far solution was found. From that state, the algorithm chooses moves for walking in other directions for exploring other areas of the unexplored solution space. In this case, the SaveBestSoFarSolution process saves the best solution found for all cycles so far and the corresponding temperature $T_{previuos}$. The amount of repetitions to get through T_k was experimented, and we conclude that a maximum of two is enough. As the title suggests, 2P in MWPT-SA-2P stands for *double pass*, as the algorithm returns at most once on the path traveled.

2 Experimental Design

Representation. The pseudo-triangulations are planar subdivisions induced by planar embeddings of graphs. For their representation we use a *Doubly-Connected Edge List* (DCEL) [3]. For evaluation purposes in SA, a solution must be transformed from the DCEL into a nxn matrix, where n is the number of points.

Objective Function. The weight of a pseudo-triangulation PT, named $f_w(PT)$, is the sum of the euclidean lengths of all the edges of PT.

Instances collection. An ad-hoc software was designed and implemented by the authors for generating the collection of problem instances, each one being a set P of n points in the plane. A collection of ten (10) instances of size n were generated, with n equal to 40/80/120. Each one is called LDni, $1 \leq i \leq 10$. The points were randomly generated, uniformly distributed, with coordinates $x, y \in [0, 1000]$. For implementation purposes, there are non collinear points.

Parameter Settings for the SA algorithms. The proposed algorithms were executed thirty (30) times using different random seeds for the complete collection of instances. The initial solution is a pseudo-triangulation. We consider two types of initial solutions for MWPT-SA-2P; the initial pseudo-triangulation is: a) a randomly generated solution and b) a pseudo-triangulation obtained by the GPT algorithm [4]. The initial temperature T_0 depends on the number m of edges in the initial solution and the objective function f_w. $T_0 = m \times l$, where l is the average length of the edges of the initial solution. The number of moves at each temperature $N(T_k)$ is $N(T_k) = T_k$ ensuring that the amount of moves is directly proportional to the actual temperature. In each case, for the Temperature decrement rule $\mathcal{R}$, three different types of rules were considered: (i) Fast Simulated Annealing ($T_{k+1} = \frac{T_0}{(1+k)}$); (ii) Very Fast Simulated Annealing ($T_{k+1} = \frac{T_0}{e^k}$); and (iii) Geometric Decrease ($T_{k+1} = \alpha T_k$, where $\alpha \in [0, 1]$). The Geometric cooling scheme had the best performance according to previous experiments, therefore it was chosen for the study presented in this work. For this cooling scheme, we consider $\alpha = 0.8, 0.9$, and 0.95. The setting $\alpha = 0.95$ was chosen due to its high performance. For the termination condition, the search process is finished when the temperature is less than or equal to $T_f = 0.005$.

Resources. The algorithms were implemented in C and, for the statistical analysis, $MATLAB$ was used.

3 Experimental Evaluation and Statistical Analysis for the proposed SA algorithms

This section shows the applicability of MWPT-SA and MWPT-SA-2P algorithms through experimental evaluation. The initial solution can be a randomly solution generated, or a greedy pseudo-triangulation obtained by the GPT algorithm. SA using the last strategy for generating the initial solution can be considered a hybrid approach. We reference the MWPT-SA-2P algorithm as MWPT-SA-2P-RPT (RPT stands for Random Pseudo-Triangulation) or MWPT-SA-2P-GPT (GPT stands for Greedy Pseudo-Triangulation). The experimental and statistical study considers the mentioned set of instances, the best objective, median, average, and standard deviation values. Using Kolmogorov-Smirnov test, we detect that the obtained values do not follow a normal distribution, then non parametric statistical tests were applied to determine if there is significant difference between algorithms. Wilcoxon rank-sum test was applied for allowing systematic pairwise comparisons and assessing whether one of two samples of independent observations came from populations with the same median. MWPT-SA and MWPT-SA-2P-RPT were compared, being the the null hypothesis rejected in all cases. The test rejected the null hypothesis of equal medians with p-value less than 0.01 in all cases. Then, MWPT-SA-2P-RPT and MWPT-SA-2P-GPT were compared, and also the the null hypothesis rejected at all cases, showing the best performance of MWPT-SA-2P-GPT over MWPT-SA-2P-RPT. Figures 1 and 2 display another perspective of the algorithms behavior.

The comparison between MWPT-SA-2P-GPT and GPT algorithms can be observed in Figure 3.

It is also important to highlight that MWPT-SA-2P-GPT achieved objective values that exceed the values obtained by GPT by between 14% and 71%. The SA algorithms have best behavior with respect to GPT. The objective values of the solutions obtained by the greedy algorithm have low quality with respect to those found by the SA algorithms. In summary, among the proposed algorithms, MWPT-SA-2P-GPT achieves the best performance, obtaining the highest quality solutions.

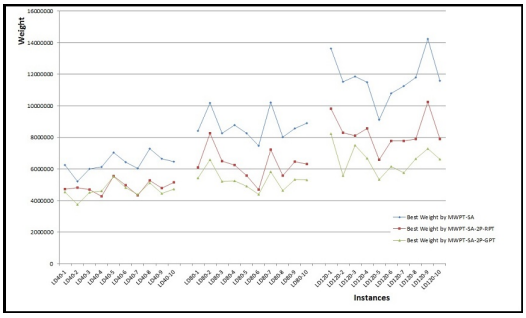


Figure 1: Comparing SA algorithms w.r.t. best values

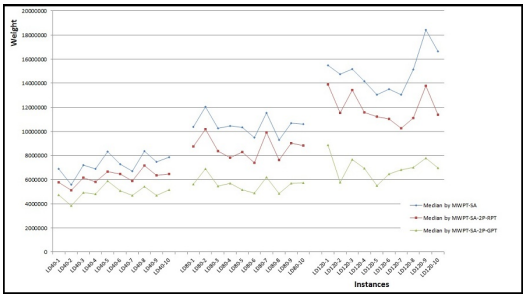


Figure 2: Comparing SA algorithms w.r.t. median values

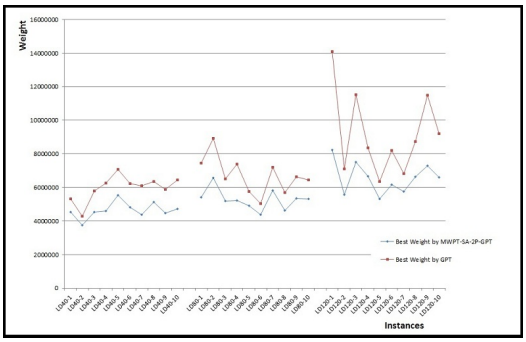


Figure 3: Comparing MWPT-SA-2P-GPT and GPT

In addition, the computational effort of the proposed algorithms applied to the MWPT problem are compared and analyzed. The metaheuristic algorithms consume much more computational resources than a greedy algorithm, but their advantage is that they are capable of achieving solutions of much higher quality. Considering the results obtained and showed in previous subsections for all strategies, Table 1 shows the average runtimes of the mentioned algorithms.

Table 1: Average runtimes of SA algorithms, adding the greedy algorithm for the MWPT problem (in milliseconds).

Instance	MWPT-SA	MWPT-SA-2P-GPT	GPT
40	11679	27210	69
80	25376	33239	83
120	48342	51646	94

4 Conclusions

Our contributions show how SA can be applied to the MWPT problem. We have developed MWPT-SA, MWPT-SA-2P-RPT, and MWPT-SA-2P-GPT algorithms. All claims were corroborated by the experimental study and the respective statical tests. Our conclusions lead us to propose the use of MWPT-SA-2P-GPT for suitably solving MWPT.

References

[1] H. Brönnimann, L. Kettner, M. Pocchiola, and J. Snoeyink. Counting and enumerating pointed pseudotriangulations with the greedy flip algorithm. *SIAM J. Comput.*, 36:721–739, 2006.

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

[3] M. de Berg, M. V. Krevel, M. Overmars, and O. Schwarzkopf. *Computational Geometry. Algorithms and Applications*. Springer-Verlag, 2000.

[4] M. G. Dorzán, E. O. Gagliardi, M. G. Leguizamón, and G. Hernández-Peñalver. Approximations on minimum weight triangulations and minimum weight pseudo-triangulations using ant colony optimization metaheuristic. *Fundamenta Informarticae*, 119(1):1–27, 2012.

[5] E. O. Gagliardi, M. G. Dorzán, M. G. Leguizamón, and G. Hernández-Peñalver. Approximations on minimum weight pseudo-triangulation problem using ant colony optimization. *XXX International Conference of the Chilean Computer Science Society. Jornadas Chilenas de Computación*, 2011.

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

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

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

[9] M. Pocchiola and G. Vegter. Pseudo-triangulations: Theory and applications. In *Symposium on Computational Geometry*, pages 291–300, 1996.

[10] G. Rote, F. Santos, and I. Streinu. *Pseudo-triangulations — a survey*. Contemporary Mathematics. American Mathematical Society, December 2008.

A symbolic-numeric dynamic geometry environment for the computation of equidistant curves

Miguel Á. Abánades*[1] and Francisco Botana†[2]

[1]CES Felipe II, Universidad Complutense de Madrid, 28300 Aranjuez, Spain
[2]Dpto. Matemática Aplicada I, Universidad de Vigo, Campus A Xunqueira, 36005 Pontevedra, Spain

Abstract

A web-based system that determines point/curve and curve/curve bisectors in a dynamic geometry system in a completely automatic way is presented. The system consists of an interactive drawing canvas where the bisector is displayed together with the initial point/curve elements. Algebraic methods are used to provide the equation of an algebraic variety containing the bisector. A numeric approach is followed to provide the graph of the semi-algebraic subset corresponding to the true bisector. It is based on the free dynamic geometry system GeoGebra and the open source computer algebra system Sage.

Introduction

A Dynamic Geometry System (DGS) is a computer application that allows the on-screen drawing of (generally) planar geometric diagrams and the manipulation of these diagrams by mouse dragging. The first standard systems to appear (in the late 80's) were *Cabri* in France [10] and *The Geometer's Sketchpad* in USA [9]. Nowadays special mention deserves *GeoGebra* [6], whose free software model and effective community development has resulted in a spectacular world wide distribution.

From the beginning, DGS have been the paradigm of new technologies applied to Math education. However, most DGS rely on numeric computations and approximate graphs, which make them prone to inaccuracies. Moreover, their lack of symbolic tools prevent DGS from realizing a thorough algebraic treatment of geometry.

In this work we develop a symbolic treatment of bisectors in the plane (locus set of points equidistant to two geometrical elements).

In [1], symbolic algorithms to determine algebraic descriptions of point/curve and curve/curve bisectors are described. In this note we present the implementation of these algorithms in an open web-based system in which symbolic capabilities are added to the DGS GeoGebra by connecting it (remotely) to the Computer Algebra System (CAS) Sage [13].

The presented prototype is based on the remote use of Sage rather than GeoGebraCAS [8], the CAS available within GeoGebra, due to higher versatility of Sage, that includes multiple specialized software packages. The prototype wants to illustrate a general philosophy of DGS-CAS connection that we find more appropriate to implement with the most general systems possible. For this same reason, direct use of Python from GeoGebra has not been considered.

Together with these symbolic web applications, numeric graphic alternative tools are provided to visualize a bisector when the exact symbolic computation of its algebraic description is not computationally possible.

1 Bisectors

Given two geometrical elements (points, curves, surfaces, etc.), their bisector is the locus set of points equidistant to them. We consider bisectors of two geometric objects O_1 and O_2 in the Euclidean 2-space E^2, where each object is a point or an algebraic curve (see for instance Figure 1). Bisectors play an important role when constructing Voronoi diagrams, medial axis transformations and in a variety of algorithms related to shape decomposition (see [12]). A systematic study of plane bisectors can be traced back to [4], where the curves are parametrically described, and [7], where a set containing the bisector is obtained by solving a system of nonlinear equations.

Standard DG systems do not consider bisectors. Besides the usual computation of the parabola via its focus and directrix, there are not other primitives for such computations. Nevertheless, the bisector of a point and a linear object in a DG environment can be partially determined through an elementary locus operation. Since a bisector point is the center of a circle

*Email: abanades@ajz.ucm.es. Research partially supported by the project MTM2011-25816-C02-(01,02) funded by the Spanish *Ministerio de Economía y Competitividad*.

†Email: botana@uvigo.es. Research partially supported by the project MTM2011-25816-C02-(01,02) funded by the Spanish *Ministerio de Economía y Competitividad*.

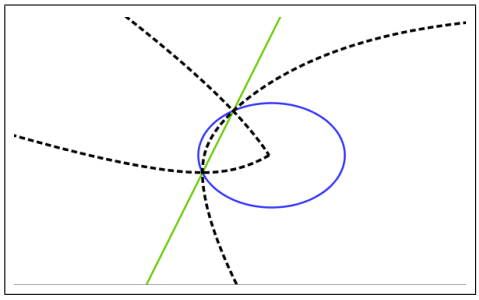


Figure 1: Bisector (dotted) of a line and an ellipse.

tangent to the linear object and passing through the point, the locus tool can suggest a graphical path containing the bisector. Similarly, following the symbolic locus approach in [2], an algebraic variety containing the bisector can be found by solving the corresponding nonlinear polynomial system. We refer to this set as the *algebraic* or *untrimmed* bisector.

In [1], a mixed algebraic-numeric approach for the study of bisectors of points and lines within a DGS is sketched. More precisely, elimination techniques are used for obtaining the detailed description of a bisector. Since the complete bisector description falls out of the algebraic setting, a numerical approach to trim the algebraic bisectors is shown to provide an easy generation of bisectors.

More concretely, the following is the algorithm proposed to compute point/curve bisectors:

Input: curve $c : f(x, y) = 0$, point A
Step 1: Compute (symbolically) the untrimmed (algebraic) bisector of A and c
Step 2: Define this object as an implicit curve d
Step 3: Construct a point B on d
Step 4: Construct the point D on c closest to B
Step 5: If D and A are at the same distance from B, then construct the point E, $E = B$
Return:
i) the locus graphic object *truelocus* of E when B moves along d
ii) the equation(s) of the untrimmed bisector

Moreover, to compute the untrimmed bisector in step 1, a solution based on the remote use of a CAS is indicated.

The system presented here results mainly from the implementation of these algorithms in a web system with a GeoGebra applet through an automatic connection with a remote Sage server.

Determining the algebraic description of some bisectors is computationally out of reach. To obtain the graph of these bisectors we provide an alternative numerical method based on the dynamic color property of GeoGebra whose details can be found in Section 2.2.

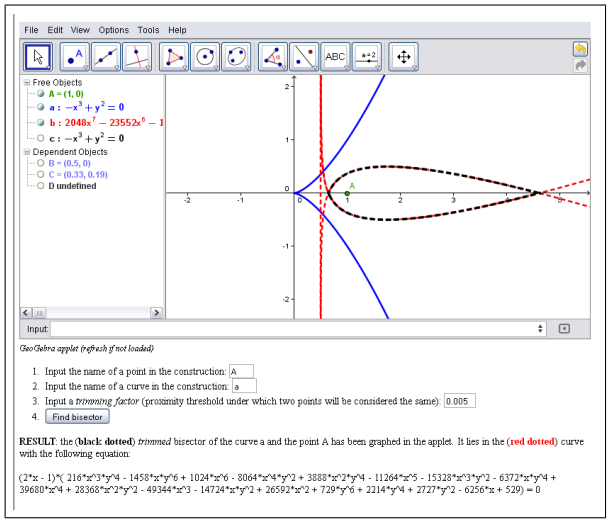


Figure 2: Algebraic bisector (red dotted) and true trimmed bisector (black dotted) of point $A(1, 0)$ and curve $y^2 = x^3$.

2 System Description

The system consists of two main web applications corresponding to the symbolic treatment of point/curve and curve/curve bisectors. Moreover, two auxiliary web applications showing a graphic illustration of a bisector are also provided. They all have been included in a simple web page together with examples and instructions freely available at [15].

All four applications consist of a drawing canvas where the bisectors are displayed together with the initial elements. They all are based on the DGS GeoGebra and the CAS Sage.

GeoGebra is a free DGS with multiple representations of objects in different windows: graphics, algebra, and spreadsheet. Its remarkable world wide use makes GeoGebra a *de facto* standard in the field. Sage is an open source CAS that integrates more than 100 open-source packages (including Singular).

2.1 Symbolic web applications

After the user has input the point and the curve in the applet, he/she just has to press the *Find bisector* button. The aleph.sagemath.org sagecell server [14] is then used to remotely obtain an algebraic variety containing the bisector whose graph is input in the applet. To determine the true (trimmed) bisector, a numeric comparison of distances is carried out. Figure 2 shows how the answer provides both the algebraic description of the bisector together with the true (trimmed) bisector.

The algebraic treatment of the geometric data obtained from the applet is done in Sage with some ad-hoc Python code composed of several hundred lines of

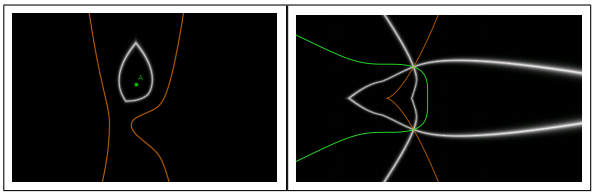


Figure 3: Bisector (white) of point $(0,2)$ and curve $y^4+y^2-y-x^{10}-x^3-x=0$ (left) and bisector (white) of curves $y^2-x^3=0$ and $x^3+y^4=9$ (right).

code. More precisely, once the data are sent to Sage, the appropriate variables are initialized and the ideal corresponding to the task is generated. Singular (a CAS included in Sage with special emphasis on commutative algebra) is then called to basically compute a Groebner basis for this ideal.

It has to be noted that the integration of GeoGebra and Sage has been implemented without loosing interactivity. The DGS-CAS communication is synchronous. That is, changing an element in the GeoGebra construction does automatically trigger the corresponding update on the Sage side.

The structure of the symbolic web application for the computation of simple curve/curve bisectors is similar. It implements in a GeoGebra applet the analytic method sketched in [1] (algorithm 1, Section 3).

2.2 Graphic web applications

In the case of bisectors, even point/curve bisectors in simple situations can be very involved. For instance, the bisector of the point $(2,2)$ and the curve $y=x^7+2$ is a polynomial of degree 20 with more than 150 terms.

A curve/curve bisector is symbolically computed as the intersection of two envelopes (see [3]), each one previously obtained with an elimination procedure using Groebner bases. This makes the process too complex for the symbolic computation of bisectors of curves other than lines and circles with current algorithms and standard hardware.

When obtaining the algebraic description of a point/curve or curve/curve bisector is not possible, two web applications providing a graphic illustration of the bisector have been implemented, one for each type of bisector

The idea of these graphic applications is to scan the different 1-pixel points in the applet to change their RGB color code. The color of a point P is changed according to a formula related to the distances to, for instance, curves a and b, in such a way that the closer the value $distance(P,a)$ is to $distance(P,b)$ the *whiter* the point P becomes. Figure 3 shows a point/curve and a curve/curve bisector as graphed by the applications.

This idea, based on the dynamic color property of GeoGebra, was first used by R. Losada [11]. Nevertheless, this approach should be taken with care. Given the numerical character of the application, misleading answers can be returned.

3 Point/Curve Bisectors

In Figure 2 above we have already seen how the answer given by the application provides a complete description of the bisector of a point and a curve, both algebraically and graphically. Here we give a rough description of the method.

If the curve is non-singular, obtaining the algebraic bisector is a direct application of the elementary method for computing bisector points. Each bisector point must lie on the intersection of the normal line to the curve by a generic point on it, and the perpendicular bisector of this point and the given one. Computing the locus of these points we get the algebraic bisector, which will be trimmed in a subsequent step. Note that if the initial point lies on the curve itself, the bisector is contained in the normal line to the curve, as noted in [5]. However, if the curve is singular, the normal line will remain undefined when the generic point is a singular one, thus including a spurious factor (the perpendicular bisector of the singular point and the initial point) in the elimination result. Nevertheless, after the trimming process, all but a finite number of points in this perpendicular bisector will be excluded.

If the initial point is a singular point on the curve, the normal line will be undefined and the perpendicular bisector will be the whole plane, so the process returns the ideal $\langle 0\rangle$ after elimination. In this case, the trimming procedure does not have a proper variety to trim, since the algebraic bisector is the whole plane. Following [7], we exclude the singular points from our locus finding algorithm. In this way, the application returns some partial information about the bisector. For instance, we have a bi-dimensional bisector for the curve $y^2=x^3$ and the point $(0,0)$ as shown by the point/curve graphic application in [15] (Figure 4, left). In this case, the symbolic web application provides the algebraic description of the boundary of this bi-dimensional bisector (Figure 4, right).

As a conclusion, we note that the proposed applications deal efficiently with regular curves, while for singular ones the results, although sometimes clever, must be taken with caution. The final decision about bisectors in such cases should be guided by an ad-hoc and specific study. The automatic determination of the bisectors in these cases is work in progress.

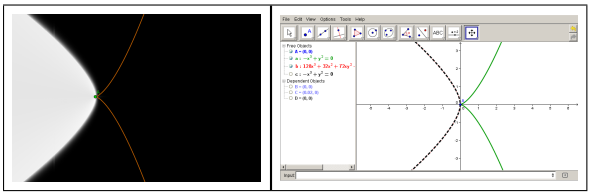


Figure 4: 2D bisector of curve $y^2 = x^3$ and its cusp $(0, 0)$ (left) and its boundary (right).

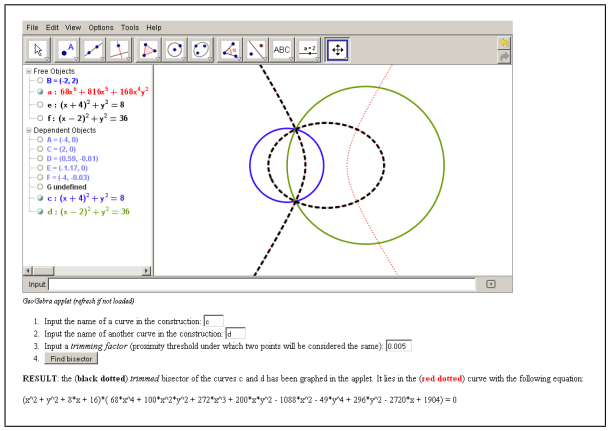


Figure 5: Algebraic bisector (dotted red) and true trimmed bisector (dotted black) of circles $(x + 4)^2 + y^2 = 8$ and $(x - 2)^2 + y^2 = 36$.

4 Curve/Curve Bisectors

As mentioned above, for algebraic curves of low degree the symbolic application for the computation of curve/curve bisectors in [15] provides an algebraic variety where the bisector lies. This algebraic bisector is then trimmed numerically within GeoGebra to display the true bisector.

Figure 5 shows the algebraic bisector and true bisector of the intersecting circles $(x+4)^2+y^2 = 8$ and $(x-2)^2+y^2 = 36$ as provided by the prototype.

For curves of higher degree, the computation of the algebraic bisector is (currently) computationally out of reach. In these cases we have already seen in 2.2 how the graphic web application allows the display of curve/curve bisectors for curves of any degree (see Figure 3).

5 Conclusion

The prototype presented provides tools for the study of bisectors in the plane (point/curve and curve/curve). Complete graphic information is provided together with an exact algebraic description when computationally possible.

The system, web-based and interactive, shows the power of the remote automatic connection of CAS and DGS. Moreover, the exclusive use of free software, shows that it can be done without resorting to expensive commercial systems.

References

[1] F. Botana. Computing bisectors in a dynamic geometry environment. *International Journal of Mathematical Education in Science and Technology*, 44(2):299–310, 2012.

[2] F. Botana and J. L. Valcarce. A software tool for the investigation of plane loci. *Mathematics and Computers in Simulation*, 61(2):139–152, 2003.

[3] R. Farouki and J. Johnstone. Computing point/curve and curve/curve bisectors. In R. B. Fisher, editor, *Design and Application of Curves and Surfaces*, number V in The Mathematics of Surfaces, pages 327–354. Oxford University Press, London, 1994.

[4] R. T. Farouki and J. K. Johnstone. The bisector of a point and a plane parametric curve. *Computer Aided Geometric Design*, 11:117–151, 1994.

[5] R. T. Farouki and R. Ramamurthy. Degenerate point/curve and curve/curve bisectors arising in medial axis computations for planar domains with curved boundaries. *Computer Aided Geometric Design*, 15:615–635, 1998.

[6] GeoGebra. `http://www.geogebra.org`, Last accessed May 2013.

[7] C. M. Hoffmann and P. J. Vermeer. Eliminating extraneous solutions in curve and surface operations. *International Journal of Computational Geometry and Applications*, 1:47–66, 1991.

[8] http://dev.geogebra.org/trac/wiki/GeoGebraCAS, Last accessed May 2013.

[9] N. Jackiw. *The Geometer's Sketchpad v 4.0*. Key Curriculum Press, 2002.

[10] J. M. Laborde and F. Bellemain. *Cabri Geometry II*. Texas Instruments, Dallas, 1998.

[11] R. Losada. Propiedad de color dinámico en GeoGebra, `http://geogebra.es/color_dinamico/color_dinamico.html`, Last accessed May 2013.

[12] M. Peternell. Geometric properties of bisector surfaces. *Graphical Models and Image Processing*, 62:202–236, 2000.

[13] Sage. `http://www.sagemath.org`, Last accessed May 2013.

[14] `https://github.com/sagemath/sagecell`, Last accessed May 2013.

[15] `http://webs.uvigo.es/fbotana/bisectors`, Last accessed May 2013.

Simulating distributed algorithms for lattice agents

Oswin Aichholzer*[1], Thomas Hackl†[1], Vera Sacristán‡[2], Birgit Vogtenhuber*[1], and Reinhard Wallner[1]

[1]Institute for Software Technology, Graz University of Technology, Graz, Austria.
[2]Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, Barcelona, Spain.

Abstract

We present a practical Java tool for simulating synchronized distributed algorithms on sets of 2- and 3-dimensional square/cubic lattice-based agents. This *AgentSystem* assumes that each agent is capable to change position in the lattice and that neighboring agents can attach and detach from each other. In addition, it assumes that each module has some constant size memory and computation capability, and can send/receive constant size messages to/from its neighbors. The system allows the user to define sets of agents and sets of rules and apply one to the other. The *AgentSystem* simulates the synchronized execution of the set of rules by all the modules, and can keep track of all actions made by the modules at each step, supporting consistency warnings and error checking. Our intention is to provide a useful tool for the researchers from geometric distributed algorithms.

Introduction

Mainly due to their scalability, distributed algorithms are a powerful tool for the control of self-organizing systems. One of the most interesting examples of a field of application is the control of modular robotic systems and, in particular, the development of geometric algorithms for their locomotion, reconfiguration, and self-repair. When dealing with these systems and algorithms, it is still often unaffordable to actually implement and run the algorithms on a big set of real prototypes and, in any case, it is recommended to simulate the behavior of the algorithms prior to their actual physical implementation. From a different viewpoint, frequently algorithmic results of theoretical nature are obtained but cannot immediately be translated into physical prototypes, as they may require miniaturization or precision to a level which is still out of reach. Having a simulator at hand is then very convenient. In this paper we present and describe the functionalities of a practical and very general simulator that we hope will be useful in many different research contexts. Other multi-agents simulators are available with different scopes, as for example MASON [5]. Our tool supplies a framework for lattice-based modular robots. The instruction set for the modules/agents is specific, simple, and intentionally compact, as to make the use easy also for non-experts. In the following descriptions we present the 2D information followed, if applicable, by additional information needed for 3D in squared brackets.

1 The agents

The initial agents setting is stored in the file `agents.txt`. Each line of the file defines one agent by its initial (global) coordinates (mandatory) plus (optional) its state, its attachments and initializations for (some of) its counters. Optionally it is possible to state the size of the universe in the agents file.

Universe size `UminX,maxX,minY,maxY[,minZ,maxZ]`
To be positioned at the beginning of the file.

Initial (global) position `x,y[,z]`
The initial position is written as integer x-, y- [and z-]coordinates, separated by a comma.

State `S_____`
The state of an agent consists of exactly 5 characters, written with a leading S.

Attachments `A____[__]`
The attachments of an agent are written as A followed by 4 [6] booleans (0 for not attached, 1 for attached), in the order north, west, east, south[, above, below].

Counters `C__ _____`
Each agent has 25 [45] integer counters, `C00,...,C24 [,C25,...,C44]`, which can be set to any 16-bit integer between -32767 and 32767.

*Email: {oaich,bvogt}@ist.tugraz.at. Research partially supported by ESF EUROCORES programme EuroGIGA - ComPoSe, Austrian Science Fund (FWF): I 648-N18.

†Email: thackl@ist.tugraz.at. Research supported by the Austrian Science Fund (FWF): P23629-N18 "Combinatorial Problems on Geometric Graphs".

‡Email: vera.sacristan@upc.edu. Research partially supported by projects MTM2012-30951, MTM2009-07242, Gen. Cat. DGR 2009SGR1040, and ESF EUROCORES programme EuroGIGA, CRP ComPoSe: MICINN Project EUI-EURC-2011-4306, for Spain.

2 The rules

The definition of what a robot may do is stored in the file `rules.txt`. Each rule definition consists of 4 lines:

1. the name of the rule,
2. the priority of the rule,
3. the precondition, and
4. the postcondition.

The name is a nonempty string. Priorities are used by each agent to decide which of the possibly several rules that apply to its situation to execute. The priority of a rule is a positive integer between 1 and 32767. Higher priorities win over lower ones. The precondition defines whether or not an agent may apply the rule. Finally, the postcondition defines the actions to be performed when a rule is applied to an agent.

2.1 Precondition

The precondition of a rule is any boolean combination of: compare priorities, check neighboring empty/filled positions, check own connections, match states/text or counters/integers, and compare calculation results with counters, messages and integers.

More precisely: a precondition is an *AND* combination of the following.

Neighbors `N____[__]`
The situation of the direct neighboring positions (north, west, east, south[, above, below]). For each of them, `0` denotes empty (no agent), `1` denotes filled (an agent), and `*` denotes indifferent.

Empty position `Edx,dy[,dz]`, `EC__,dy[,dz]`,
An empty position requirement. Written as an E, followed by the relative coordinates of the lattice position required to be empty, separated by a comma. Alternatively instead of each value `dx,dy` or `dz` the name of any counter can be inserted, where a counter starts with a `C`, followed by the two digits number of the counter.

Filled position `Fdx,dy[,dz]`
A filled position requirement. The restrictions and the syntax are the same as in the *Empty position* condition.

Priorities `P____[__]`
Compare the priority of (the applied rule/s) of the direct neighboring agents (north, west, east, south[, above, below]) with the agents' own priority. For each of them, `<` denotes that the priority of such agent needs to be (strictly) smaller, `=` denotes smaller or equal, and `*` denotes indifferent.

Smaller Priority `Ldx,dy[,dz]`
A (strictly) less priority agent requirement. The L is followed by the relative coordinates of the agent required to have smaller priority, separated by a comma. The usage of counters is the same as in the *Empty position* condition.

Smaller or equal Priority `Qdx,dy[,dz]`
A less or equal priority agent requirement. Syntax and usage is analogous to the *Smaller Priority* condition.

Attachments `A____[__]`
The attachment states to the direct neighbors (north, west, east, south[, above, below]), where `0` denotes not attached, `1` denotes attached, and `*` denotes indifferent.

State `S_____`
The agent state can be required to match a simple pattern, where an asterisk matches any character.

State of a remote agent `Tdx,dy[,dz],_____`
This is a combination of the *Filled position* and the *State* precondition. Written as a T, followed by the relative coordinates of the lattice position that needs to be filled, and ended by the state that the remote agent must have. The usage of counters is the same as in the *Empty position* condition.

(Text) messages from direct neighbors
`M*_____`, `MN_____`, `MW_____`, `ME_____`, `MS_____`, `[`, `MA_____`, `MB_____]`
Every agent has four [six] text messages from its direct neighbors (`*`=any, `N`=north, `W`=west, `E`=east, `S`=south[, `A`=above, `B`=below]), each consisting of exactly 5 characters. Any of these messages can be required to match a pattern, where an asterisk matches any character and at most four asterisks are allowed.

Numeric comparisons `<(-)____ (-)____`, `>(-)____ (-)____`, `=(-)____ (-)____`
In addition to its 25 [45] counters, every agent has $4 * 8 = 32$ [$6 * 3 = 18$] numeric messages from its direct neighbors, denoted `#N01,...,#N08`, `#W01,...,#W08`, `#E01,...,#E08`, `#S01,...,#S08` in 2D, and limited to 3 counters per direction in 3D, including `#A01,...,#A03`, and `#B01,...,#B03`. Asterisks can be used instead of a specific direction. Any of these numeric values can be required to fulfill a comparison with respect to any other such value or to any four digit number.

Remote numeric comparisons
`Vdx,dy,C___ (-)____`, `Wdx,dy,C___ (-)____`
These options are only available in 2D. They allow to compare the first value with the second value. `V` indicates strictly smaller and `W` indicates smaller or equal. The first numeric value is a counter from a remote agent at relative coordinates `dx,dy`. It requires the agent to exist. The second numeric value can be a counter, a numeric message from a neighbor or any four digit number. See more details in the *Numeric comparisons* description.

The following two operators enable generating any boolean combination:

Parenthesis `( )`
Group the expressions they surround.

Negation `!`
Negates the expression it precedes.

2.2 Postcondition

The postcondition of a rule defines the actions performed by an agent when it applies the rule. It is any *AND* combination of the following: change position, change attachments, modify state, compute and update counters, and send messages. More precisely:

Position change `Pdx,dy[,dz]`
Move the agent to the given relative coordinates. Counters can be used as in the *Empty position* condition.

Attachments `A____[__]`
For each of the four [six] possible directions (in the already described order), the possibilities are: `0` detach (if attached) before moving, and stay detached afterwards; `1` detach (if attached) before moving, and attach afterwards (if possible); `*` detach (if attached) before moving, and attach afterwards if attached before (and possible); `+` stay attached along the movement. In this case, attached agents are carried along with the moving agent.

State `S_____`
New state of the agent. An asterisk denotes that the according character remains unchanged.

(Text) messages to direct neighbors
`M*_____`, `MN_____`, `MW_____`, `ME_____`, `MS_____`, `[`, `MA_____`, `MB_____]`
Send messages to neighbors (`*` = all).

Calculations on counters and numerical messages `C___ _ ____ ____`, `#___ _ ____ ____`
2D only:
`C___ _ dx,dy,C____ ____`,
`#___ _ dx,dy,C____ ____`,
`C___ _ C__,C__,C____ ____`,
`#___ _ C__,C__,C____ ____`
Every calculation action starts with a position to write the result to (counter or outgoing message), followed by the operation to be performed, and two (readable) values on which the operation is performed. Possible operations are `+` (add), `-` (subtract), `*` (multiply), `/` (divide), `M` (modulo), `A` (maximum), and `I` (minimum). As values for an operation, either four-digit-numbers or internal counters (or one external counter, only in 2D) or incoming numerical messages can be used. The external counter is defined by first indicating the coordinates (`dx,dy` or `C__,C__`) of the agent and then the counter.

Swap `XN`, `XW`, `XE`, `XS`, `[XA`, `XB]`
Exchange the positions of two neighboring agents. Written as a `X`, followed by the desired swap neighbor.

3 The program flow

The program synchronously runs the rules on the agents. It starts by reading the initial setting as well as the set of rules. At every step, the following operations are performed in the order listed below. Alternatively, the order of steps 2 and 3 can be transposed by the user, if desired. It is also possible for the user to make all rules not involving position changes to be applied before those involving position changes.

1. Check and get valid rules. For all agents, check which rules would apply (ignoring priorities) and store valid rules sorted by priority. Store the highest priority of valid rules as current priority and set the agents priority to *open*. For all *open* priority agents sorted by priority, do until all agents have *fixed* priority: i) fetch current rules to current priority, and ii) check priority-conditions for all rules. If they are fulfilled, set priority to *fixed*. If a condition is not fulfilled, remove this rule from the specified agent and if the agents rule list is empty, reduce the current priority to the highest priority of the remaining rules. If a circular dependency between rules on different agents is detected, remove all related rules. Finally, for each agent remove all rules with priority lower than the priority of the agent.

2. Perform actions. For all agents, for all previously stored rules for the agent, perform applicable actions in the following order: i) detach, ii) compute attachment decisions, iii) change position (includes collision detection test), iv) update attachments, v) update state, and vi) swap agents.

3. Compute calculations and send messages. For all agents and for all previously stored rules for the agent, do all calculations (in the order they are listed in the rule) and send numerical messages and all text messages to the post-office. Then, deliver all messages from the post-office to their recipients.

4 The interface

The main window of the program consists of a menu and a tabbed panel with five tabs, as can be seen in the topmost portion of Figure 1.

Universe. This tab allows to visualize the agents as they apply the rules. The algorithm can be visualized

step by step or can be let to run, it can be stopped, and it is also possible to jump one or more steps forwards and backwards. In addition to the colors that can be used to distinguish the agents' states and their attachments, clicking on an agent allows to show its id, position, attachments, state, counter values, messages, and current priority. Zooming and translating the scene is always possible. In the 3-dimensional simulator rotations are also possible. Figure 1 shows a screen shot of the universe of the 2D simulator, in which the information of one of the agents can be seen. The universe panel also shows the current number of iterations, and all warning and error messages.

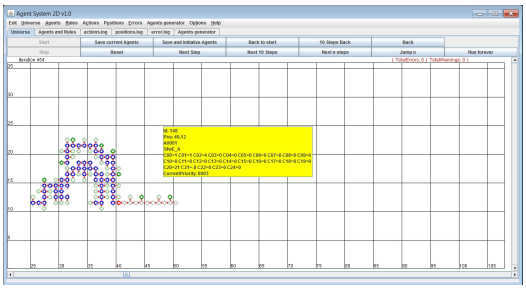


Figure 1: A screen shot of the program, showing the visualization of a set of rules running on a set of agents.

Agents and Rules. The tab consists of two text panels. The left one shows the agents file, the right one shows the rules file. Both files can be independently loaded, modified and saved. Editing shortcuts are provided. When saving any of the files, inconsistencies and syntax errors are detected and marked. See Figure 2 for an illustration.

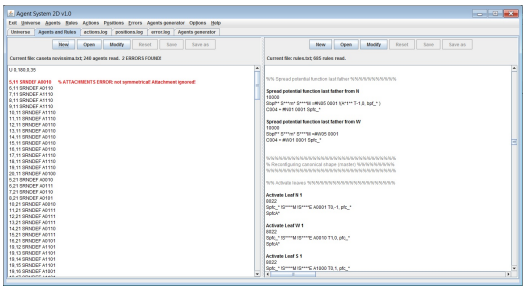


Figure 2: A screen shot of the panel showing the current agents (left) and rules (right). An error detection is shown.

Log tabs. There are three log tabs, each showing the corresponding file. The *actions.log* file stores the information of the rules applied by all agents at each iteration. The *positions.log* file stores the complete information of all agents at each iteration (position, state, attachments, counters, etc.). The *error.log* file stores all error messages at each iteration.

Agents generator. This tab allows to graphically generate or modify a set of agents, together with their attachments, states and counters.

5 Implemented algorithms

We have designed and implemented a large set of distributed algorithms, and we have run them on different configurations of agents. The implemented algorithms cover tasks from self-organization to self-reconfiguration. Self-organization includes: choosing a leader, building a spanning tree, counting the number of agents, and computing the minimum bounding box. All these self-organization tasks refer to connected sets of agents. Details can be found in [4]. Among the self-reconfiguration algorithms, we have implemented generic reconfiguration strategies for arbitrary connected shapes either assuming linear force per module [4], inspired by the centralized algorithm proposed in [1], or only constant force [3], following [2]. In addition, we have also implemented path finding algorithms, as well as some screen-saver-like amusement ones.

6 Conclusion

Our simulator is robust, and it is our strong belief that it will be useful to researchers wishing to run experiments on a wide range of distributed algorithms for self-organizing agents. For practical purposes the system scales linearly in nk, where n is the number of agents and k is the number of rules.

We therefore offer both the simulator and the aforementioned examples to the scientific community. They can be downloaded from the web page [6], which also includes i) the source files, ii) a user guide, and iii) the details of the already implemented algorithms.

References

[1] G. Aloupis, S. Collette, M. Damian, E. D. Demaine, R. Flatland, S. Langerman, J. O'Rourke, S. Ramaswami, V. Sacristán, S. Wuhrer, Linear reconfiguration of cube-style modular robots, *Computational Geometry – Theory and Applications*, **42, 6-7** (2009), 652–663.

[2] F. Hurtado, E. Molina, S. Ramaswami, V. Sacristán, Distributed universal reconfiguration of 2D lattice-based modular robots, in: *Proc. 29th European Workshop on Computational Geometry*, 2013, 139–142.

[3] O. Rodríguez, *Simulació de l'actuació distribuïda de robots modulars*, Degree thesis, Universitat Politècnica de Catalunya, Spain, 2013.

[4] R. Wallner, *A System of Autonomously Self-Reconfigurable Agents*, Degree thesis, Graz University of Technology, Austria, 2009.

[5] http://cs.gmu.edu/∼eclab/projects/mason/
Last visited: May 15, 2013.

[6] http://www-ma2.upc.edu/vera/AgentSystems/

Empty convex polytopes in random point sets

József Balogh*[1], Hernán González-Aguilar†[2], and Gelasio Salazar‡[3]

[1]Department of Mathematics, University of Illinois at Urbana-Champaign. Urbana, IL, United States.
[2]Facultad de Ciencias, Universidad Autónoma de San Luis Potosí. San Luis Potosí, SLP, México.
[3]Instituto de Física, Universidad Autónoma de San Luis Potosí. San Luis Potosí, SLP, México.

Abstract

Given a set P of points in $\mathbb{R}^d$, a *convex hole* (alternatively, *empty convex polytope*) of P is a convex polytope with vertices in P, containing no points of P in its interior. Let R be a bounded convex region in $\mathbb{R}^d$. We show that if P is a set of n random points chosen independently and uniformly over R, then the expected number of vertices of the largest hole of P is $\Theta(\log n/(\log\log n))$, regardless of the shape of R. This generalizes the analogous result proved for the case $d = 2$ by Balogh, González-Aguilar, and Salazar.

Introduction

Given a set P of points in $\mathbb{R}^d$, a *convex hole* (alternatively, *empty convex polytope*) of P is a convex polytope with vertices in P, containing no points of P in its interior.

Recently, we showed that the expected size of the largest convex hole in a random n-point set in the plane is $\Theta(\log n/\log\log n)$ [3]. One anonymous referee of this paper asked if this could be generalized to $d > 2$ dimensions. Joe O'Rourke asked the same question in MathOverflow, and Douglas Zare replied that the $\Omega(\log n/\log\log n)$ lower bound carries over easily to the d-dimensional case [10]. At the end of his reply, Zare wrote: "I don't know whether their harder upper bound of the same form also extends to higher dimensions, but I suspect that it does."

Our aim in this note is to show that, indeed, the upper bound also holds for higher dimensions. Thus, our main result is:

Theorem 1 *Let $d \geq 2$ be an integer, and let R be a bounded convex region in $\mathbb{R}^d$. Let R_n be a set of n points chosen independently and uniformly at random from R, and let $\textsc{Hol}(R_n)$ denote the random variable that measures the number of vertices of the largest convex hole in R_n. Then*

$$\mathbf{E}(\textsc{Hol}(R_n)) = \Theta\left(\frac{\log n}{\log\log n}\right).$$

Moreover, a.a.s.

$$\textsc{Hol}(R_n) = \Theta\left(\frac{\log n}{\log\log n}\right).$$

The proof, which is an immediate consequence of Theorems 2 and 3 below, follows very closely the main ideas of the proof of [3, Theorem 3]. Indeed, the strategy and the main ideas are so close that it seems best to follow as closely as possible the structure of [3]. As we shall see below, some of the results proved in [3] follow without any modification to arbitrary dimensions. The main adaptations needed are:

1. a generalization of the results in [3, Section 2] to $d > 2$ dimensions, to approximate convex sets in $\mathbb{R}^d$ with lattice polytopes; and
2. an adaptation to $d > 2$ dimensions of the results on the probability that a random n-point set is in convex position, from the exact results of Valtr [11, 12] in $\mathbb{R}^2$ to the asymptotic results of Bárány [6] in $\mathbb{R}^d$, for any $d \geq 2$.

The workhorse for the proof of Theorem 1 for arbitrary regions R is the following statement, which takes care of the particular case in which R is a parallelotope.

Theorem 2 *Let R be a parallelotope in $\mathbb{R}^d$. Let R_n be a set of n points chosen independently and uniformly at random from R, and let $\textsc{Hol}(R_n)$ denote the random variable that measures the number of vertices of the largest convex hole in R_n. Then*

$$\mathbf{E}(\textsc{Hol}(R_n)) = \Theta\left(\frac{\log n}{\log\log n}\right).$$

Moreover, a.a.s.

$$\textsc{Hol}(R_n) = \Theta\left(\frac{\log n}{\log\log n}\right).$$

*Email: jobal@math.uiuc.edu. Research supported by NSF CAREER Grant DMS-0745185.

†Email: hernan@fc.uaslp.mx. Research supported by PROMEP.

‡Email: gsalazar@ifisica.uaslp.mx. Research supported by CONACYT Grant 106432.

The other essential fact is that the order of magnitude of the expected number of vertices of the largest convex hole is independent of the shape of R:

Theorem 3 *There exist absolute constants b, b' with the following property. Let R and S be bounded convex regions in $\mathbb{R}^d$. Let R_n (respectively, S_n) be a set of n points chosen independently and uniformly at random from R (respectively, S). Let $\textsc{Hol}(R_n)$ (respectively, $\textsc{Hol}(S_n)$) denote the random variable that measures the number of vertices of the largest convex hole in R_n (respectively, S_n). Then, for all sufficiently large n,*

$$b \leq \frac{\mathbf{E}(\textsc{Hol}(R_n))}{\mathbf{E}(\textsc{Hol}(S_n))} \leq b'.$$

Moreover, there exist absolute constants c, c' such that a.a.s.

$$c \leq \frac{\textsc{Hol}(R_n)}{\textsc{Hol}(S_n)} \leq c'.$$

We remark that Theorem 3 is in line with the following result proved by Bárány and Füredi [5]: the expected number of empty simplices in a set of n points chosen uniformly and independently at random from a convex set A with non-empty interior in $\mathbb{R}^d$ is $\Theta(n^d)$, regardless of the shape of A.

Remark (Proof of Theorem 1). Theorem 1 is an immediate consequence of Theorems 2 and 3.

The proof of Theorem 2 is in Section 1. As we explain in Section 2, the proof of Theorem 3 is totally analogous to the proof of [3, Theorem 2].

We make a few final remarks before we move on to the proofs. For the rest of the paper we let $\mathrm{Vol}(U)$ denote the volume of a region U in $\mathbb{R}^d$. We also note that, throughout the paper, by $\log x$ we mean the natural logarithm of x. Finally, since we only consider sets of points chosen independently and uniformly at random from a region, for brevity we simply say that such point sets are chosen at random from this region.

1 Proof of Theorem 2

We start by noting that if Q, Q' are two regions such that Q' is obtained from Q by an affine transformation, then $\textsc{Hol}(Q_n) = \textsc{Hol}(Q'_n)$. Thus we may assume without loss of generality that R is the isothetic unit area square centered at the origin.

We prove the lower and upper bounds separately. More specifically, we prove that for all sufficiently large n:

$$\Pr\left(\textsc{Hol}(R_n) \geq \frac{1}{2}\frac{\log n}{\log\log n}\right) \geq 1 - n^{-2}. \quad (1)$$

$$\Pr\left(\textsc{Hol}(R_n) \leq d(2+2d^2)\,\frac{\log n}{\log\log n}\right) \geq 1 - n^{-1}. \quad (2)$$

We note that (1) and (2) imply immediately the a.a.s. part of Theorem 2. Now the $\Omega(\log n/\log\log n)$ part of the theorem follows from (1), since $\textsc{Hol}(R_n)$ is a non-negative random variable, whereas the $O(\log n/\log\log n)$ part follows from (2), since $\textsc{Hol}(R_n)$ is bounded by above by n.

Thus we complete the proof by showing (1) and (2).

Proof of (1)

Let R_n be a set of n points chosen at random from R. We prove that a.a.s. R_n has an empty convex polytope of size at least $\frac{\log n}{2\log\log n}$.

Consider the 2-dimensional projection $\pi : \mathbb{R}^d \to \mathbb{R}^2$ defined by $(x_1, x_2, x_3,\ x_4, \ldots, x_d) \to (x_1, x_2, 0, 0, \ldots, 0)$. Note that $\pi(R_n)$ is a set of n points chosen (independently and uniformly) at random from the unit square. Thus it follows from Eq. (1) in [3] that a.a.s. $\pi(R_n)$ has a convex hole H of size at least $\frac{\log n}{2\log\log n}$. Clearly, $\pi^{-1}(H)$ is an empty convex polytope of R_n of size at least $\frac{\log n}{2\log\log n}$. □

Proof of (2)

Let R_n be a set of n points chosen at random from R. We remark that throughout the proof we always implicitly assume that n is sufficiently large.

We shall use the following easy consequence of Chernoff's bound. This is derived immediately, for instance, from Theorem A.1.11 in [1].

Lemma 4 *Let $Y_1, \ldots, Y_m$ be mutually independent random variables with $\Pr(Y_i = 1) = p$ and $\Pr(Y_i = 0) = 1 - p$, for $i = 1, \ldots, m$. Let $Y := Y_1 + \ldots + Y_m$. Then*

$$\Pr\big(Y \geq (3/2)pm\big) < e^{-pm/16}. \qquad \square$$

Let S be the isothetic d-cube of volume 3^d, also (as R) centered at the origin.

We need the following result on approximating convex sets by lattice parallelotopes.

Claim A. *For each positive integer $d > 0$ there exist integers $f_1(d)$ and $f_2(d)$ with the following property. Let H be a convex set in $\mathbb{R}^d$. Then there exists a lattice parallelotope Q_1 such that $H \subseteq Q_1$ and $\mathrm{Vol}(Q_1) \leq (f_1(d)+1)\,\mathrm{Vol}(H)$. Moreover, if $\mathrm{Vol}(H) \geq 2^{d-1} \cdot 1000/n$, then there is a lattice parallelotope Q_0 such that $Q_0 \subseteq H$ and $\mathrm{Vol}(Q_0) \geq (f_2(d) - 1)\mathrm{Vol}(H)$.*

Sketch of Proof. By a the theorem of M. Balla [2], for every convex compact set $H \subset \mathbb{R}^d$ there exists a parallelotope P such that $P \subset H \subset dP = \widehat{P}$ where dP is the image of P under a homothety with ratio d. This implies that $d^{-d}\,\mathrm{Vol}(P) \leq \mathrm{Vol}(H) \leq d^d\,\mathrm{Vol}(P)$.

For each vertex v_i, $i = 0, \ldots, 2^d$, of $\widehat{P}$, let us denote by Q_{v_i} the parallelotope with side length $2/n$ with

facets parallel to the facets of $\widehat{P}$ that has v_i as one of its vertices and $\widehat{P} \cap Q_{v_i} = \{v_i\}$. Observe that each Q_{v_i} contains a d-ball of diameter $2/n$ and for that, there is a lattice point v_i' in the interior of each Q_{v_i}. Let Q_1 be the convex hull of the points $v_1', \ldots, v_{2^d}'$. Note that $d(v_i, v_i') \leq \frac{2}{n}\sqrt{d}$ for each $i = 1, \ldots, 2^d$, this implies that $\varrho(\widehat{P}, Q_1) \leq \frac{2}{n}\sqrt{d}$ where $\varrho(\cdot,\cdot)$ is the Hausdorff metric. Then

$$\begin{aligned}\mathrm{Vol}(Q_1) &\leq \mathrm{Vol}(\widehat{P}) + \frac{2}{n}\sqrt{d}\cdot \mathrm{Surf}(\widehat{P}) + \mathrm{Vol}(B)\\ &\leq d^d\,\mathrm{Vol}(H) + \frac{2}{n}\sqrt{d}\cdot \mathrm{Surf}(\widehat{P}) + \mathrm{Vol}(B)\\ &\leq (f_1(d)+1)\,\mathrm{Vol}(H)\end{aligned}$$

where B is the d-ball of diameter $\frac{2}{n}\sqrt{d}$ and $\mathrm{Surf}(\cdot)$ is the volume $(d-1)$-dimensional.

Now, for each vertex w_i, $i = 0, \ldots, 2^d$, of P consider the parallelotope Q_{w_i} with side length $2/n$ with facets parallel to the facets of P that has w_i as one of its vertices and $Q_{w_i} \subset P$. Because each Q_{w_i} contains a d-ball of diameter $\frac{2}{n}$, then there exists a lattice point in each Q_{w_i}, let w_i' this point. The existence of these points is guaranteed provided that $\mathrm{Vol}(H) \geq 2^{d-1}\cdot 1000/n$ Let Q_0 be the convex hull of the points $w_1', \ldots, w_{2^d}'$. Note that $d(w_i, w_i') \leq \frac{2}{n}\sqrt{d}$ for each $i = 1, \ldots, 2^d$, this implies that $\varrho(P, Q_0) \leq \frac{2}{n}\sqrt{d}$. Then $\mathrm{Vol}(P) - \frac{2}{n}\sqrt{d}\cdot\mathrm{Surf}(Q_0) - \mathrm{Vol}(B) \leq \mathrm{Vol}(Q_0)$ This implies

$$\begin{aligned}\mathrm{Vol}(Q_0) &\geq \mathrm{Vol}(P) - \frac{2}{n}\sqrt{d}\cdot \mathrm{Surf}(Q_0) - \mathrm{Vol}(B)\\ &\geq \mathrm{Vol}(P) - \frac{2}{n}\sqrt{d}\cdot \mathrm{Surf}(P) - \mathrm{Vol}(B)\\ &\geq d^{-d}\,\mathrm{Vol}(H) - \frac{2}{n}\sqrt{d}\cdot \mathrm{Surf}(H) - \mathrm{Vol}(B)\\ &\geq (f_2(d)-1)\mathrm{Vol}(H).\end{aligned}$$ □

For the rest of the proof, for simplicity we define $f_3(d) := f_3(d)$, where $f_1(d)$ and $f_2(d)$ are as in Claim A.

Since there are $(9n+1)^d < (10n)^d$ lattice points, out of which $(n+1)^d$ are in R, it follows that there are fewer than $\binom{(10n)^d}{2^d} < (10n)^{d2^d}$ lattice d-parallelotopes in total, and fewer than $\binom{n^d}{2^d} < n^{d2^d}$ lattice d-parallelotopes all of whose vertices are in R.

Claim B. *With probability at least $1 - n^{-10}$ every lattice parallelotope Q with $\mathrm{Vol}(Q) < 20 f_3(d)\log n/n$ satisfies that $|R_n \cap Q| \leq (3/2)\cdot 20 f_3(d)\log n$.*

Proof. We note that, since $(f_1(d)+1)/f_2(d) > 1$, it follows that $20 f_3(d) > d2^d + 10$. Let Q be a lattice parallelotope with $\mathrm{Vol}(Q) < 20\, f_3(d)\log n/n$, and let $Z = Z(Q) \subseteq R$ be any lattice parallelotope containing Q, with $\mathrm{Vol}(Z) = 20\, f_3(d)\log n/n$. Let X_Q (respectively, X_Z) denote the random variable that measures the number of points of R_n in Q (respectively, Z). We apply Lemma 4 with $p = \mathrm{Vol}(Z)$ and $m = n$, to obtain $\Pr(X_Z \geq (3/2)\cdot 20\, f_3(d)\log n) < e^{-(3/2)20\, f_3(d)/24\log n} = n^{-(5/4) f_3(d)}$. Since $Q \subseteq Z$, it follows that $\Pr(X_Q \geq (3/2)\cdot 20\, f_3(d)\log n) < n^{-(5/4) f_3(d)}$. As the number of choices for Q is at most $(10n)^{d2^d}$, with probability at least $(1-(10n)^{d2^d}\cdot n^{-(5/4) f_3(d)}) > 1 - n^{-10}$, no such Q contains more than $(3/2)\cdot 20\, f_3(d)\log n$ points of R_n. □

A polytope is *empty* if its interior contains no points of R_n.

Claim C. *With probability at least $1 - n^{-10}$, there is no empty lattice parallelotope $Q \subseteq R$ with $\mathrm{Vol}(Q) \geq 20\,(d2^d + 10)\log n/n$.*

Proof. The probability that a fixed lattice parallelotope $Q \subseteq R$ with $\mathrm{Vol}(Q) \geq d2^d + 10\log n/n$ is empty is $(1 - \mathrm{Vol}(Q))^n < n^{-d2^d+10}$. Since there are fewer than n^{d2^d} lattice parallelotopes in R, it follows that the probability that at least one of the lattice parallelotope with area at least $(d2^d + 10)\log n/n$ (and hence with area at least $20\,(d2^d + 10)\log n/n$) is empty is less than $n^{d2^d}\cdot n^{-(d2^d+10)} \leq n^{-10}$. □

For the rest of the proof, we let H be a maximum size convex hole of R_n.

Claim D. *With probability at least $1 - n^{-10}$ we have $\mathrm{Vol}(Q_1) < f_3(d)\log n/n$.*

Proof. Suppose first that $\mathrm{Vol}(H) < 2^{d-1}1000/n$. Then $\mathrm{Vol}(Q_1) \leq 2^{d-1}\cdot 1000(f_1(d)+1)/n$. Since this is obviously smaller than $f_3(d)\log n/n$, in this case we are done. Now suppose that $\mathrm{Vol}(H) \geq 2^{d-1}\cdot 1000/n$, so that Q_0 (from Claim A) exists. Moreover, $\mathrm{Vol}(Q_1) \leq (f_1(d)+1)\mathrm{Vol}(H)$. Since $Q_0 \subseteq H$, and H is a hole of R_n, it follows that Q_0 is empty. Thus, by Claim C, with probability at least $1 - n^{-10}$ we have that $\mathrm{Vol}(Q_0) < (d2^d + 10)\log n/n$. Now since $\mathrm{Vol}(Q_1) < (f_1(d)+1)\,\mathrm{Vol}(H)$ and $\mathrm{Vol}(Q_0) \geq f_2(d)\cdot\mathrm{Vol}(H)$, it follows that $\mathrm{Vol}(Q_1) \leq (f_1(d)+1)\,\mathrm{Vol}(Q_0)/f_2(d)$. Thus with probability at least $1 - n^{-10}$ we have that $\mathrm{Vol}(Q_1) \leq \big((f_1(d)+1)(d2^d+10)/f_2(d)\big)\log n/n = f_3(d)\log n/n$. □

Claim E. *For each fixed integer $d > 0$, there exist a universal positive constant $c_2 := c_2(d)$ with the following property. Let K be any convex polytope in $\mathbb{R}^d$. Then the probability that r points chosen at random from K are in convex position is at most $(c_2\, n^{\frac{2}{d-1}})^{-n}$.*

Proof. This is an immediate consequence of [6] (see for instance [4, Theorem 2.1]). □

Claim F. *With probability at least* $1 - 2n^{-2}$ *the random point set* R_n *satisfies that no lattice parallelotope* Q *with* $\mathrm{Vol}(Q) < 20\, f_3(d) \log n/n$ *contains* $d(2+2d^2)\log n/(\log\log n)$ *points of* R_n *in convex position.*

Proof. Let Q be a lattice parallelotope with $\mathrm{Vol}(Q) < 20\, f_3(d)\log n/n$. By Claim B, with probability at least $1 - n^{-10}$ we have $|R_n \cap Q| \le (3/2)\cdot 20\, f_3(d)\log n$. Thus it suffices to show that the probability that there exists a lattice parallelotope Q with with $|R_n \cap Q| \le (3/2)\cdot 20\, f_3(d)\log n$ and $d(2+2d^2)\log n/(\log\log n)$ points of R_n in convex position is at most n^{-2}.

Let $c_2 := c_2(d)$ be as in Claim E. Thus the expected number of r-tuples of R_n in Q in convex position is at most

$$\begin{aligned}
&\binom{|R_n \cap Q|}{r} \left(c_2 r^{\frac{2}{d-1}}\right)^{-r} \\
&\qquad \le \binom{(3/2)\cdot f_3(d)\log n}{r} \left(c_2 r^{\frac{2}{d-1}}\right)^{-r} \\
&\qquad \le \left(\frac{e\cdot(3/2)\cdot F_2 \log n}{r}\right)^{r} \left(c_2 r^{\frac{2}{d-1}}\right)^{-r} \\
&\qquad < \left(c_3 \log n \cdot r^{-1-\frac{2}{d-1}}\right)^{r},
\end{aligned}$$

where $c_3 := 3e f_3(d)/2c_2$.

Since there are at most n^{d2^d} choices for Q, it follows that the expected total number of such r-tuples with $r = d(2+2d^2)\log n/\log\log n$ is at most $n^{d2^d}\cdot\left(c_3\log n\cdot r^{-1-\frac{2}{d-1}}\right)^r < n^{-2}$ (this last inequality follows from an elemenary but long manipulation). This completes the proof, since it follows that the probability that such an r-tuple exists is at most n^2. □

To finish the proof of (2), recall that H is a maximum size empty convex polytope of R_n, and that $H \subseteq Q_1$. It follows immediately from Claims D and F that with probability at least $1 - n^{-1}$ the parallelotope Q_1 does not contain a set of $d(2+2d^2)\log n/(\log\log n)$ points of R_n in convex position. In particular, with probability at least $1 - n^{-1}$ the size of H is at most $d(2+2d^2)\log n/(\log\log n)$. □

2 Proof of Theorem 3

The proof of Theorem 3 is totally analogous to the proof of [3, Theorem 2]. Indeed, in that proof, essentially all the arguments are independent of the dimension. The only adaptation that needs to be done is that we need a version of [3, Corollary 6] for $d > 2$ dimensions. We recall that [3, Corollary 6] claims that if H is a closed convex set in $\mathbb{R}^2$, then there exist rectangles U, K such that $U \subseteq H \subseteq K$, $\mathrm{Vol}(U) \ge \mathrm{Vol}(H)/8$, and $\mathrm{Vol}(K) \le 2\mathrm{Vol}(H)$.

A d-dimensional analogue of this statement follows from the following result in [9]: if H is a convex body in $\mathbb{R}^d$, then H contains a parallelotope P such that some translate of dP contains K. Indeed, this implies at once that if H is a closed convex set in $\mathbb{R}^d$, then there exist parallelotopes U, K such that $U \subseteq H \subseteq K$, $\mathrm{Vol}(U) \ge \mathrm{Vol}(H)/d$, and $\mathrm{Vol}(K) \le d\cdot\mathrm{Vol}(H)$.

References

[1] N. Alon and J. Spencer. The probabilistic method, 3rd. Edition. Wiley, 2008.

[2] M. Y. Balla, Approximation of convex bodies by parallelotopes, *Internal Report IC/87/310, International Centre for Theoretical Physics, ICTP*, Trieste, 1988

[3] J. Balogh, H. González-Aguilar, and G. Salazar, Large convex holes in random point sets. *Comp. Geom.* **46** (2013), 725–733.

[4] I. Bárány, Random points, convex bodies, lattices. *Proceedings of the International Congress of Mathematicians, Vol. III* (Beijing, 2002), 527–535. Higher Ed. Press, Beijing, 2002.

[5] I. Bárány and Z. Füredi, Empty simplices in Euclidean space, *Canad. Math. Bull.* **30** (1987) 436–445.

[6] I. Bárány, A note on Sylvester's four-point problem. *Studia Sci. Math. Hungar.* **38** (2001), 73–77.

[7] I. Bárány, Sylvester's question: the probability that n points are in convex position, *Ann. Probab.* **27** (1999), 2020–2034.

[8] I. Bárány and P. Valtr, Planar point sets with a small number of empty convex polygons. *Studia Sci. Math. Hungar.* **41** (2004), 243–266.

[9] G.D. Chakerian and S. K. Stein, Some intersection properties of convex bodies. *Proc. Amer. Math. Soc.* **18** (1967), 109–112.

[10] J. O'Rourke, Empty convex polytopes for random point sets. `http://mathoverflow.net/questions/107875` (2012).

[11] P. Valtr, Probability that n Random Points are in Convex Position. *Discrete and Computational Geometry* **13** (1995), 637–643.

[12] P. Valtr, The Probability that n Random Points in a Triangle Are in Convex Position. *Combinatorica* **16** (1996), 567–573.

Note on the number of obtuse angles in point sets

Ruy Fabila-Monroy*[1], Clemens Huemer†[2], and Eulàlia Tramuns‡[2]

[1]Departamento de Matemáticas, Cinvestav-IPN
[2]Departament de Matemàtica Aplicada IV, Universitat Politècnica de Catalunya, BarcelonaTech

Abstract

In 1979 Conway, Croft, Erdős and Guy proved that every set S of n points in general position in the plane determines at least $\frac{n^3}{18} - O(n^2)$ obtuse angles and also showed the upper bound $\frac{2n^3}{27} - O(n^2)$ on the minimum number of obtuse angles among all sets S. We prove that every set S of n points in convex position determines at least $\frac{2n^3}{27} - o(n^3)$ obtuse angles, hence matching the upper bound (up to sub-cubic terms) in this case. Also on the other side, for point sets with low rectilinear crossing number, the lower bound on the minimum number of obtuse angles is improved.

Introduction

A point set S in the plane is in general position if no three points of the set lie on a common straight line. Throughout, all considered point sets S will be in general position in the plane and $|S| = n$. An angle abc at b determined by three points $\{a, b, c\}$ of S is obtuse if it is greater than $\frac{\pi}{2}$. Prominent problems and results on obtuse and acute angles in point sets can be found in [4]. Here we are interested in the number of obtuse angles determined by point sets S. Conway et al. [3] proved that the minimum number of obtuse angles among all sets S is between $\frac{n^3}{18} - O(n^2)$ and $\frac{2n^3}{27} - O(n^2)$. In this note we prove that point sets S in convex position determine at least $\frac{2n^3}{27} - o(n^3)$ obtuse angles. Interestingly, this matches (up to sub-cubic terms) the upper bound example from [3]. We conjecture that $\frac{2n^3}{27}$ is indeed the right order of magnitude for the minimum number of obtuse angles. Point sets in convex position are characterized as the point sets that maximize the *rectilinear crossing number*. The rectilinear crossing number $cr(S)$ of a point set S equals the number of convex quadrilaterals with vertices in S. Hence, an upper bound of $\binom{n}{4}$ on the rectilinear crossing number is obvious. As for point sets with low crossing number, the current best lower bound is $\frac{277}{729}\binom{n}{4} + \Theta(n^3)$ [2] and there are point sets S that only have $cr(S) = 0.380488\binom{n}{4} + \Theta(n^3)$ [1]. We show that point sets S whose crossing number is not too large, at most $\frac{2}{3}\binom{n}{4}$, have more than $\frac{n^3}{18}$ obtuse angles.

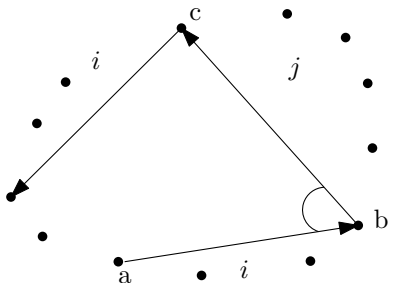


Figure 1: Alternately skipping i and j points in the polygonal path for class (i, j).

Proofs

Theorem 1 *Every set S of n points in convex and general position in the plane determines at least $\frac{2n^3}{27} - o(n^3)$ obtuse angles.*

Proof. First we consider the case when n is a prime number; the case when n is not a prime number will be treated at the end of the proof. We label the points of S from 0 to $n-1$ in counter-clockwise order. For three points $a, b, c \in S$ in counter-clockwise order, we say that the angle abc at point b is of class (i, j) if the open halfplane bounded by the line through points a and b, and not containing point c contains i points of S, and if the open halfplane bounded by the line through points b and c, and not containing point a contains j points of S; see Figure 1. (Then ab is an i-edge and bc is a j-edge.) Hence each angle defined by S belongs to some class (i, j), where $0 \leq i + j \leq n - 3$. For i, j fixed, $i \neq j$, we consider the polygon P that starts at point 0, visits points of S in counter-clockwise order, alternately skipping i points and j points of S, until it returns to point 0 the second time. Three steps of

*Email: ruyfabila@math.cinvestav.edu.mx. Partially supported by Conacyt of Mexico, Grant 153984.

†Email: clemens@ma4.upc.edu. Partially supported by projects MEC MTM2012-30951 and Gen. Cat. DGR 2009SGR1040 and ESF EUROCORES programme EuroGIGA, CRP ComPoSe: grant EUI-EURC-2011-4306, for Spain.

‡Email: etramuns@ma4.upc.edu. Partially supported by MTM2011-28800-C02-01 from Spanish MEC.

such a polygonal path of P are shown in Figure 1. Note that the polygon P is self-intersecting and can visit vertices more than once.

- *Claim:* Each angle of class (i,j) and each angle of class (j,i) is encountered exactly once in P.

  We modify P to obtain a new polygon P' by pairing two consecutive steps of P which skip i points and j points respectively; that is, we now move from a point m to point $m+i+j+2 \mod n$. Since n is a prime number, each non-zero element of the additive group $\mathbb{Z}_n$ is a generator of the group; in particular also $i+j+2$. This implies that P' returns to the starting point 0 after it visited each point of $S\backslash\{0\}$ exactly once. We now retrieve the original polygon P by splitting the paired steps into steps skipping alternately i points and j points. It follows that each point of S is visited twice in P, and each angle of class (i,j) and each angle of class (j,i) is encountered exactly once in P.

- *Claim:* The rotation number [5] of the polygon P is $i+j+2$.

  The rotation number measures how many times the polygon turns around. Note that the underlying point set is in convex position and all steps are done in counter-clockwise order. The polygon visits each vertex m twice; from a point m the polygonal path continues once to point $m+i+1 \mod n$, and once to point $m+j+1 \mod n$; in total the path advances $i+j+2$ points from m. Hence, summing over all n vertices, we count $(i+j+2)n$ steps between consecutive points of the point set in counter-clockwise order. n steps between consecutive points describe one full turn. Thus the rotation number is $i+j+2$.

- *Claim:* At least $2n-3(i+j+2)$ angles of the $2n$ angles of classes (i,j) and (j,i) encountered in P are obtuse.

  For the sake of contradiction, suppose that P contains less than $2n-3(i+j+2)$ obtuse angles. Then, P contains more than $3(i+j+2)$ acute or right angles. By an averaging argument, at least one of the $i+j+2$ full turns of the polygon contains more than three acute or right angles. But this is not possible, unless P contains four right angles forming a 4-cycle contradicting n being a prime number.

  Hence, each pair of classes (i,j) and (j,i) of angles, with $i \neq j$, contains at least $2n-3(i+j+2)$ obtuse angles. Summing over all possible values i,j we thus get the lower bound on the number of obtuse angles in S

$$\frac{1}{2}\sum_{i=0}^{\lfloor \frac{2n}{3}-2 \rfloor} \sum_{j=0, j\neq i}^{\lfloor \frac{2n}{3}-2-i \rfloor} 2n-3(i+j+2) = \frac{2n^3}{27} - O(n^2).$$

  It remains to consider the case when n is not a prime number. In this case it suffices to only count the number of obtuse angles in a subset S' of S consisting of n_p points, where n_p is the largest prime number smaller than n. Since $n_p > n - o(n)$, see e.g. [6], we get the lower bound on the number of obtuse angles in S

$$\frac{2(n-o(n))^3}{27} - O(n^2) = \frac{2n^3}{27} - o(n^3).$$

□

Lemma 2 *Every set S of n points in general position in the plane with rectilinear crossing number $cr(S)$ determines at least $\frac{n^3}{12} - \frac{cr(S)}{n-3} - O(n^2)$ obtuse angles.*

Proof. We first remark that the number of right angles formed by S is negligible for our purpose. In fact, it is enough to observe that no edge spanned by S is incident to more than two right angles, due to the general position assumption. Hence we upper bound the number of right angles by $2\binom{n}{2}$. Each 4-tuple of points in convex position forms at least one obtuse angle or four right angles; and each 4-tuple of points not in convex position forms at least two obtuse angles. Thus, the total number of obtuse angles in S is at least $\frac{\left(cr(S)-2\binom{n}{2}/4\right)\cdot 1+\left(\binom{n}{4}-cr(S)\right)\cdot 2}{n-3}$, where we divide by $n-3$ because each obtuse angle is counted $n-3$ times. Simplifying gives the claimed bound. □

References

[1] B.M. Ábrego, M. Cetina, S. Fernández-Merchant, J. Leaños, G. Salazar, 3-symmetric and 3-decomposable geometric drawings of K_n, *Discrete Applied Mathematics* **158** (2010), 1240–1258.

[2] B.M. Ábrego, M. Cetina, S. Fernández-Merchant, J. Leaños, G. Salazar, On $(\leq k)$-edges, crossings, and halving lines of geometric drawings of K_n, *Discrete and Computational Geometry* **48** (2012), 192–215.

[3] J.H. Conway, H.T. Croft, P. Erdős, M.J.T. Guy, On the distribution of values of angles determined by coplanar points, *Journal of the London Mathematical Society* **(2) 19** (1979), 137–143.

[4] P. Erdős, Z. Füredi, The greatest angle among n points in the d-dimensional Euclidean space, *Annals of Discrete Mathematics* **17** (1983), 275–283.

[5] B. Grünbaum, G.C. Shephard, Rotation and winding numbers for planar polygons and curves, *Transactions of the American Mathematical Society* **322** (1990), 169–187.

[6] R. Guy. *Unsolved problems in number theory*, Third Edition, Springer, New York, 2004.

Stabbing simplices of point sets with k-flats

Javier Cano*[1], Ferran Hurtado†[2], and Jorge Urrutia‡[1]

[1]Instituto de Matemáticas, Universidad Nacional Autónoma de México, Mexico City, Mexico.
[2]Departament de Matemàtica Aplicada, Universitat Politécnica de Catalunya, Barcelona, Spain.

Abstract

Let S be a set of n points in $\mathbb{R}^d$ in general position. A set $\mathcal{H}$ of k-flats is called an m_k-stabber of S if the relative interior of any m-simplex with vertices in S is intersected by at least one element of $\mathcal{H}$. In this paper we give lower and upper bounds on the size of minimum m_k-stabbers of point sets in $\mathbb{R}^d$. We study mainly m_k-stabbers in the plane and in $\mathbb{R}^3$.

Introduction

A set $\{x_0, \ldots, x_k\} \in \mathbb{R}^d$ of $k+1$ points is called *linearly independent* if there is no linear combination $\lambda_0 x_0 + \cdots + \lambda_k x_k = \mathbf{0}$ in which at least one $\lambda_i \neq 0$, $k \leq d$. A set of points $\{x_0, \ldots, x_k\} \in \mathbb{R}^d$ is called *affinely independent* if the set $\{x_1 - x_0, \ldots, x_k - x_0\}$ is linearly independent, $k \geq 1$. A set consisting of a single point will also be considered as affinely independent. An affine combination of a set of $k+1$ affinely independent points in $\mathbb{R}^d$ is a linear combination $\lambda_0 x_0 + \cdots + \lambda_k x_k$ of $x_0, \ldots, x_k$ such that $\lambda_0 + \cdots + \lambda_k = 1$. The set of affine combinations of $k+1$ affinely independent points of $\mathbb{R}^d$ is called a k-flat. In particular, a $(d-1)$-flat of $\mathbb{R}^d$ will be called a *hyperplane*. Following our intuition, a 1-flat is a line, and a 2-flat is a plane. A k-flat is isomorphic to the k-dimensional space $\mathbb{R}^k$. A point set S in $\mathbb{R}^d$ is in general position if any subset of S with at most $d+1$ elements is affinely independent.

An *m-simplex* of $\mathbb{R}^d$ is the convex hull of a set of $m+1$ affinely independent points in $\mathbb{R}^d$, $m \leq d$. For example, in $\mathbb{R}^2$ a 0-simplex is a point, a 1-simplex is a segment and a 2-simplex is a triangle.

Given a k-flat h and an m-simplex $\mathcal{P}$ of $\mathbb{R}^d$, we say that h stabs $\mathcal{P}$ if h intersects the relative interior of $\mathcal{P}$. A set of k-flats $\mathcal{H}$ is called an *m_k-stabber* of a set of points S if every m-simplex induced by the elements of S is stabbed by at least one element of $\mathcal{H}$.

For example, in the plane a set of points Q (respectively lines) is a 3_0-stabber (respectively a 3_1-stabber) of point set S if any triangle with vertices in S contains an element of Q in its interior (respectively is intersected by a line in Q).

In this paper we study the following problem: Given two integers $k < m$, $k < d$, and $m \leq d+1$, and a set S of n points in $\mathbb{R}^d$ in general position, how many k-flats are needed to stab all the m-simplices generated by the elements of S?

In this paper we focus mainly on stabbing all the r-simplices of point sets on the plane or in $\mathbb{R}^3$, and give some results for higher dimensions.

The problem of finding sets of points that stab all the triangles of a point set has been studied by Katchalsky and Meir [4], and independently by Czyzowicz, Kranakis, and Urrutia [2]. They prove that for any point set S in the plane in general position with n elements, such that its convex hull has c elements, the set of triangles of S can be stabbed with exactly $2n - c - 2$ points, and that such a bound is tight, as any triangulation of S has $2n-c-2$ triangles. Stabbers for other convex holes, such as quadrilaterals and pentagons, have also been studied [1].

Given a point set S in $\mathbb{R}^d$, we define $f_k^m(S)$ as the size of the smallest m_k-stabber of S. We define $f_k^m(n)$ as the largest value that a m_k-stabber can have over all the point sets S of $\mathbb{R}^d$ with n elements. With this terminology, the preceding result in the plane translates to $f_0^2(S) = 2n - c - 2$.

In this paper we show upper and lower bounds for $f_k^m(n)$, for point sets on the plane and $\mathbb{R}^3$ that are tight up to a constant.

1 Well-separated sets and generalized ham-sandwich cuts

A family of convex sets $C_0, \ldots, C_k$ of $\mathbb{R}^d$, $k \leq d$, is *well separated* if for any choice of $x_i \in C_i$, the set of points $\{x_0, \ldots, x_k\}$ is affinely independent. A family $P_0, \ldots, P_k$ of finite point sets in $\mathbb{R}^d$ is *well separated* if their convex hulls $Conv(P_0), \ldots, Conv(P_k)$ are well separated, $k \leq d$.

*Email: jcano@ciencias.unam.mx. Research supported by project number 178379 Conacyt, México.

†Email: ferran.hurtado@upc.edu. Research partially supported by projects MINECO277 MTM2012-30951, Gen. Cat. DGR2009SGR1040, and ESF EUROCORES programme EuroGIGA, CRP ComPoSe IP04: MICINN Project EUI-EURC-2011-4306.

‡Email: urrutia@matem.unam.mx. Research supported by project number 178379 Conacyt, México..

Notice that in particular, this implies that for any pair of different indexes $i, j \leq k$, the convex hulls of P_i and P_j do not intersect.

Let $P_0, \ldots, P_k$ be a family of pairwise disjoint finite point sets in $\mathbb{R}^d$, $k \leq d$. Given positive integers $a_i \leq |P_i|$, an $(a_0, \ldots, a_k)$-*cut* is a hyperplane h for which $h \cap P_i \neq \emptyset$ and $|h^+ \cap P_i| = a_i$, $0 \leq i \leq d$, where h^+ is the half-space bounded below by h.

The following result by Steiger and Zao will be useful to us:

Theorem 1 ([8]) *Let $P_0, \ldots, P_d$ be well-separated point sets in $\mathbb{R}^d$ such that $P_0 \cup \cdots \cup P_d$ is in general position, and let $a_0, \ldots, a_d$ be positive integers such that $a_i \leq |P_i|$. Then there is a unique $(a_0, \ldots, a_d)$-cut of $P_0, \ldots, P_d$.*

Finally, we recall the almost folklore result about ham-sandwich cuts:

Theorem 2 (Ham-sandwich theorem, [9]) *Every d finite sets in $\mathbb{R}^d$ can be simultaneously bisected by a hyperplane. A hyperplane h bisects a finite set P if each of the open half-spaces defined by h contains at most $\lfloor \frac{|P|}{2} \rfloor$ points of P.*

Now we have all the tools that we will use to give an algorithm for constructing a worst-case optimal m_k-stabber for point sets in the plane and in the 3-dimensional space.

2 Stabbers in the plane

We start by studying the following problem: How many lines are necessary to stab the set of line triangles (segments) determined by triples (pairs) of elements of a point set S in the plane? In our previous terminology, determine lower and upper bounds for $f_1^1(n)$ and $f_1^2(n)$ for point sets on the plane. We first prove:

Theorem 3 *For any set S of n points in the plane $r \leq f_1^2(S) \leq \lceil \frac{n}{4} \rceil$, where r is the smallest number such that $\frac{n}{2} \leq 2 + 2 + 3 + \cdots + r$. Our bounds are tight.*

Proof. The lower bound follows from the fact that r lines, no three of which intersect at a point, divide the plane into $2 + 2 + 3 + \cdots + r$ convex regions, and if a set of r lines stabs all of the triangles with vertices in S, then S has at most two elements in each of these regions.

We now prove that $f_1^2(S) \leq \lceil \frac{n}{4} \rceil$. To this end, we now show how to obtain a set $\mathcal{H}$ with $\lceil \frac{n}{4} \rceil$ straight-lines that is a 2_1-stabber of S. To prove our result, it is sufficient to find a set $\mathcal{H}$ with $\lceil \frac{n}{4} \rceil$ lines such that any cell of the arrangement generated by $\mathcal{H}$ contains at most two elements of S.

First, we put in $\mathcal{H}$ one straight line ℓ that separates S into two subsets of size $\lceil \frac{n}{2} \rceil$ and $\lfloor \frac{n}{2} \rfloor$ respectively. Refer to these sets as S_1 and S_2. Clearly, these sets are well separated, and by Theorem 1 we can find a $(2, 2)$-cut for S_1 and S_2. This cut is a line ℓ_1 that leaves two elements x_1, x_2 of S_1 and two elements x_1', x_2' of S_2 above it. These points are separated from the remaining points of S by ℓ and ℓ_1, if ℓ_1 contains some element of S_1 or S_2 move it slightly. Thus any triangle containing at least one vertex in $\{x_1, x_2, x_1', x_2'\}$ is intersected by ℓ or ℓ_1.

We repeat this recursively on $S_1 \setminus \{x_1, x_2\}$ and $S_2 \setminus \{x_1', x_2'\}$, $\lceil \frac{n}{4} \rceil - 2$ times, obtaining a set of $\lceil \frac{n}{4} \rceil$ lines (including ℓ and ℓ_1) that is a 2_1-stabber of S.

To show that our bound is tight, let S be a set of n points in the plane in convex position labeled $p_0, p_1, \ldots, p_{n-1}$ in counterclockwise order, with n even. Consider the set of $\frac{n}{2}$ triangles $p_i p_{i+1} p_{i+2}$, i even, and $i \leq \frac{n}{2}$ for every even value of i. Observe that any two of these triangles intersect at exactly one vertex, for $n \geq 6$.

We claim that any straight line stabs at most 2 of these triangles. This is true since every such triangle has exactly two edges of the convex hull of P_n; then any straight line stabbing one such triangle stabs at least one edge of the convex hull. Any straight line stabs at most two edges of the convex hull, so then it can stab at most two of such triangles. Thus to stab all of those triangles we need at least one straight line for every two triangles. Observe next that if the elements of S are in convex position, then to stab the edges of the convex hull of S we need at last $\lceil \frac{n}{4} \rceil$ lines. Thus our upper bound is tight. □

The key property of $\mathcal{H}$ in the proof of this result is the fact that any cell of the arrangement induced by $\mathcal{H}$ contains at most two elements of S. Observe that if instead of finding a $(2, 2)$-cut for S_1 and S_2, we use $(m-1, m-1)$-cuts, then we obtain a set of lines such that any cell of the arrangement generated by them contains at most $m - 1$ elements of S. An m island of S is a subset S' of S with exactly m elements and such that the convex hull of S' contains no element of $S \setminus S'$ in its convex hull. The next result follows easily from the proof of Theorem 3.

Theorem 4 *For any set S of n points in the plane, $\lceil \frac{n}{2(m-1)} \rceil$ lines are always sufficient and sometimes necessary to stab all of the m islands of S.*

In particular, for $m = 2$, this proves that $f_1^1(S) \leq \lceil \frac{n}{2} \rceil$; that is, to stab the segments of S we need at most $\lceil \frac{n}{2} \rceil$ lines. It is easy to see that the bound $\lceil \frac{n}{2(m-1)} \rceil$ in Theorem 4 is achieved for point sets in convex position.

We close this section by observing that the problem of calculating $f_1^1(S)$ is equivalent to calculating the minimum number of lines such that in each face of

the arrangement defined by these lines, there is at most one element of S. This problem is known as the shattering problem and its decision version is known to be *NP*-complete [5].

3 Stabbers in the space

We consider now the problem of stabbing the tetrahedra induced by an n point set S in $\mathbb{R}^3$ using sets of planes. Our approach is similar to that of the previous section.

For the lower bound we will use the following set of n points in the plane. Let $f\colon \mathbb{R} \to \mathbb{R}^3$, $f(a) = (a, a^2, a^3)$. $f(\mathbb{R})$ is known as the *momentum curve* in $\mathbb{R}^3$. Clearly, any plane in $\mathbb{R}^3$ intersects $f(\mathbb{R})$ in at most 3 points. Let $C_n = \{p_0, \ldots, p_{n-1}\}$ with $p_i = f(i)$. Since any plane intersects $f(\mathbb{R})$ in at most 3 points, C_n is in general position. Given any segment p_ip_j, let $f([i,j])$ be the curve containing the points $f(x)$, with $i \leq x \leq j$. We call $f([i,j])$ the *shadow* of the segment p_ip_j in the momentum curve. The following observation will be useful.

Observation 1 *Let $p_i, p_j \in C_n$. Then any plane intersecting the relative interior of the segment p_ip_j intersects its shadow.*

Consider the set $\{t_0, \ldots, t_{\lfloor \frac{n-1}{3} \rfloor}\}$ of tetrahedra, such that the vertex set of t_i is $\{p_{3i}, p_{3i+1}, p_{3i+2}, p_{3i+3}\}$. Observe that the interiors of these tetrahedra are pairwise disjoint, and that t_i and t_{i+1} share exactly one vertex, namely p_{3i+3}.

Observation 2 *If a plane intersects the interior of a tetrahedron, then it intersects the relative interior of one of its edges.*

We define the *shadow* of t_i as the shadow of the segment $p_{3i}p_{3i+3}$ which is in fact the union of the shadows of the edges of t_i. Then the relative interiors of the shadows of $\{t_0, \ldots, t_{\lfloor \frac{n-1}{3} \rfloor}\}$ are pairwise disjoint. The next lemma follows now directly from Observations 1 and 2:

Lemma 5 *If a plane stabs t_i, then it intersects the relative interior of its shadow.*

Since any plane intersects the momentum curve at most three times, it follows that any plane intersects at most three elements of $\{t_0, \ldots, t_{\lfloor \frac{n-1}{3} \rfloor}\}$.

By Observation 1, any plane stabbing t_i must intersect its shadow; thus any any plane stabs at most three elements of $\{t_0, \ldots, t_{\lfloor \frac{n-1}{3} \rfloor}\}$.

Thus we have:

Lemma 6 $f_2^3(n) \geq \lceil \frac{n-1}{9} \rceil$.

We will now prove that $f_2^3(n) \leq \lceil \frac{n}{9} \rceil + 2$. To this end, we are going to give an algorithm that splits any point set S into a a family of well-separated families of subsets of S.

First find two parallel planes π_1 and π_2 such that they split S into three subsets S_1, S_2 and S_3 such that $||S_i| - |S_j|| \leq 1$, $i \neq j \leq 3$. That is, each S_i contains $\lfloor \frac{n}{3} \rfloor$ or $\lceil \frac{n}{3} \rceil$ elements.

We now apply the ham-sandwich theorem in $\mathbb{R}^3$ to find a plane π_3 that simultaneously bisects S_1, S_2 and S_3. Let S_i^+ be the subset of S_i that lies above π_3 and S_i^- the one that lies below π_3, with $i = 1, 2, 3$. This gives us two disjoint well-separated families $\mathcal{F} = \{S_1^+, S_2^-, S_3^+\}$ and $\mathcal{F}' = \{S_1^-, S_2^+, S_3^-\}$. Similarly to the algorithm in the previous section, iteratively find $(3,3,3)$-cuts for $\mathcal{F}$, removing the nine points above the cut until each set in $\mathcal{F}$ contains at most three points, and adding the cut planes to the 3_2-stabber. Then repeat the same process for $\mathcal{F}'$.

It is now easy to see that the set of planes we obtain has at most $\lceil \frac{n}{9} \rceil + 2$ elements, and that they form a stabber of all the tetrahedra with vertices in S. Thus we have proved:

Theorem 7 $\lceil \frac{n-1}{9} \rceil \leq f_2^3(n) \leq \lceil \frac{n}{9} \rceil + 2$.

Observe that using similar arguments we can prove that $f_2^2(n) \approx n/6$ (stabbing triangles with planes) and $f_2^1(n) \approx n/3$ (stabbing segments with planes).

3.1 Stabbing 2-simplices of point sets with lines

For the problem of stabbing triangles with lines, it is easy to see that in some instances, we need at least $\frac{2n-4}{2}$ lines. Indeed, consider any set of points S in $\mathbb{R}^3$ with n points in convex position. Observe that the convex hull of S contains exactly $2n-4$ triangles, and that any line intersects the interior of at most two of these triangles. Thus we have proved that in $\mathbb{R}^3$, $f_1^2(n) \geq n - 2$.

We now prove that $2n-5$ lines are always sufficient to stab all of the triangles of S. To prove this, first project the elements of S into the real plane, thus obtaining a point set S' in $\mathbb{R}^2$. It is known that if the convex hull of S' has c elements, it is always possible to find a set of points Q with at most $2n - c - 2$ points that stabs all of the triangles of S'; see [2, 4]. Each point $q \in Q$ generates a vertical line ℓ_q passing through q. Since each triangle T of S projects to a triangle T' of S', there is a line ℓ_q, $q \in Q$ that stabs T' and thus T. Since in the worst case, the convex hull of S' has three points, we have proved:

Theorem 8 *In $\mathbb{R}^3$, $f_1^2(n) \leq 2n - 5$.*

3.2 Stabbing the 3-simplices of a point set in $\mathbb{R}^3$ with points

We now turn our attention to the problem of stabbing the set of tetrahedra generated by a point set S in $\mathbb{R}^3$ using sets of points. We will assume that no two elements of S lie on a line vertical to the real plane.

Consider again the momentum curve $f(a) = (a, a^2, a^3)$, and the set of points $C_n = \{p_0, \ldots, p_n\}$ as defined above. Let $0 \leq i, j \leq n$ such that $i + 1 < j$. Let $T_{i,j}$ be the tetrahedra with vertex set $\{p_i, p_{i+1}, p_j, p_{j+1}\}$. It is well known [3] that the set of tetrahedra $T_{i,j}$, with i and j as above, have disjoint interiors, and form a tetrahedralization of the convex hull of $C_n = \{p_0, \ldots, p_n\}$. Since any set of points that stabs the tetrahedra of S contains at least one point in each of these $T_{i,j}$, it follows that $f_0^3(n+1)$ is at least $\binom{n-1}{2}$.

We now prove an upper bound on $f_0^3(n+1)$. We proceed as follows: For every two elements $x_i, x_j \in S$, place a point $p_{i,j}$ slightly *above* the midpoint of the line segment joining x_i to x_j. For every $x_i \in S$, place a point p_i slightly *below* x_i.

We now prove that the set of points containing all the $p'_{i,j}s$ and the p_i's stabs all the tetrahedra of S.

Consider any four points x_i, x_j, x_k, x_ℓ of S. Let H be the convex hull of $\{x_i, x_j, x_k, x_\ell\}$. Two cases arise. In the first case, H projects to a convex quadrilateral in the plane. In this case, one edge of the convex hull of $\{x_i, x_j, x_k, x_\ell\}$ is not visible from below. Suppose w.l.o.g. that it is the edge joining x_k to x_ℓ. Then the point $p_{i,j}$ stabs H.

Suppose then that H projects to a triangle T on the plane. Then one vertex of H, say x_i, belongs to the interior of T. Two sub-cases arise. In the first sub-case, when we see H from below, x_i is not visible. In this case, the point p_i stabs the convex hull of $\{x_i, x_j, x_k, x_\ell\}$. In the remaining sub-case, $p_{i,j}$ stabs H.

Thus we have that we can always choose $\binom{n+1}{2} + n$ points that stab the tetrahedra of S. It is not hard to see that we can reduce the above number by 9 by observing that if the line segment joining x_i to x_j is not visible from below, and this line segment belongs to the convex hull of S; then the point $p_{i,j}$ is redundant. Also if a point x_i belongs to the convex hull of S and is visible from below, then p_i is redundant.

Thus we have proved:

Theorem 9 $\binom{n-1}{2} \leq f_0^3(n+1) \leq \binom{n+1}{2} + n - 9$.

4 Some thoughts on higher dimensions

Some of the results in this paper; e.g. Lemma 5 and Lemma 6, generalise easily to higher dimensions, yielding the following result:

Lemma 10 *In* $\mathbb{R}^d$, $\lceil \frac{n-1}{d^2} \rceil \leq f_{d-1}^d(n)$.

And in general:

Lemma 11 *In* $\mathbb{R}^d$, $\lceil \frac{n-1}{dm} \rceil \leq f_{d-1}^m(n)$.

To finish this paper, we conjecture:

Conjecture 1 $f_m^d(n) \approx \frac{n}{m \cdot d}$.

References

[1] J. Cano, A. García, F. Hurtado, T. Sakai, J. Tejel, J. Urrutia, Blocking the k-holes of point sets in the plane, *submitted.*

[2] 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.

[3] B. Grünbaum, *Convex Polytopes*, Interscience Monographs in Pure and Applied Mathematics, Vol. XVI, John Wiley & Sons, London-New York-Sydney.

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

[5] R. Freimer, J. S. B. Mitchell, C. D. Piatko, On the Complexity of Shattering Using Arrangements, in: *Proceedings of the 2nd Canadian Conference on Computational Geometry*, 1990, 218–222.

[6] A. Pór, D. R. Wood, On visibility and blockers, *J. Computational Geometry* **1(1)** (2010), 29–40.

[7] T. Sakai, J. Urrutia, Covering the convex quadrilaterals of point sets, *Graphs and Combinatorics* **38** (2007), 343–358.

[8] W. Steiger, J. Zhao, Generalized Ham-Sandwich Cuts, *Discrete and Computational Geometry* **44(3)** (2010), 535–545.

[9] H. Steinhaus, A note on the ham sandwich theorem, *Mathesis Polska* **9** (1938), 26–28.

Stackable tessellations

Lluís Enrique*[1] and Rafel Jaume†[2]

[1]ETH Zürich, Switzerland
[2]Freie Universität Berlin, Germany

Abstract

We introduce a class of solids that can be constructed gluing stackable pieces, which has been proven to have advantages in architectural construction. We derive a necessary condition for a solid to belong to this class. This helps to specify a simple sufficient condition for the existence of a stackable tessellation of a given solid. Finally, we show the compatibility of our method with some discretization techniques appearing in the literature.

Introduction

During the last decades, the increasing demand of architectural freeform shapes has motivated the study of surface tesselations in tiles which can be manufactured with a low economic impact. Substantial research has been done on planar tesselations with triangular meshes, quadrilateral meshes [1, 2] and its special cases of spherical and conical meshes [3]. Nevertheless, tessellating methods in non-standard non-planar panels have been rarely considered, since for most construction materials the production of such building components is very expensive. The innovative CASTonCAST project [4] proposes a system which consists of two parts: a geometric method for the construction of double curved shapes by means of stackable tiles and an efficient fabrication technique of curved precast panels which relies on shaping building components in stacks by using the previous component as a mold for the next one. The latter relies on using the top surface of each component as part of the mold for the next one, after applying a chemical barrier. Once the components have been fabricated in stacks, they are detached and assembled by glueing them through their lateral faces, forming a new solid called strip. This procedure features obvious advantages concerning storage and transport as well.

We analyse the kind of solids that can be constructed using stacks under some reasonable assumptions. More precisely, we focus on constructions where the tiles form a face-to-face tessellation whose adjacency graph is a grid both in stack and strip forms, as in Figure 1. We also require the junction faces in the strip configuration to be planar. After stating necessary conditions for a solid to admit a stackable tessellation as desired, we provide sufficient conditions and a procedure to obtain the stacks that generate it. It turns out that the solids for which a stackable tessellation exists are naturally defined, skipping some technical details, as the ones generated sweeping a surface around an axis or in a fixed direction. In addition, we prove that, under certain conditions, a one-parametric family of stackable tessellations exists. Finally, we approach the problem of approximating a target solid using polyhedral tiles.

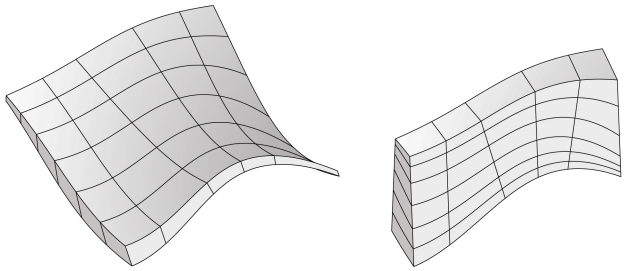


Figure 1: Stacks (right) and associated strips (left)

We normally use capital letters for points, lowercase letters for vectors, lines and rays, and Greek letters for curves. We add a * to the names in order to indicate the correspondences between the 2- and 3-dimensional elements of the constructions in Sections 1 and 2. We will often omit the indices ranges, for ease of reading. The curves and surfaces are considered to have no self-intersections. Abusing notation, we denote the image set of a curve or a surface by its name. Curves are assumed to be parametrized using the unit interval. We define the wedge spanned by two rays to be the cone defined by their supporting lines and containing an unbounded part of both rays.

*Email: enrique@arch.ethz.ch. Research supported by Obra Social "la Caixa".

†Email: jaume@mi.fu-berlin.de. Research supported by Obra Social "la Caixa" and the DAAD.

1 Planar stacks

Before starting the analysis of the 3-dimensional case, we study a 2-dimensional object we call *planar stack*. Its construction is described in the next subsection and illustrated by Figure 2.

1.1 Construction

Let p and q be two infinite rays emanating from a point K and spanning a wedge W of angle $\alpha \in (0, \pi)$ in $\mathbb{R}^2$. Let $\sigma_0, ..., \sigma_n$ be pairwise disjoint, simple curves contained in W. Assume the indices correspond to the inverse order a ray shot from K would intersect them. Assume further that $\sigma_i \cap p = \sigma_i(0)$ and $\sigma_i \cap q = \sigma_i(1)$ and require that

$$d(\sigma_i(0), \sigma_{i+1}(0)) = d(\sigma_{i-1}(1), \sigma_i(1)), \qquad (1)$$

where d denotes the Euclidean distance in the plane. Let $\mathcal{T}_i$ denote the closed subset of W bounded by σ_i and σ_{i+1}. Each of these simply connected sets will be called a *tile* and each σ_i will be called a *separator*. The whole construction will be referred to as a *planar stack of angle* α. It may also be required that the curves σ_i are *separable*, i.e., that a tile can be moved away from the next one by a rigid motion preventing them to overlap. A simple condition ensuring this property is that each σ_i is monotone in some direction but we will ignore this requirement in this article.

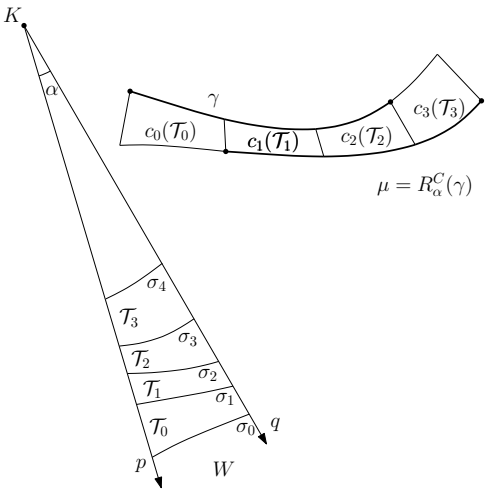


Figure 2: A planar stack and the corresponding strip.

Note that condition (1) ensures that the tiles can be rearranged gluing $\sigma_{i+1}(0)$ with $\sigma_i(1)$ and $\sigma_i(0)$ with $\sigma_{i-1}(1)$. This configuration will be called the *strip* associated to the planar stack.

It will be handy to define $s_i : \mathbb{R}^2 \to \mathbb{R}^2$ to be the congruence that maps $\mathcal{T}_i$ to its position in the strip, assuming $s_0 = Id$. Given a point $C \in \mathbb{R}^2$ and an angle α, we define R^C_α to be the counterclockwise rotation of angle α and center C.

Theorem 1 *If the strip $\bigcup_{i=0}^{n-1} s_i(\mathcal{T}_i)$ associated to a planar stack of angle α is a simply connected set, then its boundary contains the curves $\gamma = \bigcup_{i=0}^{n-1} s_i(\sigma_{i+1})$ and $R^C_\alpha(\gamma)$, for some point $C \in \mathbb{R}^2$.*

Proof. The assumptions that all the σ_i start in p and end in q, are pairwise disjoint and have no self-intersections ensure that the tiles are topologically disks. Provided that $\bigcup_{i=0}^{n-1} s_i(\mathcal{T}_i)$ is simply connected, the tiles must intersect only in the glued segments. In addition, condition (1) ensures that $s_i(\sigma_i(1)) = s_{i+1}(\sigma_{i+1}(0))$ and $s_i(\sigma_{i+1}(1)) = s_{i+1}(\sigma_{i+2}(0))$, for $i \in \{0, ..., n-2\}$. Consequently, it is clear that γ and $\mu = \bigcup_{i=1}^{n} s_i(\sigma_i)$ are contained in the boundary of the strip. Observe now that when $\mathcal{T}_{i+1}$ is glued next to $\mathcal{T}_i$, the two copies of σ_{i+1} involved are congruent via $R^{C_i}_\alpha$, for some point C_i. But $R^{C_i}_\alpha$ maps $s_i(\sigma_{i+1}(1))$ to $s_{i+1}(\sigma_{i+1}(1))$ and $R^{C_{i+1}}_\alpha$ maps $s_{i+1}(\sigma_{i+2}(0))$ to $s_{i+2}(\sigma_{i+2}(0))$, which are the same pair of points. Since for any pair of points $P \neq Q$ and a given α there is only one point O such that $R^O_\alpha(P) = Q$, it has to be $C_i = C$ for all $i \in \{0, ..., n-1\}$. Thus, we have that $\mu = R^C_\alpha(\gamma)$. □

1.2 Tessellation

Constructing a planar stack and developing it can be used as a modelling tool. However, we are now interested in the other direction of the procedure. That is, given a curve γ, an angle α and a center of rotation C, construct a planar stack such that, when reconfigured into strip, contains γ and $R^C_\alpha(\gamma)$ in its boundary. Such a planar stack may not exist. If it does, it may not be unique. Indeed, there is in general a one-parametric family of possibly suitable planar stacks.

Theorem 1 indicates that we need to construct a stack of angle α if we aim to obtain a shape with γ and $R^C_\alpha(\gamma)$ forming part of its boundary. Consider the locus of points that see a counterclockwise angle α between $\gamma(0)$ and $R^C_\alpha(\gamma(0))$, i.e., the set

$$\kappa = \{P \in \mathbb{R}^2 : R^P_\alpha(\overrightarrow{P\gamma(0)}) = \overrightarrow{PR^P_\alpha(\gamma(0))}\}, \qquad (2)$$

where $\overrightarrow{XY}$ is the ray starting at X and going through Y. It is well known that κ is a circumference arc going from $\gamma(0)$ to $R^C_\alpha(\gamma(0))$. It is not hard to see that $C \in \kappa$. Obviously, any candidate K to be the vertex of the stack must belong to κ and there is a possibly valid planar stack leading to our target for each of such points.

Consider a fixed $K \in \kappa$ and let r_0 be the ray containing K, passing through $\gamma(0)$ and such that $R^C_\alpha(r_0)$ touches but does not cross r_0. We define $r_i = R^C_{i\alpha}(r_0)$ for $i \in \mathbb{N}$. Let W_i be the wedge of angle α spanned by r_i and r_{i+1} and define also $\mathcal{U}_i$ to be the region in W_i bounded by γ and $R^C_\alpha(\gamma)$. For simplicity, we assume that γ does not intersect $R^C_{-\alpha}(r_0)$. This last restriction is not necessary, but simplifies the technicalities.

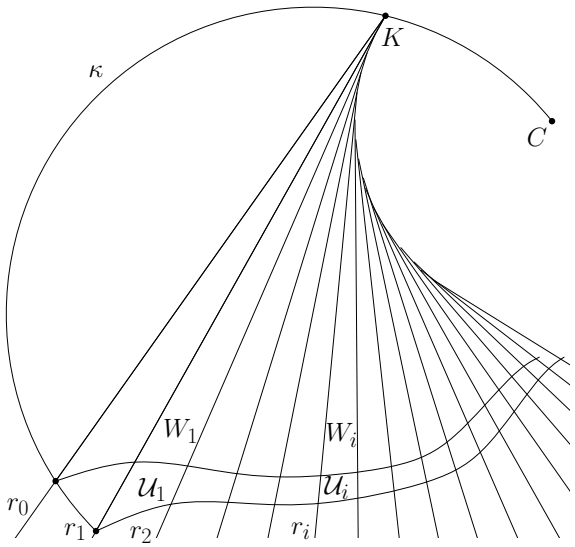


Figure 3: Tessellation procedure.

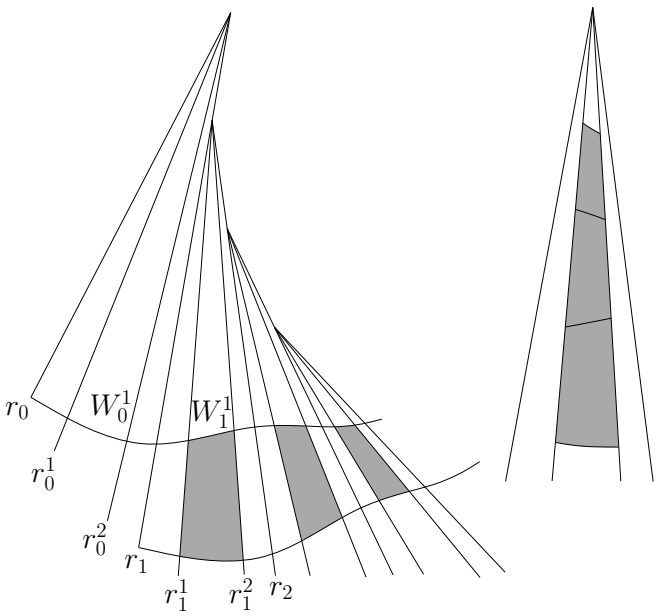


Figure 4: Shaded tiles can be arranged in stack form.

Theorem 2 *If $r_0, ..., r_n$ intersect γ in a single point, the tiles $\mathcal{U}_1, ..., \mathcal{U}_{n-1}$ can be arranged as a planar stack.*

Proof. We will prove that $\{R^C_{-i\alpha}(\mathcal{U}_i)\}_{i=1}^{n-1}$ is a set of cells tessellating a simply connected subset of W_0. Observe first that $W_{i+1} = R^C_\alpha(W_i)$, since $r_{j+1} = R^C_\alpha(r_j)$ for all $j \in \mathbb{N}$. For the same reason, $r_1, ..., r_{n+1}$ intersect $R^C_\alpha(\gamma)$ in a single point. Since γ and $R^C_\alpha(\gamma)$ do not intersect, the boundary of $\mathcal{U}_i$ is formed by a subcurve of γ and a subcurve of $R^C_\alpha(\gamma)$ which are mutually disjoint, one segment contained in r_i and one contained in r_{i+1}. It remains to be proven that each tile matches the next one and they all fit in W_0. This can be derived from the facts that $R^C_\alpha(\mathcal{U}_i \cap \gamma) = \mathcal{U}_{i+1} \cap R^C_\alpha(\gamma)$ and that $R^C_{-i\alpha}(W_i) = W_0$. □

Using the previous theorem, it is not hard to see that under an additional technical condition, one can construct a stack containing the whole γ and $R^C_\alpha(\gamma)$ in its boundary. We state this in the following corollary, omitting its simple proof.

Corollary 3 *If $r_0, ..., r_l$ intersect γ in a single point and r_{l+1} does not intersect γ, then there exists a planar stack containing γ and $R^C_\alpha(\gamma)$ in its boundary.*

1.3 Refinement

For obvious reasons, it can be useful to refine a stackable tessellation. This feature will be essential in the discussion concerning approximation in Section 3.

Adopting the notation used in the previous section, consider the angularly ordered rays $r_0 = r_0^0, ..., r_0^m = r_1$ emanating from K and contained in W_0. Define also r_i^j to be $R^C_{i\alpha}(r_0^j)$ for $i \in \{1, ..., n\}$ and $j \in \{0, ..., m\}$. Let W_i^j be the wedge spanned by r_i^j and r_i^{j+1}. Define also $\mathcal{U}_i^j$ to be the space in W_i^j bounded by γ and $R^C_\alpha(\gamma)$.

Proposition 4 *If $r_0^0, ..., r_0^m = r_1^0, r_1^1, ..., r_l^m$ intersect γ in a single point, the tiles $\mathcal{U}_1^j, ..., \mathcal{U}_{n-1}^j$ can be arranged as a planar stack, for $j \in \{0, ..., m-1\}$.*

Proof. The way we defined r_i^j ensures that $R^C_{-i\alpha}(W_i^j) = W_0^j$. Therefore, the arguments in the proof of Theorem 2 apply to each j individually. □

Note that, in addition, the m stacks obtained from the refined tessellation can be obtained by cutting the original stack by straight radial cuts.

The results presented in this section can easily be extended to the case where K is thought of as a point at infinity. That is, we consider the wedge W to be the space between two parallel lines. Then, the two copies of γ appearing in the boundary of the strip are related by a simple translation. Conversely, given a curve and a translated copy of it, a one-parametric family of possible stacks can be again evaluated. In this case, we may consider κ to be the set of directions of the plane, except for the direction of W. We can then cut the curves by a set of equally spaced lines in any direction $u \in \gamma$. This case is specially interesting because it generates offset shapes. Moreover, requiring that γ is monotone in u is sufficient to ensure the existence of the corresponding stack with the additional separability property.

2 Solid stacks

We consider now a 3-dimensional extension of the previous results, which is the actual target of this work. In this way, we model a meaningful subset of the solids constructible using the CASTonCAST fabrication technique.

The initial object is now the space W^* between two non-parallel planes p^* and q^*, which we further bound by two planes orthogonal to $K^* = p^* \cap q^*$. Cutting W^* with pairwise-disjoint surfaces $\sigma_0^*, ..., \sigma_n^*$, each of them dividing W^* in two parts, a sequence of 3-dimensional tiles $\mathcal{T}_i^*$ is defined. If we require that $R_\alpha^{K^*}(\mathcal{T}_i^* \cap p^*) = \mathcal{T}_{i+1}^* \cap q^*$ (the condition analogous to (1)), the resulting object will be called a *solid stack*. The tiles can be rearranged by glueing congruent faces contained in p^* and q^* in order to form the associated *solid strip*. We can repeat the arguments in Section 1.1 and derive that if the resulting solid strip is simply connected, its boundary contains two congruent copies of a surface γ^*. The congruency relating them is a rotation around a linear axis C^* parallel to K^* by the amount corresponding to the dihedral angle between p^* and q^*. The results in Section 1.2 also extend to the 3-dimensional case if the conditions are properly adapted.

In addition to the refinement possibilities analogous to the planar case, we can now split the tessellation of a solid stack in order to obtain multiple stacks that can be then converted into strips and arranged one next to the other. To ensure that the tessellation is face-to-face in the strip form, we assume here that the solid stack is refined only cutting by planes $h_1^*, ..., h_l^*$ orthogonal to K^*. However, this is not a necessary condition, as can be appreciated in Figure 1.

3 Discretization

This section studies how polyhedral tiles can be used to approximate the stacks constructed in previous sections. This can be thought of as simplifying the tiles one would get in the continuous case while preserving their *stackability* and congruency relations.

We assume here that a solid stack is given and that it has been refined by means of the planes $(r_0^j)^*$ through K^* and the planes h_k^* orthogonal to K^*. A possible way to discretize the tiles is to substitute each separator σ_i^* by the lower convex hull of the points resulting of its intersection with the rays $r_0^{jk} = (r_0^j)^* \cap h_k^*$. It is an easy exercise to prove that this construction preserves the topology and the congruency relations of the tiles. However, the tiles obtained in this way may not be convex, a property that can be useful in some practical applications. This restriction forces the separators to be planar within each of the stacks. This is equivalent to approximate the surface γ^* by a planar-quadrilateral mesh (PQ-mesh) $\widetilde{\gamma}$, whose vertices are constrained to lie on prescribed rays $r_i^{jk} = R_{i\alpha}^{C^*}(r_0^{jk})$. Then, the vertices V_i^{jk} of $\widetilde{\gamma}$ can be expressed as $V_i^{jk} = O_i^{jk} + \lambda_i^{jk} u_i^{jk}$, for some $\lambda_i^{jk} \in \mathbb{R}^+$, where O_i^{jk} is the initial point of r_i^{jk} and u_i^{jk} is its direction. Therefore, the vertices of $R_\alpha^{C^*}(\widetilde{\gamma})$ will also lie on these rays and their positions can be retrieved from the variables λ_i^{jk}. However, one should ensure that the perturbed versions of $\widetilde{\gamma}$ and $R_\alpha^{C^*}(\widetilde{\gamma})$ do not intersect. But provided that $\widetilde{\gamma}$ and $R_\alpha^{C^*}(\widetilde{\gamma})$ are PQ-meshes, these conditions translate into the simple linear inequalities $\lambda_i^{jl} < \lambda_{i-1}^{jl}$.

Existing algorithms, like the PQ perturbation detailed in [3], implement the planarity constraints and minimize functions measuring the distance to the surface and the quality of the mesh. Since the specific requirements of our method are, as hinted before, translated into linear equations and inequalities in the space of coordinates of the vertices, the mentioned algorithm, based on sequential quadratic programming [5], can be applied to our construction.

The results on 3-space generalize to case where p^* and q^* are parallel as well. In this situation, the vertices are constrained to lie in rays that have all the same direction. Therefore, the planarity constraints can be expressed as linear equations and optimizing with respect to linear functions can be done very efficiently by linear programming.

4 Conclusion

We give the first geometric analysis of the solids constructible with the fabrication method CASTonCAST. We restrict the study imposing some constraints to the tiles of the tessellation, which lead to simple and geometrically meaningful constructions. We also provide a scheme to approximate the tiles using polyhedra while preserving their stackability. More general cases can be studied using the introduced framework and are left to future work.

References

[1] R. Sauer, *Differenzengeometrie*, Springer, 1970.

[2] A.I. Bobenko, Y.B. Suris, *Discrete Differential Geometry: Integrable Structure*, Graduate Studies in Mathematics Series, Volume 98, AMS, 2008.

[3] Y. Liu, H. Pottmann, J. Wallner, Geometric modeling with conical meshes and developable surfaces, *ACM Trans. Graphics* **25** (2006), 681–689.

[4] Designed by L. Enrique, P. Cepaitis, D. Ordoñez, C. Piles at the Architectural Association School of Architecture, 2009.

[5] K. Madsen, H. B. Nielsen, O. Tingleff, *Optimization with Constraints, 2nd ed.*, 2004.

Improved enumeration of simple topological graphs*

Jan Kynčl† [1]

[1]Department of Applied Mathematics and Institute for Theoretical Computer Science, Charles University, Faculty of Mathematics and Physics, Malostranské nám. 25, 118 00 Praha 1, Czech Republic

Abstract

A *simple topological graph* $T = (V(T), E(T))$ is a drawing of a graph in the plane where every two edges have at most one common point (an endpoint or a crossing) and no three edges pass through a single crossing. Topological graphs G and H are *isomorphic* if H can be obtained from G by a homeomorphism of the sphere, and *weakly isomorphic* if G and H have the same set of pairs of crossing edges.

We generalize results of Pach and Tóth and the author's previous results on counting different drawings of a graph under both notions of isomorphism. We prove that for every graph G with n vertices, m edges and no isolated vertices the number of weak isomorphism classes of simple topological graphs that realize G is at most $2^{O(n^2 \log(m/n))}$, and at most $2^{O(mn^{1/2} \log n)}$ if $m \leq n^{3/2}$. As a consequence we obtain a new upper bound $2^{O(n^{3/2} \log n)}$ on the number of intersection graphs of n pseudosegments. We improve the upper bound on the number of weak isomorphism classes of simple complete topological graphs with n vertices to $2^{n^2 \cdot \alpha(n)^{O(1)}}$, using an upper bound on the size of a set of permutations with bounded VC-dimension recently proved by Cibulka and the author. We show that the number of isomorphism classes of simple topological graphs that realize G is at most $2^{m^2+O(mn)}$ and at least $2^{\Omega(m^2)}$ for graphs with $m > (6+\varepsilon)n$.

1 Introduction and the results

A *topological graph* $T = (V(T), E(T))$ is a drawing of a graph G in the plane with the following properties. The vertices of G are represented by a set $V(T)$ of distinct points in the plane and the edges of G are represented by a set $E(T)$ of simple curves connecting the corresponding pairs of points. We call the elements of $V(T)$ and $E(T)$ the *vertices* and the *edges* of T. The drawing has to satisfy the following general position conditions: (1) the edges pass through no vertices except their endpoints, (2) every two edges have only a finite number of intersection points, (3) every intersection point of two edges is either a common endpoint or a proper crossing ("touching" of the edges is not allowed), and (4) no three edges pass through the same crossing. A topological graph is *simple* if every two edges have at most one common point, which is either a common endpoint or a crossing. A topological graph is *complete* if it is a drawing of a complete graph.

We use two different notions of isomorphism to enumerate topological graphs.

Topological graphs G and H are *weakly isomorphic* if there exists an incidence preserving one-to-one correspondence between $V(G), E(G)$ and $V(H), E(H)$ such that two edges of G cross if and only if the corresponding two edges of H do.

Note that every topological graph G drawn in the plane induces a drawing G_{S^2} on the sphere, which is obtained by a standard one-point compactification of the plane. Topological graphs G and H are *isomorphic* if there exists a homeomorphism of the sphere which transforms G_{S^2} into H_{S^2}. The isomorphism can be also defined in a combinatorial way.

Unlike the isomorphism, the weak isomorphism can change the faces of the involved topological graphs, the order of crossings along the edges and also the cyclic orders of edges around vertices.

For counting the (weak) isomorphism classes, we consider all the graphs labeled. That is, each vertex is assigned a unique label from the set $\{1, 2, \ldots, n\}$, and we require the (weak) isomorphism to preserve the labels. Mostly it makes no significant difference in the results as we operate with quantities asymptotically larger than $n!$.

For a graph G, let $T_{\mathrm{w}}(G)$ be the number of weak isomorphism classes of simple topological graphs that realize G. Pach and Tóth [13] and the author [6] proved the following lower and upper bounds on $T_{\mathrm{w}}(K_n)$.

Theorem 1 [6, 13] *For the number of weak isomorphism classes of simple drawings of K_n, we have*

$$2^{\Omega(n^2)} \leq T_{\mathrm{w}}(K_n) \leq ((n-2)!)^n = 2^{O(n^2 \log n)}.$$

*The full version of the paper is available [8].

†Email: kyncl@kam.mff.cuni.cz. The author was supported by the GraDR EUROGIGA GACR project No. GIG/11/E023 and by the grant SVV-2013-267313 (Discrete Models and Algorithms). Part of the research was conducted during the Special Semester on Discrete and Computational Geometry at École Polytechnique Féderale de Lausanne, organized and supported by the CIB (Centre Interfacultaire Bernoulli) and the SNSF (Swiss National Science Foundation).

We prove generalized upper and lower bounds on $T_{\mathrm{w}}(G)$ for all graphs G.

Theorem 2 *Let G be a graph with n vertices and m edges. Then*

$$T_{\mathrm{w}}(G) \leq 2^{O(n^2 \log(m/n))}.$$

If $m < n^{3/2}$, then

$$T_{\mathrm{w}}(G) \leq 2^{O(mn^{1/2} \log n)}.$$

Let $\varepsilon > 0$. If G is a graph with no isolated vertices and at least one of the conditions $m > (1+\varepsilon)n$ or $\Delta(G) < (1-\varepsilon)n$ is satisfied, then

$$T_{\mathrm{w}}(G) \geq 2^{\Omega(\max(m, n \log n))}.$$

We also improve the upper bound from Theorem 1.

Theorem 3 *We have*

$$T_{\mathrm{w}}(K_n) \leq 2^{n^2 \cdot \alpha(n)^{O(1)}}.$$

Here $\alpha(n)$ is the inverse of the Ackermann function. It is an extremely slowly growing function, which can be defined in the following way [10]. $\alpha(m) := \min\{k : \alpha_k(m) \leq 3\}$ where $\alpha_d(m)$ is the dth function in the *inverse Ackermann hierarchy*. That is, $\alpha_1(m) = \lceil m/2 \rceil$, $\alpha_d(1) = 0$ for $d \geq 2$ and $\alpha_d(m) = 1 + \alpha_d(\alpha_{d-1}(m))$ for $m, d \geq 2$. The constant in the $O(1)$ notation in the exponent is huge (roughly 4^{30^4}), due to a Ramsey-type argument used in the proof.

In the proof of Theorem 3 we use the fact that for simple complete topological graphs, the weak isomorphism class is determined by the rotation system [7, 13]. This is combined with a Ramsey-type theorem by Pach, Solymosi and Tóth [12], which says that a simple complete topological graph with sufficiently many vertices contains a subgraph weakly isomorphic to a *convex* graph or a *twisted* graph of given size; see Figure 1. Once we have a convex graph with 5 vertices or a twisted graph with 6 vertices, we may restrict the set of possible rotations of other vertices in terms of forbidden subpermutations. The last main ingredient is a recent combinatorial result, a slightly superexponential upper bound on the size of a set of permutations with bounded VC-dimension obtained together with Josef Cibulka [4].

The method in the proof of Theorem 2 is more topological, gives a slightly weaker upper bound, but can be generalized to all graphs. Here the main tool is a construction of a *topological spanning tree* $\mathcal{T}$ of $\mathcal{G}$, which is a simply connected subset of the single topological component of $\mathcal{G}$ containing all vertices of $\mathcal{G}$ and satisfying the property that the only nonseparating points of $\mathcal{T}$ are the vertices of $\mathcal{G}$. We find such a tree consisting of $O(n)$ connected portions of edges

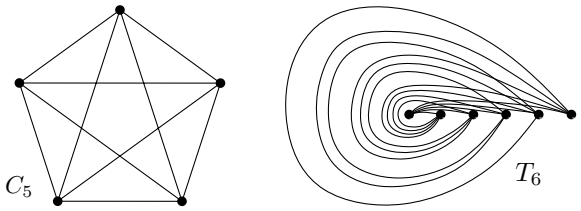


Figure 1: The convex graph C_5 and the twisted graph T_6.

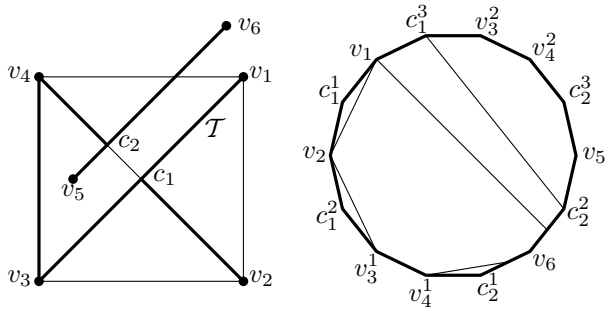


Figure 2: A topological spanning tree $\mathcal{T}$ of a simple topological graph with two components (left) and the corresponding $\mathcal{T}$-representation (right).

of $\mathcal{G}$. By cutting the plane along $\mathcal{T}$, we obtain the *$\mathcal{T}$-representation* of G, which is equivalent to a disc with at most $2mn$ chords, each chord corresponding to a portion of some edge of G. See Figure 2. We give an upper bound on the number of inequivalent $\mathcal{T}$-representations, exploiting the fact that many portions of edges do not cross.

We further generalize Theorem 3 by removing almost all topological aspects of the proof. The resulting theorem is a purely combinatorial statement, involving n-tuples of cyclic permutations avoiding a certain simple substructure.

We also consider the class of simple complete topological graphs with maximum number of crossings and suggest an alternative method for obtaining an upper bound on the number of weak isomorphism classes of such drawings.

An arrangement of *pseudosegments* (or also 1-*strings*) is a set of simple curves in the plane such that any two of the curves cross at most once. An *intersection graph of pseudosegments* (also called a *string graph of rank* 1) is a graph G such that there exists an arrangement of pseudosegments with one pseudosegment for each vertex of G and a pair of pseudosegments crossing if and only if the corresponding pair of vertices forms an edge in G. Using tools from extremal graph theory, Pach and Tóth [13] proved that the number of intersection graphs of n pseudosegments is $2^{o(n^2)}$. As a special case of Theorem 2 we

obtain the following upper bound.

Theorem 4 *There are at most $2^{O(n^{3/2}\log n)}$ intersection graphs of n pseudosegments.*

The best known lower bound for the number of (unlabeled) intersection graphs of n pseudosegments is $2^{\Omega(n\log n)}$. This follows by a simple construction or from the the fact that there are $2^{\Theta(n\log n)}$ nonisomorphic permutation graphs with n vertices.

Let $T(G)$ be the number of isomorphism classes of simple topological graphs that realize G. The following theorem generalizes the result $T(K_n) = 2^{\Theta(n^4)}$ from [7].

Theorem 5 *Let G be a graph with n vertices, m edges and no isolated vertices. Then $T(G) \le 2^{m^2+O(mn)}$. More precisely,*

$$\begin{aligned} T(G) &\le \binom{6mn}{2mn}\binom{m^2+6mn}{\frac{m^2}{2}+2mn}\cdot 2^{O(n\log n)} \\ &\le 2^{m^2+2mn(1+3\log_2 3)+O(n\log n)}, \text{ and} \\ T(G) &\le 2^{m^2+4mn}\cdot\binom{2mn+\frac{m^2}{2}}{2mn}\cdot 2^{O(n\log n)} \\ &\le 2^{m^2+2mn(\log(1+\frac{m}{4n})+2+\log_2 e)+O(n\log n)}. \end{aligned}$$

Let $\varepsilon > 0$. For graphs G with $m > (6+\varepsilon)n$ we have

$$T(G) \ge 2^{\Omega(m^2)}.$$

For graphs G with $m > \omega(n)$ we have

$$T(G) \ge 2^{m^2/60} - o(1).$$

The two upper bounds on $T(G)$ come from two essentially different approaches to enumerating isomorphism classes of $\mathcal{T}$-representations. In the first approach, we reduce the problem to enumerating *simple quadrangulations* of the disc [9]. In the second approach, we split the problem into two parts: enumerating *chord diagrams* [14] and enumerating isomorphism classes of *arrangements of pseudochords*. The first method gives better asymptotic results for dense graphs, whereas the second one is better for sparse graphs (roughly, with at most $35n$ edges). For graphs with $m = O(n)$ the second term in the exponent becomes more significant. Since $m \ge n/2$, the exponent in the first upper bound can be bounded by $23.118m^2 + o(1)$, using the entropy bound for the binomial coefficient. Similarly, the exponent in the second upper bound can be bounded by $11.265m^2+o(1)$. For such very sparse graphs (for example, matchings), however, better upper bounds can be deduced more directly from other known results.

The upper bound $T(G) \le 2^{O(m^2)}$ is trivially obtained from the upper bound on the number of unlabeled plane graphs (or planar maps). Indeed, every drawing $\mathcal{G}$ of G can be transformed into a plane graph H by subdividing the edges of $\mathcal{G}$ by its crossings and regarding the crossings of $\mathcal{G}$ as new 4-valent vertices in H. The graph H has thus at most $n + \binom{m}{2}$ vertices, at most $m + 2\binom{m}{2} = m^2$ edges, no loops and no multiple edges. Tutte [17] showed that there are

$$\frac{2(2M)!3^M}{M!(M+2)!} = 2^{(\log_2(12)+o(1))M}$$

rooted connected planar maps with M edges (see also [2, 3, 5]). Walsh and Lehman [18] showed that the number of rooted connected planar loopless maps with M edges is

$$\frac{6(4M+1)!}{M!(3M+3)!} = 2^{(\log_2(256/27)+o(1))M}.$$

This implies the upper bound $T(G) \le 2^{(\log_2(256/27)+o(1))m^2}$. Somewhat better estimates could be obtained by reducing the problem to counting 4-regular planar maps [15, 16], since typically almost all vertices in H are the 4-valent vertices obtained from the crossings of $\mathcal{G}$. But such a reduction would be less straightforward and the resulting upper bound $2^{(\frac{1}{2}\log_2(196/27)+o(1))m^2}$ still relatively high for dense graphs (for graphs with more than $27n$ edges the two upper bounds from Theorem 5 are better).

The proof in [7] implies the upper bound $T(K_n) \le 2^{(1/12+o(1))(n^4)}$, although it is not explicitly stated there. However, the key Proposition 7 in [7] is incorrect. We prove a correct version in the full paper.

Note that by the reduction to counting planar maps, for every fixed constant k, we also obtain the upper bound $2^{O(km^2)}$ on the number of isomorphism classes of connected topological graphs with m edges where all pairs of edges are allowed to cross k times.

2 A few open problems

The problem of counting the asymptotic number of "nonequivalent" simple drawings of a graph in the plane is answered only partially. Many open questions remain.

The gap between the lower and upper bounds on $T_{\mathrm{w}}(G)$ proved in Theorem 2 is wide open, especially for graphs with low density. For graphs with cn^2 edges, the lower and upper bounds on $\log T_{\mathrm{w}}(G)$ differ by a logarithmic factor. We conjecture that the correct answer is closer to the lower bound.

We do not even know whether $T_{\mathrm{w}}(G)$ is a monotone function with respect to the subgraph relation, since there are simple topological graphs that cannot be extended to simple complete topological graphs. Due to somewhat "rigid" properties of simple complete topological graphs, we have a much better upper bound for the complete graph than, say, for the complete bipartite graph on the same number of vertices.

Problem 1 *Does the complete graph K_n maximize the value $T_w(G)$ among the graphs G with n vertices? More generally, is it true that $T_w(H) \leq T_w(G)$ if $H \subseteq G$?*

Our methods for proving upper bounds on the number of weak isomorphism classes of simple topological graphs do not generalize to the case of topological graphs with two crossings per pair of edges allowed.

Problem 2 *What is the number of weak isomorphism classes of drawings of a graph G where every two independent edges are allowed to cross at most twice and every two adjacent edges at most once?*

For the complete graph with n vertices, Pach and Tóth [13] proved the lower bound $2^{\Omega(n^2 \log n)}$ and the upper bound $2^{o(n^4)}$.

A nontrivial lower bound can be proved also in the case when G is a matching. Ackerman et al. [1] constructed a system of n x-monotone curves where every pair of curves intersect in at most one point where they either cross or touch, with $\Omega(n^{4/3})$ pairs of touching curves. Eyal Ackerman (personal communication) noted that this also follows from an earlier result by Pach and Sharir [11], who constructed an arrangement of n segments with $\Omega(n^{4/3})$ vertically visible pairs of disjoint segments. By changing the drawing in the neighborhood of every touching point, we obtain $2^{\Omega(n^{4/3})}$ different intersection graphs of 2-intersecting curves, also called *string graphs of rank* 2 [13]. This improves the trivial lower bound observed by Pach and Tóth [13].

Acknowledgements

The author thanks Josef Cibulka for discussions about enumerating various combinatorial objects.

References

[1] E. Ackerman, R. Pinchasi and S. Zerbib, On touching curves, *Bernoulli Reunion Conference on Discrete and Computational Geometry*, EPFL, Lausanne, 2012.

[2] E. A. Bender and L. B. Richmond, A survey of the asymptotic behaviour of maps, *Journal of Combinatorial Theory, Series B* **40**(3) (1986), 297–329.

[3] E. A. Bender and N. C. Wormald, The number of loopless planar maps, *Discrete Mathematics* **54**(2) (1985), 235–237.

[4] J. Cibulka and J. Kynčl, Tight bounds on the maximum size of a set of permutations with bounded VC-dimension, *Journal of Combinatorial Theory, Series A* **119**(7) (2012), 1461–1478.

[5] M. Drmota and M. Noy, Universal exponents and tail estimates in the enumeration of planar maps, *Electronic Notes in Discrete Mathematics* **38** (2011), 309–317.

[6] J. Kynčl, Crossings in topological graphs, master thesis, Charles University, Prague, 2006.

[7] J. Kynčl, Enumeration of simple complete topological graphs, *European Journal of Combinatorics* **30**(7) (2009), 1676–1685.

[8] J. Kynčl, Improved enumeration of simple topological graphs, submitted. Manuscript available at arXiv:1212.2950.

[9] R.C. Mullin and P.J. Schellenberg, The enumeration of c-nets via quadrangulations, *Journal of Combinatorial Theory* **4**(3) (1968), 259–276.

[10] G. Nivasch, Improved bounds and new techniques for Davenport–Schinzel sequences and their generalizations, *Journal of the ACM* **57**(3) (2010), 1–44.

[11] J. Pach and M. Sharir, On vertical visibility in arrangements of segments and the queue size in the Bentley-Ottmann line sweeping algorithm, *SIAM Journal on Computing* **20**(3) (1991), 460–470.

[12] J. Pach, J. Solymosi and G. Tóth, Unavoidable configurations in complete topological graphs, *Discrete and Computational Geometry* **30** (2003), 311–320.

[13] J. Pach and G. Tóth, How many ways can one draw a graph?, *Combinatorica* **26**(5) (2006), 559–576.

[14] R. C. Read, The chord intersection problem, *Annals of the New York Academy of Sciences* **319**(1) (1979), 444–454.

[15] H. Ren and Y. Liu, Enumerating near-4-regular maps on the sphere and the torus, *Discrete Applied Mathematics* **110**(2–3) (2001), 273–288.

[16] H. Ren, Y. Liu and Z. Li, Enumeration of 2-connected Loopless 4-regular maps on the plane, *European Journal of Combinatorics* **23**(1) (2002), 93–111.

[17] W. T. Tutte, A census of planar maps, *Canadian Journal of Mathematics* **15** (1963), 249–271.

[18] T. R. S. Walsh and A. B. Lehman, Counting rooted maps by genus III: Nonseparable maps, *Journal of Combinatorial Theory, Series B* **18** (1975), 222–259.

On three parameters of invisibility graphs *

Josef Cibulka[†1], Miroslav Korbelář[‡2], Jan Kynčl[§1], Viola Mészáros[¶3], Rudolf Stolař[‖1], and Pavel Valtr[**1]

[1]Department of Applied Mathematics and Institute for Theoretical Computer Science, Charles University, Faculty of Mathematics and Physics, Malostranské nám. 25, 118 00 Prague, Czech Republic
[2]Department of Mathematics and Statistics, Faculty of Science, Masaryk University, Kotlářská 2, 611 37 Brno, Czech Republic
[3]Institute for Mathematics, Technical University of Berlin, Strasse des 17. Juni 136, 10623 Berlin, Germany

Abstract

The invisibility graph $I(X)$ of a set $X \subseteq \mathbb{R}^d$ is a (possibly infinite) graph whose vertices are the points of X and two vertices are connected by an edge if and only if the straight-line segment connecting the two corresponding points is not fully contained in X. We consider the following three parameters of a set X: the clique number $\omega(I(X))$, the chromatic number $\chi(I(X))$ and the minimum number $\gamma(X)$ of convex subsets of X that cover X.

We settle a conjecture of Matoušek and Valtr claiming that for every planar set X, $\gamma(X)$ can be bounded in terms of $\chi(I(X))$. As a part of the proof we show that a disc with n one-point holes near its boundary has $\chi(I(X)) \geq \log\log(n)$ but $\omega(I(X)) = 3$.

We also find sets X in $\mathbb{R}^5$ with $\chi(I(X)) = 2$, but $\gamma(X)$ arbitrarily large.

1 Introduction

Let X be a subset of a d-dimensional Euclidean space. We say that two points $x, y \in X$ *see each other* if the straight-line segment $\overline{xy}$ connecting x and y is a subset of X. The *invisibility graph* $I(X)$ of a set $X \subseteq \mathbb{R}^d$ is a graph whose vertices are the points of X and two vertices are connected by an edge if and only if they do not see each other. Let $\chi(G)$ be the chromatic number of a graph G and let $\omega(G)$ be its clique number. For a set $X \subseteq \mathbb{R}^d$ we define $\gamma(X)$ to be the minimum possible number of convex subsets of X that cover X. Further, let $\chi(X) := \chi(I(X))$ and $\omega(X) := \omega(I(X))$.

Sets X with $\omega(X) = n - 1$ are sometimes called *n-convex* [8].

Observe that $\omega(X) \leq \chi(X) \leq \gamma(X)$ for any set X.

If a planar set X is closed, then $\gamma(X)$ can be bounded by a function of $\omega(X)$. This was proved by Breen and Kay [2] and the current best known upper bound is $\gamma(X) \leq O(\omega(X)^3)$ by Matoušek and Valtr [6]. From the other direction, there exist examples by Matoušek and Valtr [6] with $\gamma(X) \geq \Omega(\omega(X)^2)$.

However, if we don't restrict ourselves to closed sets, there is no upper bound on $\gamma(X)$ even for sets with $\omega(X) = 3$. An example is the disc D_λ with λ one-point holes punctured in the vertices of a regular convex λ-gon near the boundary of D_λ, for which $\omega(D_\lambda) = 3$, but $\gamma(D_\lambda) = \lceil \lambda/2 \rceil + 1$ (see [6]).

A *one-point hole* in a set $X \subset \mathbb{R}^d$ is a point that forms a path-connected component of $\mathbb{R}^d \setminus X$. Let $\lambda(X)$ be the number of one-point holes in the set X.

The example of the set D_λ led to studying the properties of planar sets with a limited number of one-point holes by Matoušek and Valtr [6]. In particular, they proved the following theorem.

Theorem 1 (Matoušek and Valtr [6]) *Let $X \subseteq \mathbb{R}^2$ be a set with $\omega(X) = \omega < \infty$ and $\lambda(X) = \lambda < \infty$. Then*

$$\gamma(X) \leq O(\omega^4 + \lambda\omega^2).$$

For any $\omega \geq 3$ and $\lambda \geq 0$ they also found sets X with $\omega(X) = \omega$, $\lambda(X) = \lambda$ and $\gamma(X) \geq \Omega(\omega^3 + \omega\lambda)$.

Matoušek and Valtr [6] conjectured that for an arbitrary planar set X, the value of $\gamma(X)$ is bounded by a function of $\chi(X)$. Then $\chi(X)$ cannot be bounded

*This research was supported by the project CE-ITI (GACR P202/12/G061) of the Czech Science Foundation and by the grant SVV-2013-267313 (Discrete Models and Algorithms). The second author was supported by the project CZ.1.07/2.3.00/20.0003 of the Operational Programme Education for Competitiveness of the Ministry of Education, Youth and Sports of the Czech Republic. The fourth author was also partially supported by ESF EuroGiga project ComPoSe (IP03), by OTKA Grant K76099 and by OTKA Grant 102029. The first, the third and the sixth author were partially supported by project GAUK 52410. Part of the research was conducted during the Special Semester on Discrete and Computational Geometry at École Polytechnique Féderale de Lausanne, organized and supported by the CIB (Centre Interfacultaire Bernoulli) and the SNSF (Swiss National Science Foundation).

[†]Email: cibulka@kam.mff.cuni.cz.
[‡]Email: miroslav.korbelar@gmail.com.
[§]Email: kyncl@kam.mff.cuni.cz.
[¶]Email: meszaros@math.tu-berlin.de.
[‖]Email: ruda@kam.mff.cuni.cz.
[**]Email: valtr@kam.mff.cuni.cz.

by a function of $\omega(X)$ as shows the above example with D_λ.

Lawrence and Morris [4] proved that for every k there exists $n_0(k)$ such that whenever S is a set of finitely many points in the plane and $|S| \geq n_0(k)$, then $\chi(\mathbb{R}^2 \setminus S) \geq k$. [1] Thus, whenever X is the complement of a finite set of points, $\lambda(X)$ can be bounded in terms of $\chi(X)$. This implies, by Theorem 1, that the value of $\gamma(X)$ can be bounded in terms of $\chi(X)$, settling the conjecture of Matoušek and Valtr in the special case when X is a complement of a finite set of points.

In this paper, we strengthen the result of Lawrence and Morris [4] and settle the conjecture for every planar set X.

The *tower function* $T_l(k)$ is defined recursively as $T_0(k) = k$ and $T_h(k) = 2^{T_{h-1}(k)}$. Its inverse is the iterated logarithm $\log^{(l)}(n)$, that is, $\log^{(0)}(n) = n$ and $\log^{(l)}(n) = \log(\log^{(l-1)}(n))$.

Theorem 2 *Any set $X \subseteq \mathbb{R}^2$ with $\chi(X) = \chi < \infty$ satisfies*

$$\gamma(X) \leq O(2^{4T_2(\chi)} \cdot \chi^3).$$

The proof of Theorem 2 is omitted from this extended abstract. In the full version of the paper, we also show that for every dimension d, $\lambda(X)$ can be bounded in terms of $\chi(X)$ for sets $X \subset \mathbb{R}^d$. This answers Question 6 of Lawrence and Morris [4].

A set X is *star-shaped* if X contains a point $x \in X$ that sees every other point of X.

In Sections 2 and 3 we show that χ and γ can be separated in dimensions 5 and more.

Theorem 3 *For every positive integer g there exist star-shaped sets*

1. *$X \subset \mathbb{R}^6$ satisfying $\chi(X) = 2$ and $\gamma(X) \geq g$ and*
2. *$X_c \subset \mathbb{R}^6$ that is closed and satisfies $\chi(X_c) = 4$ and $\gamma(X_c) \geq g$.*

Theorem 4 *For every positive integer g there exist star-shaped sets*

1. *$X \subset \mathbb{R}^5$ satisfying $\chi(X) = 2$ and $\gamma(X) \geq g$ and*
2. *$X_c \subset \mathbb{R}^5$ that is closed and satisfies $\chi(X_c) = 6$ and $\gamma(X_c) \geq g$.*

Problem 1 *Does there exist a function f such that*

$$\gamma(X) \leq f(\chi(X)),$$

1. *for every set $X \subseteq \mathbb{R}^3$?*
2. *for every set $X \subseteq \mathbb{R}^4$?*

All logarithms in this paper are binary. We use the notation $\overline{xy}$ for the straight line segment between points x and y.

[1] The graph $\mathcal{G}_S$ in the paper of Lawrence and Morris is precisely the invisibility graph of $\mathbb{R}^2 \setminus S$.

2 Constructions in dimension 6

2.1 Set with chromatic number 2

We prove part 1 of Theorem 3. Part 2 is omitted from this extended abstract.

Let P_n be the *cyclic polytope* on $n \geq 7$ vertices (see for example [5]) and V_n its set of vertices. Thus the convex hull of every triple of points from V_n is a triangular face of P_n.

Lemma 5 *Let K_n be the complete graph on the set V of $n \geq 7$ vertices and let $k := \lceil 2\log(n) + 2 \rceil$. It is possible to orient the edges of K_n so that every set $V' \subseteq V$ of size at least k contains a directed triangle.*

Proof. For brevity, we call a set V' *good* if it contains a directed triangle.

We orient the edges randomly and show that with positive probability, every set $V' \subseteq V$ of size at least k is good.

First, we will bound the probability b_k that a given set V' of k vertices is bad.

If there exists a directed cycle on V' of length greater than 3, then one of the two cycles created by adding an arbitrary diagonal to the cycle is again directed. Thus there exists a directed triangle on V'.

There are $2^{k(k-1)/2}$ possible orientations of edges of a complete graph on k vertices out of which $k!$ are acyclic. Thus

$$b_k = \frac{k!}{2^{k(k-1)/2}} = k! 2^{-k^2/2+k/2}.$$

The probability that some k-tuple V' of vertices is bad is thus at most

$$\begin{aligned}\binom{n}{k} b_k \leq \frac{n^k}{k!} b_k = 2^{k\log(n) - k^2/2 + k/2} = \\ = 2^{k(\log(n) - k/2 + 1/2)} \leq \\ \leq 2^{k(\log(n) - \log(n) - 1 + 1/2)} \leq 2^{k(-1/2)} < 1.\end{aligned}$$

□

We fix the orientation of the edges of P_n in which every k-tuple of vertices is good. A triangular face of P_n has *directed boundary* if the three edges of the face form a directed cycle. The set X is constructed by puncturing a one-point hole in the barycenter of triangular faces of P_n with directed boundary.

Vertices of P_n are colored black. Edges are cut in thirds. In every edge, the interior of the middle third together with the point at one third closer to the end of the edge is colored white. The rest of the edges is black. The coloring of triangular faces with directed boundary is depicted on Fig. 1. The rest of X is colored black.

All the edges of the invisibility graph of X are between pairs of points lying on the same triangular face

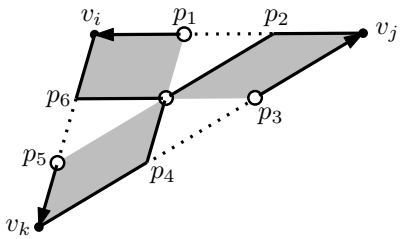


Figure 1: Coloring of a face with directed boundary. Lines determined by pairs (p_1, p_4), (p_2, p_5) and (p_3, p_6) intersect in the barycenter and split the triangle into monochromatic regions. Full lines and gray areas represent black color, the rest is white.

with directed boundary. The coloring is proper on each of these faces and thus the 2-coloring of the whole X is proper.

If a convex set C contains at least k vertices of X, then it contains a triangular face with directed boundary and thus C contains a one-point hole. Therefore

$$\gamma(X) \geq \frac{n}{2\log(n)+3}.$$

3 Constructions in dimension 5

Here we present some parts of the proof of part 1 of Theorem 4. The rest of the proof is omitted from this extended abstract.

The constructions are similar to those in dimension 6: for part (1) of the theorem, the set X is a closed cyclic polytope with one-point holes in some of the 2-dimensional faces. For part (2), instead of points we remove small 5-dimensional polytopes attached to the 2-dimensional faces. The difference from the construction in dimension 6 is in the placement of the holes: here we cannot apply the same argument as in the previous section since for the cyclic polytope in dimension 5 only quadratically many triples of vertices induce a 2-dimensional face and there is a 2-coloring of the vertex set in which no 2-dimensional face is monochromatic.

In the full version of the paper, we show two different ways how to choose the holes. In the first construction we essentially show that randomly chosen holes will do, but the proof (interestingly) requires a rather nontrivial result from group theory. Also the construction proves only part (1) of the theorem. In the second construction we specify the locations of the holes precisely. Moreover, we show that the holes can be enlarged to open pyramids, which shows part (2) of the theorem. The proof of part (2) of the theorem is analogous to the proof of part (2) of Theorem 3 and is omitted from this abstract.

Let P_n be the 5-dimensional cyclic polytope on $n \geq 6$ vertices with (ordered) vertex set $V_n = \{v_1, v_2, \ldots, v_n\}$. For brevity, we call the triangular face with vertices v_i, v_j and v_k the ijk triangle. Similarly, the ij edge is the edge between vertices v_i and v_j. The 2-dimensional faces of P_n are the triangles

- $1ij$ for every $1 < i < j \leq n$ (type $1ij$ triangles),
- ijn for every $1 \leq i < j < n$ (type ijn triangles),
- $i(i+1)j$ for every $1 < i < j-1 < n$ (type $i(i+1)j$ triangles) and
- $ij(j+1)$ for every $1 < i < j < n-1$ (type $ij(j+1)$ triangles).

3.1 Covering with convex sets

In the constructions proving part (1) of Theorem 4, we remove a one-point hole from every type $1ij$ triangle. In the construction proving part (2), we remove an open flat simplex instead of the point (as in Section 2). The following lemma shows that in both cases, the resulting set can not be covered by a bounded number of convex sets.

Lemma 6 *Let X be a subset of P_n such that every edge of P_n is a subset of X and none of the type $1ij$ triangles is a subset of X. Then $\gamma(X) \geq \Omega(\log n / \log\log n)$.*

Proof. Let $X = C_1 \cup C_2 \cup \cdots \cup C_k$ be a covering of X with convex subsets of X. The covering induces a partition of each open edge $1i$, $2 \leq i \leq n$, into at most k intervals $I_i^1, I_i^2, \ldots, I_i^{k_i}$, where each of the intervals I_i^j is covered by one of the convex sets $C_{l(i,j)}$. Since the convex sets in the covering may overlap, this partition need not be unique; in such a case we just pick one.

We say that the partitions of two edges $1i$ and $1i'$ are of the same *type* if $k_i = k_{i'}$, $l(i,p) = l(i',p)$ for each $p = 1, 2, \ldots, k_i$ (in other words, the "colors" appear in the same order along the edges), and for each $p = 1, 2, \ldots, k_i$ the type of the interval I_i^p (that is, closed, open, or half-closed from the left/right) is the same as the type of the interval $I_{i'}^p$. Degenerate one-point intervals are considered as closed. The number of types of the partitions is at most $2^k \cdot k! \cdot 2^{k-1}$. Indeed, there are at most 2^k subsets of "colors", each of the subsets can be linearly ordered in at most $k!$ ways, and there are at most $k-1$ boundary points shared by two intervals, where one of the intervals is locally closed and the other one locally open.

It follows that if $n > 2^k \cdot k! \cdot 2^{k-1} + 1$, then there are two edges $1i$ and $1i'$ of the same type. The convex hulls $\mathrm{conv}(I_i^p \cup I_{i'}^p)$ cover the whole open triangle $1ii'$, including the one-point hole inside, which is a contradiction. Therefore $n \leq 2^k \cdot k! \cdot 2^{k-1} + 1$, which implies that $\gamma(X) \geq \Omega(\log n / \log\log n)$. □

4 Concluding remarks

To solve Problem 1 in dimension 4 we could use a construction similar to those in dimensions 5 and 6 provided the following problem has a positive answer.

Problem 2 *Does there exist for every k a convex simplicial polytope $P(k)$ in $\mathbb{R}^4$ such that in every coloring of vertices of $P(k)$ by k colors we can find a triangular face whose vertices are monochromatic?*

Assuming the polytope $P(k)$ from Problem 2 exists, the set X from Problem 1 is obtained from $P(k)$ by making a one-point hole in an arbitrary point inside every triangular face. Such a set X cannot be covered by k convex sets since otherwise one of the convex sets would contain three vertices of a triangular face.

The invisibility graph $I(X)$ can be colored by 13 colors in the following way. All the vertices of $P(k)$ get color 1. Tancer [7] has shown that the edges of every 2-dimensional simplicial complex PL-embeddable in $\mathbb{R}^3$ can be colored by 12 colors so that for every triangular face, the three edges on its boundary have three different colors. This applies, in particular, to the 2-skeleton of every 4-dimensional convex simplicial polytope. We use colors $2, 3, \ldots, 13$ to color the interiors of edges of $P(k)$ in this way. For each triangular face and each point p on its boundary, the interior of the segment conecting the one-point hole with p is colored by the color of p. All the remaining points of X are isolated in $I(X)$ and thus may be colored arbitrarily.

The boundary complex of a 4-dimensional convex simplicial polytope is a special case of a triangulation of S^3. If we relax the condition on polytopality in Problem 2 and ask only for a triangulation of S^3, then the answer is yes. Heise et al. [3] constructed, for every k, a 2-dimensional simplicial complex linearly embedded in $\mathbb{R}^3$ such that in every coloring of its vertices with k colors at least one of the triangles is monochromatic. We found the same simplicial complex independently, modifying Boris Bukh's construction, which was communicated to us by Martin Tancer. The vertices of the complex are placed on the moment curve and a suitable noncrossing subset of triangles is chosen for the faces. It remains to extend the embedded complex to a triangulation of the whole $\mathbb{R}^3$, or S^3 [1].

References

[1] K. Adiprasito, F. Lutz and J. Moller, unpublished manuscript.

[2] M. Breen and D. C. Kay, General decomposition theorems for m-convex sets in the plane, *Israel Journal of Mathematics* 24 (1976), 217–233.

[3] C. G. Heise, K. Panagiotou, O. Pikhurko and A. Taraz, Coloring d-embeddable k-uniform hypergraphs, arXiv:1209.4879 (2012).

[4] J. Lawrence and W. Morris, Finite sets as complements of finite unions of convex sets, *Discrete & Computational Geometry* 42 (2009), no. 2, 206–218.

[5] J. Matoušek, *Lectures on Discrete Geometry,* Springer-Verlag, New York (2002).

[6] J. Matoušek and P. Valtr, On visibility and covering by convex sets, *Israel Journal of Mathematics* 113 (1999), no. 3, 341–379.

[7] M. Tancer, unpublished manuscript.

[8] F. Valentine, A three point convexity property, *Pacific Journal of Mathematics* 7 (1957), no. 2, 1227–1235.

On making a graph crossing-critical

César Hernández–Vélez*[1] and Jesús Leaños†[2]

[1]Instituto de Física, Universidad Autónoma de San Luis Potosí, San Luis Potosí, México 78000
[2]Unidad Académica de Matemáticas, Universidad Autónoma de Zacatecas, Zacatecas, México 98060

Abstract

A graph is *crossing-critical* if its crossing number decreases when we remove any of its edges. Recently it was proved that if a non-planar graph G is obtained by adding an edge to a cubic polyhedral (planar 3-connected) graph, then G can be made crossing-critical by a suitable multiplication of its edges. Here we show: (i) a new family of graphs that can be transformed into crossing-critical graphs by a suitable multiplication of its edges, and (ii) a family of graphs that cannot be made crossing-critical by any multiplication of its edges.

Introduction

This work is motivated by the following question settled by Beaudou et al. in [1]: to what extent is crossing-criticality a property that is inherent to the structure of a graph, and to what extent can it be induced on a non crossing-critical graph by multiplying (all or some of) its edges? In [1] a family of non crossing-critical graphs are transformed into crossing-critical graphs by *multiplying* (adding parallel) edges.

The use of parallel edges has been essential in many other important results on crossing number. For example, in [2] was proved that for every $a > b > 0$, there exist a graph G with crossing number a in the plane, crossing number b in the torus, and crossing number 0 in the double torus. This is called the orientable crossing sequence of a graph G. In [4] was reported a conjecture of R. B. Richter which states that crossing-critical graphs have bounded maximum degree. This conjecture was disproved by Dvořák and Mohar [3], who exhibited crossing-critical graphs with large maximum degree. In all these papers the use of weighted (also called "thick") edges is essential.

No simple graph that satisfies the properties as defined in [2] and [3] are not known. On the other hand there exist simple non crossing-critical graphs such that if we multiply its edges (or equivalently, assign weights on them) then they become crossing-critical, as in [1]. Thus, important questions remain: What makes a graph crossing-critical? Does every non-planar graph become crossing-critical by a weight assignment to its edge set?

In this paper we give two infinite families $\mathcal{G}$ and $\mathcal{G}'$ of graphs such that (i) every graph in $\mathcal{G}$ remains non crossing-critical even after any multiplications of its edges, and (ii) every graph in $\mathcal{G}'$ is not crossing-critical, but after a suitable multiplication of its edges, it is transformed into a crossing-critical graph. Let us proceed to define these families.

Let W_n be the wheel with $n+1$ vertices, $n \geq 5$, and let v_0 be the degree n vertex. The remaining n vertices are labeled $v_1, v_2, \ldots, v_n$ in the order in which they appear in the n-cycle. We add a new vertex u to W_n which is joined to vertices v_0, v_1 and v_3 and denote by G_n the resulting graph. Let G'_n be the graph that results by removing the edges v_0v_1 and v_0v_3 from G_n. We define $\mathcal{G} := \{G_n : n \geq 5\}$ and $\mathcal{G}' := \{G'_n : n \geq 5\}$. See Figure 1. Note that each graph of $\mathcal{G} \cup \mathcal{G}'$ is 3-connected.

It is not difficult to find non-planar graphs which cannot be made crossing-critical by any multiplication of its edges. For example, if G is a non-planar graph and e is a cut edge of G, then G cannot be made crossing-critical by any multiplication of its edges because e cannot be made critical. Thus, some connectivity assumption is needed in order to guarantee that a non-planar graph can be made crossing-critical. In [1], for example, the graphs under consideration are assumed to be internally 3–connected and such an assumption plays a central role in its work. On the other hand, as we will see in Theorem 1, the family $\mathcal{G}$ is interesting because shows that 3-connectedness property by itself is not sufficient to ensure that a non-planar graph can be made crossing-critical.

Our main results are the following.

Theorem 1 *Any graph $G_n \in \mathcal{G}$ is not crossing-critical. Moreover, G_n cannot be made crossing-critical by any multiplication of its edges.*

Theorem 2 *Every graph $G'_n \in \mathcal{G}'$ is not crossing-critical, but there exists a suitable multiplication of its edges such that the resulting graph $\overline{G'_n}$ is crossing-critical.*

*Email: cesar@ifisica.uaslp.mx.
†Email: jleanos@matematicas.reduaz.mx.

An edge e of a graph G is a *Kuratowski edge* if there is a subgraph H of G which is isomorphic to a subdivision of a K_5 or $K_{3,3}$ and e lies on H. In [1] was conjectured that a graph whose edges are all Kuratowski becomes crossing-critical after a suitable multiplication of its edges. In this sense, note that the graphs in $\mathcal{G}'$ are consistent with this conjecture because any edge in G'_n is a Kuratowski edge (unlike the edges v_0v_1 and v_0v_3 in G_n).

1 Multigraphs and weighted graphs

We recall that the *crossing number* $\mathrm{cr}(G)$ of a graph G is the minimum number of pairwise intersections of edges in a drawing of G in the plane. An edge e of G is *crossing-critical* if $\mathrm{cr}(G-e) < \mathrm{cr}(G)$, and say that G is *crossing-critical* if all its edges are crossing-critical.

Recall that a *weighted graph* is a pair (G, w) where G is a graph and w is a function (the *weight assignment*) that assigns to each edge e of G a number $w(e)$ (the *weight*). The weight assignment is *positive* (respectively, *integer*) if $w(e)$ is a positive (respectively, integer) number for any edge e of G. We only consider positive integer weight assignments.

We extend the concept of crossing number to weighted graphs (G, w) in an analogous way, just taking to account that a crossing between the edges e and e' contributes $w(e) \cdot w(e')$ to $\mathrm{cr}\,(G, w)$. A drawing $\mathcal{D}$ of (G, w) is *optimal* if $\mathrm{cr}(\mathcal{D}) = \mathrm{cr}(G, w)$.

We now proceed to define what a crossing-critical edge is in a weighted graph. Let (G, w) be a weighted graph, an edge e of (G, w) is *crossing-critical* if $\mathrm{cr}\,(G, w_e) < \mathrm{cr}\,(G, w)$, where the weight assignment w_e is defined by,

$$w_e(f) = \begin{cases} w(f) & \text{if } f \neq e, \\ w(f) - 1 & \text{if } f = e. \end{cases}$$

As usual, (G, w) is *crossing-critical* if all its edges are crossing-critical.

Let G^* be a multigraph, and let G be its underlying simple graph. We define the *associated weighted graph* (G, w^*) of G^* as follows: for every edge e of G, we define $w^*(e)$ as the multiplicity of e in G^*. The following observation is straightforward.

Remark 1 *Let G^* be a multigraph and let (G, w^*) be its associated weighted graph. Then G^* is crossing-critical if and only if (G, w^*) is crossing-critical.*

2 Proofs of Theorems 1 and 2

For brevity, let (i) $\alpha_0 := v_0u$ and $\alpha_i := v_0v_i, i = 1, \ldots, n$; (ii) $\beta_i := v_iv_{i+1}$, $i = 1, \ldots, n$ with $v_{n+1} = v_1$;

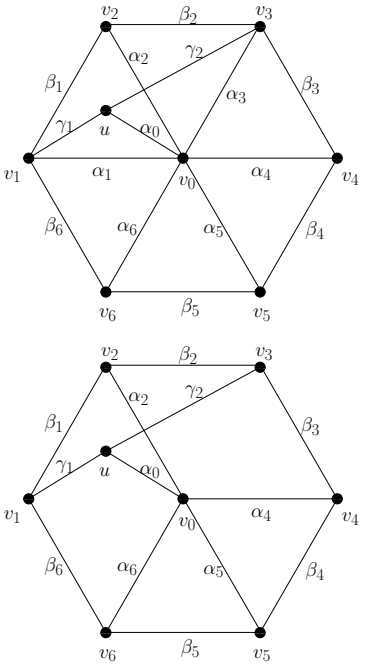


Figure 1: A drawing $\mathcal{D}$ of G_6 (above), and a drawing $\mathcal{D}'$ of G'_6 (below).

and (iii) $\gamma_1 := uv_1$ and $\gamma_2 := uv_3$. Under this notation, observe that $G'_n = G_n - \{\alpha_1, \alpha_3\}$. See Figure 1.

Lemma 3 *Let w' be a positive integer weight assignment on G'_n. Then there exists an optimal drawing of (G'_n, w') in which we can add both edges α_1 and α_3 without increasing the crossing number.*

Proof. Let $\mathcal{D}'$ be an optimal drawing of (G'_n, w'). We divide the proof according to the edges that are involved in a crossing in $\mathcal{D}'$.

(A) Suppose that α_0 is crossed in $\mathcal{D}'$ (analogously for α_2).

Since $\mathcal{D}'$ is an optimal drawing, α_0 does not cross with adjacent edges, therefore α_0 crosses with a β-edge, say β_j. Let $\mathcal{D}^*$ be the drawing defined as follows. Draw a simple regular n-polygon such that $v_1, v_2, \ldots, v_n$ (in that order) are its vertices, place v_0 in the center of such a polygon and add the α_i-edge, $i = 2, 4, 5, \ldots, n$, as straight line segments. Now draw α_0 as an straight line segment crossing the edge β_j. Finally the edges γ_1 and γ_2 can be added around the n-polygon boundary without introducing new crossings. So $\mathcal{D}^*$ has just one crossing and therefore $\mathrm{cr}(\mathcal{D}^*) \leq \mathrm{cr}(\mathcal{D}')$. But since $\mathcal{D}'$ is an optimal drawing, we must have that $\mathrm{cr}(\mathcal{D}^*) = \mathrm{cr}(G', w')$. Moreover, we can add the edges α_0 and α_3 as straight line segments without adding a new crossing, as required.

(B) Both edges in any of $\{\beta_1, \beta_2\}$, $\{\gamma_1, \gamma_2\}$, $\{\beta_1, \gamma_2\}$, $\{\gamma_1, \beta_2\}$ are not crossed in $\mathcal{D}'$.

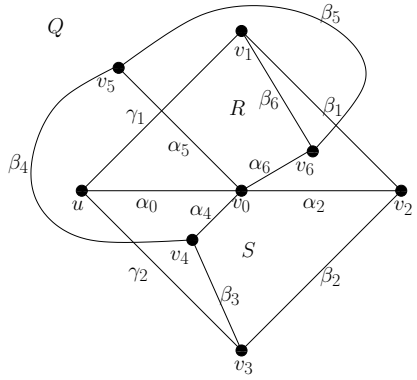


Figure 2: Drawing of G'_6 where γ_1, γ_2, β_1, β_2, α_0 and α_2 form a plane $K_{2,3}$

We only analyze the case in which both edges of $\{\beta_1, \gamma_2\}$ are not crossed in $\mathcal{D}'$ (the rest of the cases can be verified similarly). By (A) we can assume that neither α_0 nor α_2 are crossed in $\mathcal{D}'$. Because none of β_1 and γ_2 has a cross in $\mathcal{D}'$, we can add α_1 (respectively α_3) following the uncrossed path $\alpha_2\beta_1$ (respectively $\alpha_0\gamma_2$) without introducing new crossings.

(C) Suppose β_1 crosses γ_2 in $\mathcal{D}'$ (analogously, β_2 crosses γ_1).

Draw a plane wheel with vertices $v_1, \ldots, v_n$ forming a n-cycle and the vertex v_0 as the center. All edges α_i, $i \neq 1, 3$ are the spokes of this wheel. Then draw the path $\alpha_0\gamma_1$ as a spoke. Finally add the edge γ_2 only crossing the edge β_1. Since the number of crossings of such a drawing is less than or equal to the number of crossings of $\mathcal{D}'$, it is optimal. In this new drawing we can add α_1 and α_3 without increasing the crossing number.

By (A) and (C) we can assume that neither γ_1, γ_2, β_1, β_2, α_0 nor α_2 cross each other. Thus γ_1, γ_2, β_1, β_2, α_0 and α_2 form a plane $K_{2,3}$. Without any loss of generality, we may assume that v_0 is in the bounded region defined by the cycle $\gamma_1\beta_1\beta_2\gamma_2$. Let R, S and Q denote the disjoint regions bounded by the cycles $\alpha_0\alpha_2\beta_1\gamma_1$, $\alpha_0\gamma_2\beta_2\alpha_2$ and $\gamma_1\gamma_2\beta_2\beta_1$, respectively. (See Figure 2)

Let P be the path $\beta_3\beta_4 \ldots \beta_n$. Let μ, ν, and ρ be the element of $\{\beta_1, \gamma_1\}$, $\{\beta_2, \gamma_2\}$, and $\{\beta_1, \beta_2, \gamma_1, \gamma_2\}$ with less weight, respectively.

Now we proceed according to the interior vertices of P that are in each of R, S and Q.

(D) No interior vertices of P are in Q.

Then, all the interior vertices of P are in R, S, or both of them.

(D.i) All interior vertices of P are in R (analogously for S). Since $\mathcal{D}'$ is optimal, β_3 is the only edge that cross the cycle $\alpha_0\alpha_2\beta_1\gamma_1$. By (A), β_3 does not cross neither α_0 nor α_2, thus, by optimality, β_3 only crosses with μ. Then this case follows from (B).

(D.ii) Both R and S contain interior vertices of P. Because Q has no interior vertices of P, at least one β-edge of P goes from R to S. Moreover, such a β-edge is neither β_3 nor β_n. Let β_i be $(i \neq 3, n)$, the β-edge of P with less weight. Thus we can get a drawing $\mathcal{D}^*$ where the only crossings are those involving β_i with μ and β_i with ν. Since $\mathcal{D}'$ is optimal, $\mathrm{cr}(\mathcal{D}^*) = \mathrm{cr}(\mathcal{D}')$. By (B), we can add the edges α_1 and α_3 without increasing the crossing number.

(E) Some interior vertices of P are in Q.

In this case we have three possibilities.

(E.i) All interior vertices of P are in Q. In this case we can get a drawing with as many crossings as $\mathcal{D}'$ in which every crossing involves ρ and some α_i, with $i = 4, 5, \ldots, n$. This case follows from (B).

(E.ii) Each region R, S and Q contains interior vertices of P. Then there must be a β-edge, say β_i (respectively β_j), crossing the cycle that bounds the region R (respectively S). By (A), β_i (respectively β_j) must cross either γ_1 or β_1 (respectively γ_2 or β_2). If $w'(\beta_i) \leq w'(\beta_j)$, we can get an optimal drawing $\mathcal{D}^*$ by putting all vertices $v_4, v_5, \ldots, v_i$ in S, all vertices $v_{i+1}, v_{i+2}, \ldots, v_n$ in R. Thus we are back in case (D.ii). If $w'(\beta_j) \leq w'(\beta_i)$ we can proceed analogously.

(E.iii) Both R and Q contain (all the) interior vertices of P (analogously for S and Q). Let v_i be the interior vertex of P in R with the smallest index. If v_4 is in R, then the edge β_3 crosses the cycle $\alpha_0\alpha_2\beta_1\gamma_1$ and, by putting all interior vertices of P in R, we can proceed as in (D.i). Thus v_4 must be in Q and so $i \geq 5$. By the choice of v_i, and (A), β_{i-1} crosses either γ_1 or β_1, and all vertices $v_4, v_5, \ldots, v_{i-1}$ are in Q.

It is easy to get a drawing $\mathcal{D}^*$ whose only crossings are: (i) α_j, $j = 4, \ldots, i-1$ with ρ, (ii) β_{i-1} crosses μ; and in $\mathcal{D}^*$ all the other vertices $v_i, \ldots, v_n$ remain in R. This implies $\mathrm{cr}(\mathcal{D}^*) = \mathrm{cr}(\mathcal{D}')$ and satisfies (B).

□

Proof of Theorem 1. Lemma 3 implies that both α_1 and α_3 are not crossing-critical in (G_n, w) for any weight assignment w on G_n. Theorem 1 follows combining this and Remark 1. □

Finally, consider the following weight assignment w'_n on G'_n:

$$w'_n(e) = \begin{cases} 1 & \text{if } e = \alpha_i, i = 4, \ldots, n; \\ n-3 & \text{otherwise.} \end{cases}$$

Lemma 4 $\mathrm{cr}(G'_n, w'_n) = (n-3)^2$ *and* (G'_n, w'_n) *is crossing-critical.*

Proof. The proof for the crossing number is essentially that given in Lemma 3. To prove that the graph is crossing-critical, it is easy to see that for each edge e there exists an optimal drawing of (G'_n, w'_n) where e is involved in a crossing. □

References

[1] L. Beaudou, C. Hernández-Vélez and G. Salazar, Making a graph crossing-critical by multiplying its edges, *Electron. J. of Combin.,* **20(1)** (2013), #P61.

[2] M. DeVos, B. Mohar and R. Šámal, Unexpected behavior of crossing sequences, *J. Combin. Theory Ser. B* **101 (6)** (2011), 448–463.

[3] Z. Dvořák and B. Mohar, Crossing-critical graphs with large maximum degree, *J. Combin. Theory Ser. B* **100 (4)** (2010), 413–417.

[4] B. Mohar, R. J. Nowakowski and D. B. West, Research problems from the 5th Slovenian Conference (Bled, 2003), *Discrete Math.* **307(3-5)** (2007), 650–658.

Witness Bar Visibility

Carmen Cortés*[1], Ferran Hurtado†[2], Alberto Márquez‡[1], and Jesús Valenzuela§[1]

[1]Departamento de Matemática Aplicada I, Universidad de Sevilla.
[2]Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya.

Abstract

Bar visibility graphs were introduced in the seventies as a model for some VLSI layout problems. They have been also studied since then by the graph drawing community, and recently several generalizations and restricted versions have been proposed.

We introduce a generalization, *witness-bar visibility graphs*, and we prove that this class encompasses all the bar-visibility variations considered so far. In addition, we show that many classes of graphs are contained in this family, including in particular all planar graphs, interval graphs, circular arc graphs and permutation graphs.

1 Introduction and preliminary definitions

Given a set S of disjoint horizontal line segments in the plane (called *bars* hereafter) we say that G is a *bar-visibility graph* if there is a bijection between S and the vertices of G, and an edge between two of these if and only if there is a vertical segment (called *line of sight*) between the corresponding bars that does not intersect any other bar. We also say that S is a *bar visibility representation* (or a *bar visibility drawing*) of G.

Bar visibility graphs were introduced by Garey, Johnson and So [13] as a modeling tool for digital circuit design (see also [16]). These representations are also a useful tool for displaying diagrams that convey visual information on relations among data, which is why many variations of these graphs have been considered by the graph drawing community [6, 7, 8, 9, 12, 14, 15].

We need some definitions before we can pose precisely our problem; we use standard terminology as in [5]. We call *v-segment* any vertical segment. We call *ε-segment* any axis aligned rectangle having width $\varepsilon > 0$ (intuitively, a *thick* vertical segment). Let s and t be two horizontal bars. We say that a v-segment connects s and t if its endpoints are in s and t. We say that an ε-segment connects s and t if its horizontal sides are contained in s and t.

Let S be a set of non-overlapping horizontal segments (bars). Two bars $s, t \in S$ are *visible* if, and only if, there is a v-segment connecting s and t intersecting no other segment in S, and we say that s and t are *ε-visible* if, and only if there is an ε–segment connecting s and t intersecting no other segment in S.

With the preceding definition, bar visibility graphs as defined in the first paragraph of this section take as nodes a set of disjoint bars, and there is an edge between two nodes if and only if the corresponding bars are visible (this is also called a *strong visibility representation* of the graph [17]). If instead of visibility we require ε-visibility, then we get bar ε-visibility graphs or, equivalently, an *ε-visibility representation* of the graph. The latter have been characterized as those graphs that admit a planar embedding with all cutpoints on the exterior face [17, 18].

A graph G is a *weak bar visibility graph* if its nodes can be put in bijection with a set of disjoint bars and the nodes corresponding to every edge in G are ε-visible (note that not every ε-visibility need be an edge). This family of graphs is exactly the class of all planar graphs [10].

Finally, we say that G is a *bar k-visibility graph* if there is a bijection between a set of bars S and the vertices of G, and an edge between two of these if and only if there is a v-segment joining the corresponding bars that intersects at most k other bars. This generalization has been introduced in recent years [8, 12].

In this paper we introduce a stronger generalization, *witness-bar visibility graphs*, and we prove that this representation approach encompasses all the bar-visibility variations considered so far. In addition, we show that many classes of graphs are contained in this family, including in particular all planar graphs, interval graphs, circular arc graphs and permutation graphs.

For the definition of witness-bar visibility graphs we consider, in addition to the set S of bars that are in correspondence one-to-one with the vertices of the

*Email: ccortes@us.es.
†Email: Ferran.Hurtado@upc.edu. Research supported by projects MINECO MTM2012–30951, Gen. Catalunya DGR 2009SGR1040, and ESF EUROCORES programme EuroGIGA, CRP ComPoSe: MICINN Project EUI- EURC-2011-4306, for Spain.
‡Email: almar@us.es.
§Email: jesusv@us.es.

graph being constructed, a set of green bars that "favor" visibility, and a set of red bars that "obstruct" visibility. Green bars act as *positive witnesses* while red bars correspond to *negative witnesses*. The bars from S neither favor nor obstruct visibilities.

For the ease of description it is useful to consider also purple bars that obstruct visibility in a slightly different way than red bars.

Definition 1 Let S, S_G, S_P and S_R be four sets of horizontal segments (*bars*, *green-bars*, *purple-bars*, and *red-bars*, respectively) such that any two elements in $S \cup S_G \cup S_R$ are disjoint. We define:

1. The *green-bar visibility graph* of S with respect to S_G has one vertex for every element in S, and two bars $s, t \in S$ are adjacent if and only if there is an ε-segment connecting s and t that *crosses at least one green bar*.
2. The *purple-bar visibility graph* of S with respect to S_P has one vertex for every element in S, and two bars $s, t \in S$ are adjacent if and only if there is an ε-segment connecting s and t that *does not cross any purple bar*.
3. The *witness-bar visibility graph* of S with respect to S_G and S_R has one vertex for every element in S, and two bars $s, t \in S$ are adjacent if and only if there is an ε-segment connecting s and t that *crosses strictly more green bars than red bars*.

The class of green, purple and witness-bar visibility graph are denoted, respectively, by $\mathcal{GBG}$, $\mathcal{PBG}$ and $\mathcal{WBG}$.

An illustration of the three types of graphs is shown in Figure 1 (on a black and white printer, node-bars appear as thin lines, red bars as thick dark lines, purple lines as thick lines colored light grey, and the green lines are seen as thick striped lines).

This work is devoted to the study of the classes of graphs that can be represented via green, purple or bar-visibility graphs and its properties. We start by considering the classes $\mathcal{GBG}$ and $\mathcal{PBG}$, which will be proved to be subclasses of $\mathcal{WBG}$. Then we will enumerate classes of graphs that are contained in $\mathcal{WBG}$, as well as properties of this class related to planarity.

The terminology *witness-bar visibility graphs* is inspired by the concept of *witness proximity graphs*, which focuses on deciding neighborliness relations among points in a finite set according to the presence of some positive and/or negative witness points, a topic that has been studied in recent years [1, 2, 3, 4, 11].

2 The subclasses $\mathcal{GBG}$ and $\mathcal{PBG}$

In this section we study the classes $\mathcal{GBG}$ and $\mathcal{PBG}$ and its relationships with other graph classes. The

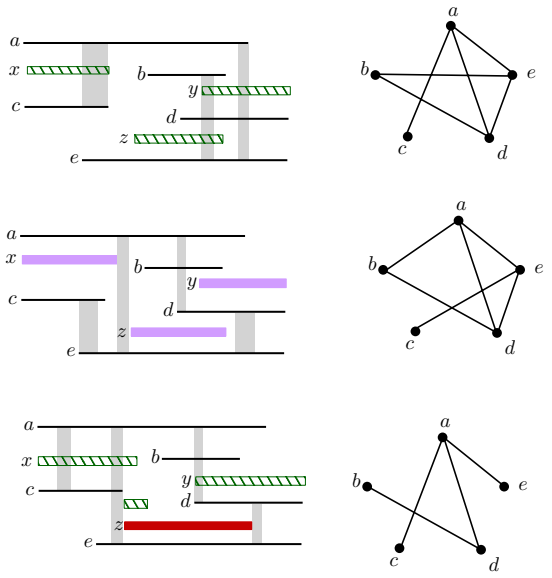


Figure 1: Examples of graphs in the families $\mathcal{GBG}$ (top), $\mathcal{PBG}$ (center) and $\mathcal{WBG}$ (bottom).

interest of this is made clear by our first result:

Lemma 2 *The class of graphs $\mathcal{WBG}$ contains strictly the classes $\mathcal{GBG}$ and $\mathcal{PBG}$.*

An *interval graph* is the intersection graph of a set of (closed) intervals on the real line.

Theorem 3 *Let G be a graph. If G is an interval graph, then $G \in \mathcal{GBG}$. The reverse is in general false.*

The graph class inclusion in Theorem 3 is strict, as one can show that C_4, which is not an interval graph, is in $\mathcal{GBG}$. However, graphs with induced cycles of length greater than 5 are not in $\mathcal{GBG}$.

Proposition 4 *If the girth of a graph in $\mathcal{GBG}$ is finite, then it is at most four.*

As a consequence of the previous result, it follows that the green-bar visibility graph class does not contain any of the bar-visibility classes described in the introduction of this paper, because C_n can be represented as weak/ε/strong bar visibilty graph for every $n \geq 3$ [17].

Note that even although one may think that the classes $\mathcal{GBG}$ and $\mathcal{PBG}$ are related by complementation, possibly by switching purple and green bar coloring, but it is not the case. For example the union of two disjoint triangular cycles is in $\mathcal{GBG}$, as seen in the preceding section, but its complement is $K_{3,3}$, which is not in $\mathcal{PBG}$, a fact that we will see below.

On the positive side, let us see that interval graphs admit a purple-bar visibility representation and prove a useful lemma.

Theorem 5 *If G is an interval graph, then $G \in \mathcal{PBG}$.*

Lemma 6 *Let G be a triangle-free graph. If $G \in \mathcal{PBG}$ then G is a planar graph.*

Proof. The idea is given in Figure 2. □

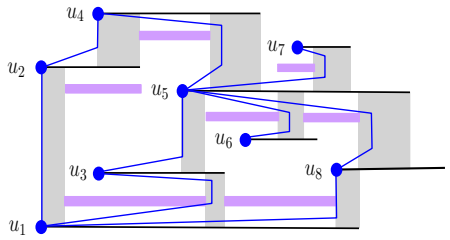


Figure 2: A purple-bar visibility representation of a triangle-free graph and the corresponding construction of its planar embedding.

Nevertheless the previous result cannot be extended to a characterization of the class $\mathcal{PBG}$:

Theorem 7 *1. $K_{3,3} \notin \mathcal{PBG}$.*

2. $K_n \in \mathcal{PBG}$, $\forall n$.

3. To admit a purple-bar visibility representation is not inherited by subgraphs.

Proposition 8 *There are nonplanar graphs with triangles that do not admit a purple-bar visibility representation.*

An example of these graphs is given in Figure 3.

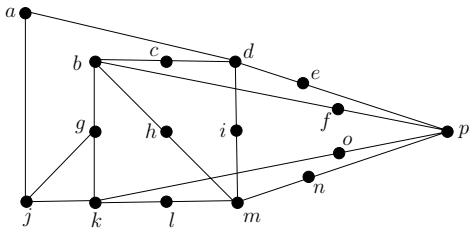


Figure 3: A nonplanar graph G with a triangle ($\triangle gjk$), which does not admit a purple-bar visibility representation.

The class $\mathcal{PBG}$ generalizes the *classical* bar-visibility representations:

Theorem 9 *Every graph G that can be represented as strong/ε/weak bar visibility graph admits as well a purple-bar visibility representation.*

Corollary 10 *Every graph G that can be represented as strong/ε/weak bar visibility graph admits as well a witness-bar visibility representation.*

3 The class $\mathcal{WBG}$ of witness-bar visibility graphs

Witness bar visibility also generalizes k-bar visibility:

Theorem 11 *Every graph G that can be represented as a bar k-visibility graph admits as well a witness-bar visibility representation.*

The idea of the proof is given in Figure 4.

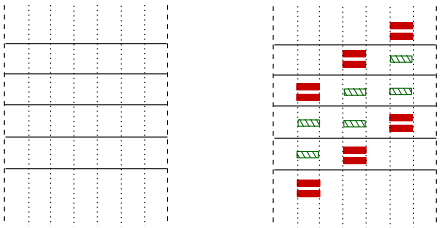


Figure 4: Assume we have to deal with bar 1-visibility, and consider a stack of 5 bars in a strip (left). We subdivide the strip into 7 slabs (right). In the second bar we mimic 1-visibility using $\mathcal{WBG}$-visibility for the 3 lowest bars. In the fourth bar we do the same for the 3 intermediate bars, and in the sixth slab for the 3 highest bars.

Theorem 12 *The circular arc graphs and permutation graphs are contained in $\mathcal{WBG}$*

Given a graph G, let $\widetilde{G}$ be the graph resulting from subdividing once every edge in G.

Lemma 13 *Let G be a graph. If $\widetilde{G} \in \mathcal{WBG}$ then G is a planar graph.*

Lemma 14 *$\widetilde{K}_{3,3} \notin \mathcal{WBG}$ and $\widetilde{K}^c_{3,3} \in \mathcal{WBG}$.*

As a consequence of the preceding lemma we immediately obtain the following result:

Theorem 15 *The class of the graphs that admit a witness-bar visibility representation is not closed under complementation.*

We conclude this section with another result on the class $\mathcal{WBG}$, that discards the possibility of characterizing the class by forbidden minors:

Theorem 16 *The property of admitting a witness-bar visibility representation is not inherited by subgraphs.*

Proof. We know that $K_6 \in \mathcal{WBG}$ from Theorem 7 and Lemma 2. On the other hand $\widetilde{K}_{3,3}$ is a subdivision of a subgraph of K_6, but we know from Lemma 14 that $\widetilde{K}_{3,3}$ is not in $\mathcal{WBG}$. This settles the claim. □

4 Concluding remarks

Let us summarize the properties we have proved for the class $\mathcal{WBG}$ of witness-bar visibility graphs:

- Every graph G that can be represented as strong/ε/weak bar visibility graph admits as well a witness-bar visibility representation.
- Every graph G that can be represented as a bar k-visibility graph admits as well a witness-bar visibility representation.
- The class of interval graphs is contained in the class $\mathcal{WBG}$.
- If G is a circular arc graph or a permutation graph then $G \in \mathcal{WBG}$.
- The class of the graphs that admit a witness-bar visibility representation is not closed under complementation.
- The property of admitting a witness-bar visibility representation is not inherited by subgraphs, which discards the possibility of characterizing the graph class $\mathcal{WBG}$ by forbidden minors.

We conclude that the graph class $\mathcal{WBG}$ is very rich and encompasses many other classes. However, to obtain a characterization or a recognition algorithm appear to be quite challenging problems.

References

[1] O. Aichholzer, R. Fabila, T. Hackl, A. Pilz, P. Ramos, M. van Kreveld, and B. Vogtenhuber. Blocking Delaunay triangulations. To appear in *Comput. Geom.* (accepted 2012). Online version available at http://dx.doi.org/10.1016/j.comgeo.2012.02.005.

[2] B. Aronov, M. Dulieu and F. Hurtado, Witness (Delaunay) graphs. *Comput. Geom.* 44(6-7):329-344, 2011.

[3] B. Aronov, M. Dulieu and F. Hurtado, Witness Gabriel graphs. To appear in *Comput. Geom.* (accepted 2011). Online version at DOI: 10.1016/j.comgeo.2011.06.004.

[4] B. Aronov, M. Dulieu, and F. Hurtado, Witness rectangle graphs. In Algorithms and Data Structures Symposium (WADS), volume 6844 of Lectures Notes in Computer Science, pages 73-85. Springer, 2011.

[5] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. Graph Drawing. Prentice Hall Inc., Upper Saddle River, NJ, 1999.

[6] P. Bose, A. M. Dean, J. P. Hutchinson, and T. C. Shermer. On rectangle visibility graphs. In S. C. North, editor, Graph Drawing 1996, volume 1190 of Lecture Notes in Comput. Sci., pages 25-44, Berlin, 1997. Springer- Verlag.

[7] G. Chen, J. P. Hutchinson, K. Keating, and J. Shen. Characterizations of [1,k]-bar visibility trees. Electr. J. Comb. 13(1), 2006.

[8] A. M. Dean, W. Evans, E. Gethner, J. D. Laison, M. A. Safari, and W. T. Trotter. Bar k-visibility graphs. J. Graph Algorithms & Applications, 11(1):45-59, 2007.

[9] A. M. Dean, E. Gethner, and J. P. Hutchinson. Unit bar-visibility layouts of triangulated polygons: Extended abstract. In J. Pach, editor, Graph Drawing 2004, volume 3383 of Lecture Notes in Comput. Sci., pages 111-121, Berlin, 2005. Springer-Verlag.

[10] P. Duchet, Y. Hamidoune, M. L. Vergnas, and H. Meyniel. Representing a planar graph by vertical lines joining different levels. Discrete Math., 46:319-321, 1983.

[11] M. Dulieu, Witness proximity graphs and other geometric problems, Ph.D. thesis, Polytechnic Institute of New York University, April 2012.

[12] S. Felsner and M. Massow, Parameters of Bar k-Visibility Graphs. J. Graph Algorithms and Appl. vol. 12, no. 1, pp. 5-27 (2008).

[13] M. R. Garey, D. S. Johnson, and H. C. So. An application of graph coloring to printed circuit testing. IEEE Trans. Circuits and Systems, CAS-23(10):591-599, 1976.

[14] J. P. Hutchinson. Arc- and circle-visibility graphs. Australas. J. Combin., 25:241-262, 2002.

[15] J. P. Hutchinson, T. Shermer, and A. Vince. On representations of some thicknesstwo graphs. Comput. Geom., 13:161-171, 1999.

[16] M. Schlag, F. Luccio, P. Maestrini, D. Lee, and C. Wong. A visibility problem in VLSI layout compaction. In F. Preparata, editor, Advances in Computing Research, volume 2, pages 259-282. JAI Press Inc., Greenwich, CT, 1985.

[17] R. Tamassia and I. G. Tollis. A unified approach to visibility representations of planar graphs. Discrete Comput. Geom., 1(4):321-341, 1986.

[18] S. K. Wismath. Characterizing bar line-of-sight graphs. In Proceedings of the First Symposium of Computational Geometry, pages 147-152. ACM, 1985.

The alternating path problem revisited

Mercè Claverol[1], Delia Garijo[2], Ferran Hurtado[1], Dolores Lara[3], and Carlos Seara[1]

[1,3]Universitat Politècnica de Catalunya, Barcelona, Spain
[2]Universidad de Sevilla, Spain

Abstract

It is well known that, given n red points and n blue points on a circle, it is not always possible to find a plane geometric Hamiltonian alternating path. In this work we prove that if we relax the constraint on the path from being plane to being 1-plane, then the problem always has a solution, and even a Hamiltonian alternating cycle can be obtained on all instances. We also extend this kind of result to other configurations and provide remarks on similar problems.

Introduction

A *geometric graph* is a graph drawn in the plane whose vertex set is a set of points and whose edges are straight-line segments connecting pairs of vertices. Two edges of a geometric graph *cross* if they have an intersection point lying in the relative interior of both edges. A *plane geometric graph* is a geometric graph without any edge crossings. A *1-plane geometric graph* is a geometric graph in which every edge has at most one crossing. Notice that the terms plane graph and 1-plane graph refer to a geometric object, while to be planar or 1-planar are properties of the underlying abstract graph. We use here standard notation for geometric graphs as in [3] and [12].

Let P be a set of points in the plane in general position (i.e., no three points are collinear), and let $CH(P)$ denote its convex hull. A *geometric spanning tree* on P, generically denoted by tree(P), is any spanning tree on P whose edges are straight-line segments connecting two points on P. Observe that if $|P| = 1$, the tree on P is a single vertex. When the tree is a path, we write path(P). The *geometric complete graph* $K(P)$ on P is the complete geometric graph with vertex set P. Notice that tree(P) is a spanning tree of $K(P)$.

[1]Email: merce@ma4.upc.edu, ferran.hurtado@upc.edu, carlos.seara@upc.edu. Partially supported by projects MINECO MTM2012-30951, Gen. Cat. DGR2009SGR1040, and ESF EUROCORES programme EuroGIGA, CRP ComPoSe: MICINN Project EUI-EURC-2011-4306.

[2]Email: dgarijo@us.es. Partially supported by projects 2009/FQM-164 and 2010/FQM-164.

[3]Email: maria.dolores.lara@upc.edu.

Let R and B be two disjoint sets of red and blue points in the plane such that no three points of $R \cup B$ lie on the same line. The *geometric complete bipartite graph* $K(R, B)$ is the graph with vertex set $R \cup B$ and whose edges are all the straight-line segments connecting any point in R and any point in B. A line segment defined by two red points is a *red segment*, and one defined by two blue points is a *blue segment*. More generally, an edge is said to be *monochromatic* when the two endpoints have the same color, and *bichromatic* otherwise. The intersections between red segments and blue segments are called *bichromatic crossings*, and those between segments having the same color are called *monochromatic crossings*.

Problems on adding edges to a given graph to obtain a new graph with some desirable properties are, in general, called *augmentation problems*. Among these, *plane augmentation* considers an initial plane graph $G = (V, E)$ (possibly empty, i.e., only the point set V is given) that has to be augmented to another plane supergraph $G' = (V, E \cup E')$ by adding a set E' of edges to G, see the survey [7].

In this work we focus on problems in which the initial point set is a bicolored set $R \cup B$. This family of problems has attracted a substantial amount of research, see for instance the surveys [8, 7].

We first consider *alternating graphs*, i.e., those in which every edge is bichromatic. A well known fact [2, 1, 10, 11] is that given n red points and n blue points on a circle (equivalently, in convex position), one cannot always obtain a plane geometric Hamiltonian alternating path. In this paper we prove that, if we relax the constrain on the geometric Hamiltonian alternating path from being plane to being 1-plane, then a solution always exists, even yielding stronger properties. We also show that the same result holds for some other configurations. These results appear in Section 1.

Regarding *monochromatic graphs*, i.e, graphs in which every edge is monochromatic, it is easy to see that one cannot always construct a plane perfect matching in $K(R) \cup K(B)$. A trivial example is given by the vertices of a convex quadrilateral in which two opposite vertices are colored red and the other two are colored blue. The same example shows that it is not always possible to obtain two geometric monochro-

matic spanning trees, tree(R) and tree(B), such that their union is plane.

The proven nonexistence of plane configurations had already suggested to researchers to allow some relaxation in the constraint, but the focus was put on constructing geometric graphs having globally few crossings [9, 13]. However, one of the constructions in [13] is in fact a 1-plane graph. We include some remarks on that particular result and its consequences in Section 2.

We conclude in Section 3 with some additional comments.

1 Alternating graphs: paths and cycles

In this section we study geometric alternating spanning graphs on $R \cup B$, i.e., spanning subgraphs of $K(R, B)$. We focus on geometric Hamiltonian alternating paths and cycles, which visit red points and blue points alternately. Notice that when there are no crossings at all, geometric Hamiltonian paths and cycles are also called *simple polygonals* and *simple polygons*, respectively.

1.1 Convex position

The problem of restricting the bicolored point set to lay on a circle has attracted a lot of attention. It is easy to see that there are sets R and B with the same number of points, say n, such that $R \cup B$ is in convex position, and a plane geometric Hamiltonian alternating path on $R \cup B$ cannot exist. Erdős (see [10]) proposed in 1989 to study the value $\ell(n)$ such that no matter how the colors are distributed a plane geometric alternating path of length at least $\ell(n)$ always exists. About the same time, Akiyama and Urrutia [2] considered independently the same problem: they proved a necessary and sufficient condition for the existence of a plane geometric Hamiltonian alternating path and derived an $O(n^2)$ time algorithm to find one, if it exists. Abellanas et al. [1], and independently Kynčl et al. [10], proved that $\ell(n) \leq \frac{4}{3}n + O(\sqrt{n})$, and Cibulka et al. [4] showed that $\ell(n) \geq n + \Omega(\sqrt{n})$. These bounds are the best to date. Remind that the total number of points is $2n$. Also, notice that we slightly abuse the notation by using inequalities, and that the two bounds have to be read together, not independently. For more information and details, see Mészáros' PhD Thesis [11].

We next show that if $R \cup B$ is in convex position then a 1-plane geometric Hamiltonian alternating cycle –not only a path– can always be drawn. The following lemma is the key tool.

Lemma 1 *Let R and B be two disjoint sets of red and blue points in the plane such that $R \cup B$ is in convex position, and $|R| = |B| = n \geq 2$. Let S be a set of disjoint bichromatic segments on the boundary of the convex hull of $R \cup B$, and $|S| = s \geq 2$. Then, there exists a 1-plane geometric Hamiltonian alternating cycle on $R \cup B$ that contains each segment of S as an edge.*

If $R \cup B$ is in convex position and $|R| = |B| = n \geq 2$ then there are at least two disjoint bichromatic segments on the boundary of $CH(R \cup B)$, say s_1, s_2. Let $S = \{s_1, s_2\}$. By Lemma 1, one can draw a 1-plane geometric Hamiltonian alternating cycle on $R \cup B$ that contains each segment of S as an edge. This argument proves our main result in this section.

Theorem 2 *Let R and B be two disjoint sets of red and blue points in the plane such that $R \cup B$ is in convex position, and $|R| = |B| = n \geq 2$. Then, there exists a 1-plane geometric Hamiltonian alternating cycle on $R \cup B$.*

1.2 Double chain

We next consider the problem of drawing a 1-plane geometric Hamiltonian alternating cycle on a double chain whose points are colored red and blue.

The double chain (formally defined below) is a configuration that has been intensively studied since it admits many triangulations, many polygonizations, many crossing-free matchings, etc., and for several families of graphs yields the maximum number known to date of such possible configurations among point sets with the same cardinality (see http://www.cs.tau.ac.il/ sheffera/counting/PlaneGraphs.html).

A *double chain* (C_1, C_2) consists of two opposite convex chains C_1 and C_2, facing each other, such that the convex hull of $C_1 \cup C_2$ is a quadrilateral, each point of C_2 lies strictly below every line determined by two points of C_1, and each point of C_1 lies strictly above every line determined by two points of C_2. When the points of (C_1, C_2) are colored red and blue, (C_1, C_2) is said to be a *bicolored double chain* (see Figure 1), and r_i, b_i denote the number of red and blue points, respectively, of C_i for $i = 1, 2$. Note that the sizes of C_1 and C_2 may be different.

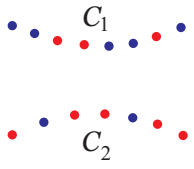


Figure 1: A bicolored double chain (C_1, C_2).

Cibulka et al. [4] proved the following theorem:

Theorem 3 [4] *(i) If $|C_i| \geq \frac{1}{5}(|C_1| + |C_2|)$ for $i = 1, 2$, then there exists a non-crossing geometric Hamiltonian alternating path on (C_1, C_2); (ii) there exist bicolored double chains in which one of the chains contains at most 1/29 of all the points, which do not admit a non-crossing geometric Hamiltonian alternating path.*

Theorem 4 below states that when $r_1+r_2 = b_1+b_2$, allowing at most one crossing per edge is strong enough to let us always draw a geometric Hamiltonian alternating cycle on (C_1, C_2). The main idea to obtain this cycle is to use Theorem 2 to construct two 1-plane geometric alternating cycles Λ_1 and Λ_2 in $CH(C_1)$ and $CH(C_2)$, respectively, and to connect them drawing another 1-plane geometric alternating cycle Λ in the exterior of $CH(C_1)$ and $CH(C_2)$. This process is described next.

Let E_i, $i = 1, 2$, be the set of edges in $CH(C_i)$ that connect consecutive points of C_i. Suppose first that $3 \leq b_1 + 1 < r_1$ and $3 \leq r_2 + 1 < b_2$. Then there exist at least two monochromatic edges in $E_1 \cup E_2$: a red edge $rr' \in E_1$ and a blue edge $bb' \in E_2$. Contract them obtaining a red point r'' and a blue point b''. By Theorem 2, we can draw a cycle Λ_1 on the point set formed by the b_1 blue points of C_1, the red point r'', and $b_1 - 1$ red points of $C_1 \setminus \{r, r'\}$. Analogously, Λ_2 is constructed on the point set formed by the r_2 red points of C_2, the blue point b'', and $r_2 - 1$ blue points of $C_2 \setminus \{b, b'\}$. Finally, the cycle Λ is drawn, as in Figure 2, on the remaining $r_1 - b_1 - 1$ red points of C_1, the remaining $b_2 - r_2 - 1 = r_1 - b_1 - 1$ blue points of C_2, and the points r'' and b''. Observe that r'' and b'' connect Λ with Λ_1 and Λ_2, respectively. By reversing the contraction and deleting the edges rr' and bb', we "open" the three cycles obtaining the desired cycle on (C_1, C_2).

Note that the preceding argument can easily be adapted for b_1 or r_2 equal to zero or one. Observe also that if all the edges in E_1 (analogous for E_2) are bichromatic then either $r_1 = b_1$ (and so $r_2 = b_2$) or $r_1 = b_1 + 1$ (and $b_2 = r_2 + 1$). For these values, even if there are monochromatic edges in E_1, we need to use slightly different arguments which are omitted for the sake of brevity.

Theorem 4 *Let R and B be the sets of red and blue points of a bicolored double chain (C_1, C_2), and $|R| = |B| \geq 2$. Then, there exists a 1-plane geometric Hamiltonian alternating cycle on (C_1, C_2).*

1.3 General position

The positive results for convex position and for the double chain make one wonder whether a similar result holds for any set of points in general position:

Question 1 *Let R and B be any two disjoint sets of red and blue points in the plane such that no three points of $R \cup B$ lie on the same line, and $|B| = |R| \geq 2$. Does there always exist a 1-plane geometric Hamiltonian alternating cycle on $R \cup B$?*

We do not know the answer to the preceding question. Geometric Hamiltonian cycles with few crossings were obtained by Kaneko et al. [9], who gave a tight upper bound of $|R| - 1$ for the number of crossings of a geometric Hamiltonian alternating cycle. Figure 2 illustrates a configuration $R \cup B$ for which this upper bound is best possible.

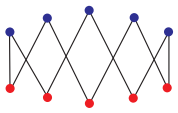


Figure 2: A 1-plane geometric Hamiltonian alternating cycle, on a point set $R \cup B$, with $|R| - 1$ crossings.

To be precise, Kaneko et al. [9] proved the following result.

Theorem 5 [9] *Let R and B be two disjoint sets of points in the plane such that $|R| = |B|$ and no three points of $R \cup B$ are on the same line. Then we can draw a geometric Hamiltonian alternating cycle on $R \cup B$ which has at most $|R| - 1$ crossings. Moreover there exist configurations $R \cup B$ for which this upper bound $|R| - 1$ is the best possible.*

To prove this theorem, the authors use several lemmas that we have carefully examined to see whether it is possible to adapt their proof to obtain a 1-plane graph. However, as far as we can see, there are cases in which the connection of the paths that exist by induction is not necessarily 1-plane.

It is unclear to us which is the right answer to Question 1, yet our study leads us to believe that it is negative in general, yet positive if only a Hamiltonian path is required, which we state as a conjecture:

Conjecture 1 *Let R and B be two disjoint sets of red and blue points in the plane such that no three points of $R \cup B$ lie on the same line, and $|B| \leq |R| \leq |B| + 1$. There always exists a 1-plane geometric Hamiltonian alternating path on $R \cup B$.*

Let us mention that this is ongoing research, currently focusing on the preceding conjecture, which we hope to answer in the next version of this paper.

2 A remark on monochromatic graphs

In this section we would like to remark that a result obtained by Tokunaga in 1996 can be rephrased in terms of 1-plane graphs, hence giving support to the idea that exploring this relaxation may be worth the effort for more problems.

On one hand, it is not always possible to construct a non-crossing perfect matching in $K(R) \cup K(B)$ as proved by Dumitrescu and Steiger [6]. The original result was improved by Dumitrescu and Kaye [5], who proved that for given R and B, with $|R| + |B| = n$, there always exists a non-crossing matching in $K(R) \cup K(B)$ which covers at least $0.8571 \cdot n$ points of $R \cup B$, while for some configurations every non-crossing matching in $K(R) \cup K(B)$ covers at most $0.9871 \cdot n$ points of $R \cup B$. On the other hand, drawing plane red and blue geometric spanning trees on R and B that avoid bichromatic crossings is not always possible, and Tokunaga [13] characterized their existence in terms of the bichromatic edges on $CH(R \cup B)$.

One may wonder whether using 1-plane graphs would always yield a positive solution to the above problems. The answer is affirmative: Tokunaga [13] also proved that for given R and B, there exists a pair $(\text{path}(R), \text{path}(B))$ of red and blue geometric simple Hamiltonian paths such that each edge of $\text{path}(R)$ intersects at most one edge of $\text{path}(B)$ and vice versa. Having the red and blue geometric simple Hamiltonian paths from that result, with at most one bichromatic crossing per edge, we already have got 1-plane spanning trees, and taking in each path one segment out of any two consecutive we get a 1-plane perfect matching with no monochromatic crossings. This 1-plane matching can also be obtained by using the Ham-sandwich theorem and induction on $|R \cup B|$.

We include this remark because the focus in [13] was to get few crossings rather than achieving the 1-plane character, but the consequences show that pursuing the latter line of research may be of interest.

3 Conclusion

We have proved that several problems on bicolored point sets asking for the construction of a plane geometric graph with some requisites, and that have in general negative answer, turn out to have a solution if the requirement of the graphs being plane is relaxed to being 1-plane.

As mentioned in a previous section, this is ongoing research and answering Conjecture 1 is our priority. On the other hand, we are also studying the same relaxation for other problems in which 1-plane graphs may provide a solution where plane graphs are not sufficient.

References

[1] M. Abellanas, A. García, F. Hurtado, and J. Tejel, Caminos alternantes, in: *Actas X Encuentros de Geometría Computacional* (in Spanish), 2003, 7–12.

[2] J. Akiyama and J. Urrutia, Simple alternating path problem, *Discr. Math.* **84** (1990), 101–103.

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

[4] J. Cibulka, J. Kynčl, V. Mészáros, R. Stolař, and P. Valtr, Hamiltonian alternating paths on bicolored double-chains, in: *Graph Drawing 2008, Lecture Notes in Computer Science* **5417** (2009), 181–192.

[5] A. Dumitrescu and R. Kaye, Matching colored points in the plane: Some new results, *Comput. Geom. Theory Appl.* **19** (2001), 69–85.

[6] A. Dumitrescu and W. Steiger, On a matching problem in the plane, *Discr. Math.* **211** (2000), 183–195.

[7] F. Hurtado and C. D. Tóth, Plane geometric graph augmentation: a generic perspective, Chapter 16 in: *Thirty Essays on Geometric Graph Theory* (J. Pach, ed.), vol. 29 of Algorithms and Combinatorics, Springer, 2013, 327–354.

[8] A. Kaneko and M. Kano, Discrete geometry on red and blue points in the plane - a survey, in: *Discrete and Computational Geometry, The Goodman-Pollack Festschrift*; edited by B. Aronov et al., Springer, 2003, 551–570.

[9] A. Kaneko, M. Kano, and Y. Yoshimoto, Alternating Hamiltonian cycles with minimum number of crossings in the plane, *Int. J. Comput. Geom. Appl.* **10** (2000), 73–78.

[10] J. Kynčl, J. Pach, and G. Tóth, Long alternating paths in bicolored point sets, in: *Graph Drawing 2004* (J. Pach, ed.), *Lecture Notes in Computer Science* **3383** (2004), 340–348. Also in *Discr. Math.* **308** (2008), 4315–4322.

[11] V. Mészáros. Extremal problems on planar point sets PhD Thesis, University of Szeged, 2011.

[12] J. Pach, ed. *Thirty Essays on Geometric Graph Theory*, vol. 29 of Algorithms and Combinatorics, Springer, 2013.

[13] S. Tokunaga, Intersection number of two connected geometric graphs, *Information Proc. Discrete Math.* **150** (1996), 371–378.

Phase transitions in the Ramsey-Turán theory

József Balogh*[1]

[1]Department of Mathematics, University of Illinois, Urbana, IL 61801, USA

Abstract

Let $f(n)$ be a function and L be a graph. Denote by $\mathbf{RT}(n, L, f(n))$ the maximum number of edges of an L-free graph on n vertices with independence number less than $f(n)$. Erdős and Sós [7] asked if $\mathbf{RT}\left(n, K_5, c\sqrt{n}\right) = o\left(n^2\right)$ for some constant c. We answer this question by proving the stronger $\mathbf{RT}\left(n, K_5, o\left(\sqrt{n\log n}\right)\right) = o\left(n^2\right)$. It is known that $\mathbf{RT}\left(n, K_5, c\sqrt{n\log n}\right) = n^2/4 + o\left(n^2\right)$ for $c > 1$, so one can say that K_5 has a Ramsey-Turán-*phase transition* at $c\sqrt{n\log n}$. We extend this result to several other K_p's and functions $f(n)$, determining many more phase transitions. We shall formulate several open problems, in particular, whether variants of the Bollobás-Erdős graph, which is a geometric construction, exist to give good lower bounds on $\mathbf{RT}\left(n, K_p, f(n)\right)$ for various pairs of p and $f(n)$. These problems are studied in depth by Balogh-Hu-Simonovits [1], where among others, the Szemerédi's Regularity Lemma and the Hypergraph Dependent Random Choice Lemma are used.

Notation

Definition 1 Denote by $\mathbf{RT}(n, L, m)$ the maximum number of edges of an L-free graph on n vertices with independence number less than m.

We are interested in the asymptotic behavior of $\mathbf{RT}(n, L, f(n))$ and its "phase transitions", i.e., in the question, when and how the asymptotic behavior of $\mathbf{RT}(n, L, f)$ changes sharply when we replace f by a slightly smaller g.

Definition 2 Let

$$\overline{\rho\tau}(L, f) = \limsup_{n\to\infty} \frac{\mathbf{RT}(n, L, f(n))}{n^2}$$

and

$$\underline{\rho\tau}(L, f) = \liminf_{n\to\infty} \frac{\mathbf{RT}(n, L, f(n))}{n^2}.$$

If $\overline{\rho\tau}(L, f) = \underline{\rho\tau}(L, f)$, then we write $\mathbf{RT}(L, f) = \overline{\rho\tau}(L, f) = \underline{\rho\tau}(L, f)$, and call $\mathbf{RT}$ the *Ramsey-Turán density* of L with respect to f, $\overline{\rho\tau}$ the upper, $\underline{\rho\tau}$ the lower *Ramsey-Turán densities*, respectively.

*Research is partially supported by NSF CAREER Grant DMS-0745185 and Arnold O. Beckman Research Award (UIUC Campus Research Board 13039). jobal@math.uiuc.edu

1 Introduction

Szemerédi [10], using his regularity lemma [11], proved $\overline{\rho\tau}(K_4, o(n)) \le 1/8$. Bollobás and Erdős [4] constructed the so-called Bollobás-Erdős graph, one of the most important constructions in this area, which shows that $\underline{\rho\tau}(K_4, o(n)) \ge 1/8$. Indeed, the Bollobás-Erdős graph on n vertices is K_4-free, with $(\frac{1}{8} + o(1))n^2$ edges and independence number $o(n)$. Later, Erdős, Hajnal, Sós and Szemerédi [5] extended these results, determining $\mathbf{RT}(K_{2r}, o(n))$.

Our focus here is somewhat different; we are interested exploring the situation when the independence number is really small.

In [7], Erdős and Sós proved that $\mathbf{RT}(n, K_5, c\sqrt{n}) \le \frac{1}{8}n^2 + o\left(n^2\right)$ for every $c > 0$. They also asked if $\mathbf{RT}(n, K_5, c\sqrt{n}) = o\left(n^2\right)$ for some $c > 0$. They also suggested the following construction, which with the current state of the art yields $\mathbf{RT}\left(n, K_5, c\sqrt{n\log n}\right) = n^2/4 + o\left(n^2\right)$ for $c > 1$: Into each of the parts of a complete balanced bipartite graph place a triangle-free graph with as small independence number as possible. Then we obtain a K_5-free graph with $(1/4 + o(1))n^2$ edges, with low independence number. One of our main results answers the question of Erdős and Sós; practically saying that this construction is optimal.

Theorem 3

$$\mathbf{RT}\left(n, K_5, o\left(\sqrt{n\log n}\right)\right) = o(n^2).$$

We finish this section with an observation, a result and an open question on K_6.

For any $c > 1$,

$$\underline{\rho\tau}\left(K_6, c\sqrt{n\log n}\right) \ge \underline{\rho\tau}\left(K_5, c\sqrt{n\log n}\right) \ge 1/4. \tag{1}$$

Sudakov [9] using the dependent random choice proved that $\sqrt{n}$ is the proper range for the phase-transition for K_6.

Theorem 4 $\mathbf{RT}\left(K_6, \sqrt{n}e^{-c\sqrt{\log n}}\right) = o(n^2)$.

Problem 1 *Is* $\underline{\rho\tau}\left(K_6, 0.1\sqrt{n\log n}\right) = 0$*?*

Balogh and Lenz [2, 3] using some variants of the Bollobás-Erdős graph solved some longstanding open problems in this field. Therefore there is a chance that using discrete geometry, someone can solve Problem 1. Here we describe the Bollobás-Erdős graph, and provide the argument for the K_5 case. These problems are studied in depth by Balogh-Hu-Simonovits [1], where among others, the Szemerédi's Regularity Lemma and the Hypergraph Dependent Random Choice Lemma are used.

2 Proof of Theorem 3

The next lemma is taken from the survey of Fox and Sudakov [8].

Lemma 5 *(Dependent Random Choice Lemma) Let a, d, m, n, r be positive integers. Let $G = (V, E)$ be a graph with n vertices and average degree $d = 2e(G)/n$. If there is a positive integer t such that*

$$\frac{d^t}{n^{t-1}} - \binom{n}{r}\left(\frac{m}{n}\right)^t \geq a, \tag{2}$$

then G contains a subset U of at least a vertices such that every r vertices in U have at least m common neighbors.

We actually prove the following quantitive version of the theorem.

Theorem 6 *For every $k > 3$, if $\omega(n) \to \infty$ then*

$$\mathbf{RT}\left(n, K_5, \frac{\sqrt{n\log n}}{\omega(n)}\right) \leq \frac{n^2}{(\omega(n))^{1/k}} = o\left(n^2\right). \tag{3}$$

Proof. In Theorem 6, we have fixed a $k > 3$ and an $\omega \to \infty$. If n is large enough, then $\varepsilon \geq \omega(n)^{-1/k}$. Assume that there is a K_5-free graph G_n with

$$e(G_n) \geq \varepsilon n^2 \text{ and } \alpha(G_n) < \frac{\sqrt{n\log n}}{\omega(n)}. \tag{4}$$

We apply Lemma 5 with $n, a = \frac{4n}{\omega(n)^2}$, $r = 3$, $d = 2\varepsilon n$, $m = \sqrt{n\log n}$, and $t = k + 3$.
Now the condition of Lemma 5, (2) is satisfied as

$$\frac{d^t}{n^{t-1}} - \binom{n}{r}\left(\frac{m}{n}\right)^t \geq (2\varepsilon)^t n - n^3\left(\frac{\log n}{n}\right)^{t/2} > \varepsilon^t n$$

$$\geq \frac{n}{\omega(n)^{1+3/k}} > a.$$

So there exists a vertex subset U of G with $|U| = a = 4n/\omega(n)^2$ such that all subsets of U of size 3 have at least m common neighbors. Either U has an independent set of size at least $\left(\frac{1}{\sqrt{2}} - o(1)\right)\sqrt{\frac{4n}{\omega(n)^2}\log\left(\frac{4n}{\omega(n)^2}\right)} > \alpha(G_n)$, or $G_n[U]$ contains a triangle. In the latter case, denote by W the common neighborhood of the vertices of the triangle. It follows that $|W| \geq m = \sqrt{n\log n} > \alpha(G_n)$, so $G_n[W]$ contains an edge, and this edge forms a K_5 with the triangle. □

3 The Bollobás-Erdős Graph

The Bollobás-Erdős Graph [4] is a surprising construction, which we describe in this section. First we list some properties of the high dimensional sphere $\mathbb{S}^k$. Denote $\mu()$ the 'normalized' measure on $\mathbb{S}^k$, i.e., $\mu(\mathbb{S}^k) = 1$. The following properties of the high dimensional sphere is crucial: Given any $\alpha, \beta > 0$, it is possible to select $\epsilon > 0$ small enough and then k large enough so that Properties (P1), (P2), and (P3) below are satisfied.

(P1) Let C be a spherical cap in $\mathbb{S}^k$ with height h, where $2h = \left(\sqrt{2} - \epsilon/\sqrt{k}\right)^2$ (this means that all points of the spherical cap are within distance $\sqrt{2} - \epsilon/\sqrt{k}$ of the center). Then $\mu(C) \geq \frac{1}{2} - \alpha$.

(P2) Let $C_1, \ldots, C_t$ be spherical caps in $\mathbb{S}^k$ with height h, where $2h = \left(\sqrt{2} - \epsilon/\sqrt{k}\right)^2$. Let z_i be the center of C_i. Assume for all $1 \leq i < j \leq t$ that $d(z_i, z_j) \leq \sqrt{2}$. Then $\mu(C_1 \cap \ldots \cap C_t) \geq \frac{1}{2^t} - t\alpha$.

(P3) Let C be a spherical cap with diameter $2 - \epsilon/(2\sqrt{k})$. Then $\mu(C) \leq \beta$.

We also use the following properties of high dimensional spheres.

(P4) For any $0 < \gamma < \frac{1}{4}$, it is impossible to have $p_1, p_2, q_1, q_2 \in \mathbb{S}^k$ such that $d(p_1, p_2) \geq 2 - \gamma$, $d(q_1, q_2) \geq 2 - \gamma$, and $d(p_i, q_j) \leq \sqrt{2} - \gamma$ for all $1 \leq i, j \leq 2$.

(P5) Let $A \subseteq \mathbb{S}^k$ and let C be a spherical cap of the same measure. Then $\mathrm{diam}(A) \geq \mathrm{diam}(C)$.

(P6) Let $A, B \subseteq \mathbb{S}^k$ with equal measure and let C be a cap of the same measure. Then $d_{\max}(A, B) \geq \mathrm{diam}(C)$.

Properties (P1) and (P2) follow directly from the formula for the measure of a spherical cap, Properties (P3), (P5), and (P6) are all folklore results that are easy corollaries of the isoperimetric inequality on the sphere, and Property (P4) can be proved by examining distances and using the triangle inequality.

In order to prove that $\mathbf{RT}(n, K_4, o(n)) \geq \frac{n^2}{8}$, we need to construct, for every $\alpha, \beta > 0$, a K_4-free graph G with n vertices, independence number at most βn, and at least $\frac{n^2}{8}(1-\alpha)$ edges. Given $\alpha, \beta \geq 0$, take ϵ small enough and k large enough so that Properties (P1) and (P3) hold. Divide the k-dimensional unit sphere $\mathbb{S}^k$ into $n/2$ domains having equal measure and diameter at most $\frac{\epsilon}{10\sqrt{k}}$. Choose a point from each domain and let P be the set of these points. Let $\phi : P \to \mathcal{P}(\mathbb{S}^k)$ map points of P to the corresponding domain of the sphere. Take as vertex set of G the disjoint union of two sets V_1 and V_2 both isomorphic to P. For $x, y \in V_i$ we make xy an edge of G if $d(x,y) \geq 2-\epsilon/\sqrt{k}$. For $x \in V_1, y \in V_2$ we make xy an edge of G if $d(x,y) \leq \sqrt{2}-\epsilon/\sqrt{k}$. Then Property (P1) shows that every vertex in V_1 has at least $\frac{1}{2}|V_2|(1-\alpha)$ neighbors in V_2 so the total number of edges is at least $\frac{1}{8}n^2(1-\alpha)$. If I is a set in V_1 with $|I| \geq \beta |V_1| = \beta\frac{n}{2}$, then $\mu(\phi(I)) = |I|/|P| \geq \beta$. Let C be a spherical cap of measure $\mu(\phi(I))$. Properties (P3) and (P5) show that $2-\epsilon/(2\sqrt{k}) \leq \operatorname{diam}(C) \leq \operatorname{diam}(\phi(I))$. For $p \in I$, each $\phi(p)$ has diameter at most $\epsilon/(10\sqrt{k})$ so we can find two points $p_1, p_2 \in I$ with $d(p_1, p_2) \geq 2-\epsilon/\sqrt{k}$, showing that I is not independent. Finally, Property (P4) shows this graph has no K_4 as a subgraph since any K_4 must take two vertices from V_1 and two vertices from V_2 (the graph spanned by V_i is triangle-free). To summarize, we have constructed a K_4-free graph G on n vertices with independence number at most βn and at least $\frac{1}{8}n^2(1-\alpha)$ edges. Since this construction holds for any $\alpha, \beta > 0$, we have proved that $\overline{\rho\tau}(K_4, o(n)) \leq 1/8$.

4 Conclusion

The aim of this note was to raise awareness for a problem in extremal graph theory, where a solution could come by geometric tools.

Acknowledgement: We thank for the anonymous referee pointing out several typoes of the manuscript.

References

[1] J. Balogh, P. Hu, and M. Simonovits, Phase transitions in the Ramsey-Turán theory, manuscript.

[2] J. Balogh and J. Lenz, On the Ramsey-Turán numbers of graphs and hypergraphs, *Israel J. Math.* **194** (2013) 45–68.

[3] J. Balogh and J. Lenz, Some exact Ramsey-Turán numbers, *Bull. London Math. Soc.*

[4] B. Bollobás and P. Erdős, On a Ramsey-Turán type problem., *J. Combinatorial Theory Ser. B*, **21**(2):166–168, 1976.

[5] P. Erdős, A. Hajnal, V. T. Sós, and E. Szemerédi, More results on Ramsey-Turán type problems, *Combinatorica*, **3**(1):69–81, 1983.

[6] P. Erdős, A. Hajnal, M. Simonovits, V. T. Sós, and E. Szemerédi, Turán- Ramsey theorems and simple asymptotically extremal structures. *Combinatorica*, **13** (1):31–56, 1993.

[7] P. Erdős and V. T. Sós, Some remarks on Ramsey's and Turán's theorem. In Combinatorial theory and its applications, II (Proc. Colloq., Balatonfüred, 1969), pages 395–404. North-Holland, Amsterdam, 1970.

[8] J. Fox and B. Sudakov. Dependent random choice. Random Structures & Algorithms, **38** (1-2):68–99, 2011.

[9] B. Sudakov. A few remarks on Ramsey-Turán-type problems, *J. Combin. Theory Ser. B*, **88**(1):99–106, 2003.

[10] E. Szemerédi. On graphs containing no complete subgraph with 4 vertices, *Mat. Lapok*, 23:113–116, 1973.

[11] E. Szemerédi. Regular partitions of graphs. In Problémes combinatoires et théorie des graphes (Colloq. Internat. CNRS, Univ. Orsay, Orsay, 1976), volume 260 of Colloq. Internat. CNRS, pages 399–401. CNRS, Paris, 1978.

On 4-connected geometric graphs

Alfredo García*[1], Clemens Huemer†[2], Javier Tejel‡[1], and Pavel Valtr§[3]

[1]Dept. Métodos Estadísticos. Universidad de Zaragoza. Spain.
[2]Dept. Matemàtica Aplicada IV. Universitat Politècnica Catalunya, Spain.
[3]Department of Applied Mathematics. Charles University, Czech Republic.

Abstract

Given a set S of n points in the plane, in this paper we give a necessary and sometimes sufficient condition to build a 4-connected non-crossing geometric graph on S.

Introduction

Given a set S of n points in the plane, a non-crossing geometric graph on S is a graph in which its vertices are the points of S and its edges are straight-line segments between these points such that no edge passes through a vertex different from its endpoints and any two edges may intersect only at a common endpoint.

Since all the geometric graphs considered in this paper are non-crossing, throughout the paper we will use the term geometric graph, meaning that the geometric graph is non-crossing.

The study of geometric graphs and, in particular, the study of problems on how to embed planar graphs as geometric graphs on given point sets is a very active area of research (for a review on geometric graphs and some related topics, see for example [1, 4]). One of these problems is the problem of building geometric graphs with a certain connectivity on a set of points S.

We will say that a set of points S is k-connectible if it admits a k-connected geometric graph on it. For $k = 1, 2, 3$, it is well-known when S is k-connectible and how to build a k-connected geometric graph (see [2, 3]). Given S, it is enough to build a non-crossing tree on S (for example the minimum spanning tree of S) when $k = 1$, and it is enough to build a simple polygonization of S when $k = 2$. For the case $k = 3$, the only set of points not admitting a 3-connected geometric graph is the convex case. Otherwise, in [3] the authors give an algorithm to build a 3-connected geometric graph using $\max\{\lceil 3n/2 \rceil, n+m-1\}$ edges, where m is the number of points on the boundary of the convex hull of S, and they prove that there is no 3-connected plane graph on S with less edges.

However, for $k > 3$, little is known about when a set of points is k-connectible. For $k = 4$, Dey et al. [2] show points sets that do not admit any 4-connected geometric graph on them and they provide a necessary and sufficient condition for point sets whose convex hull consists of exactly three points. A general characterization of 4- or 5-connectible sets of points is not known.

In this paper, we study sets of points that are 4-connectible. We define a condition (the U-condition) that any set of points must satisfy to be 4-connectible and we show that the U-condition is always sufficient for some sets of points. By denoting the convex hull of S by $CH(S)$ and the set of points on the boundary of the convex hull of S by $H(S)$, if $Q = H(S)$, $I = S \setminus Q$ and $P = H(I)$, then the U-condition is sufficient for sets of points in which $Q \cup P$ satisfies the U-condition.

1 The U-condition

In this section, we will define the U-condition and we will see that it is a necessary condition to get 4-connected geometric graphs.

A subset C of points of Q is connected if it consists of consecutive points of Q. We will denote by $h(C)$ the number of connected components of a subset C of Q.

Definition 1 A set S of points satisfies the U-condition if

i) $|Q| \leq |I|$

ii) For any set $C \subset Q$, $|H(S \setminus C)| \leq |I| + h(C)$.

Lemma 2 *Every 4-connectible set S satisfies the U-condition.*

*Email: olaverri@unizar.es. Research partially supported by projects Gob. Arag. E58-DGA, MINECO MTM2012-30951 and ESF EUROCORES programme EuroGIGA, CRP ComPoSe: MICINN Project EUI-EURC-2011-4306.

†Email: clemens.huemer@upc.edu. Research partially supported by projects MINECO MTM2012-30951 and ESF EUROCORES programme EuroGIGA, CRP ComPoSe: MICINN Project EUI-EURC-2011-4306.

‡Email: jtejel@unizar.es. Research partially supported by projects Gob. Arag. E58-DGA, MINECO MTM2012-30951 and ESF EUROCORES programme EuroGIGA, CRP ComPoSe: MICINN Project EUI-EURC-2011-4306.

§Email: valtr@kam.mff.cuni.cz.

Proof. Let us prove that if $G(S)$ is a 4-connected graph drawn on S, then S has to satisfy i) and ii). In $G(S)$, each vertex must have degree at least 4, and since there are no edges linking non-consecutive points of Q (otherwise $G(S)$ would not be 3-connected), then there are at least $2|Q|$ edges having an endpoint in Q and the other one in I. On the other hand, as $G(S)$ is 4-connected, each point of I can be linked to a maximum of two (and consecutive) points of Q. Hence, at most there are $2|I|$ edges with an endpoint in Q and the other one in I. Therefore, $2|Q| \leq 2|I|$ and the necessity of i) is proved.

Now, suppose $C \neq \emptyset$ is a subset of points of Q and $C_1, C_2, \ldots, C_{h(C)}$ are its connected components. Let us denote by $\overline{C}_i$ the points of Q placed between component C_i and component C_{i+1}. Thus, Q consists of the points $C_1, \overline{C}_1, C_2, \overline{C}_2, \ldots, C_{h(C)}, \overline{C}_{h(C)}$ in this order. When we remove the points of C, all the points in the subsets $\overline{C}_i$ remain in $H(S \setminus C)$ and perhaps, between $\overline{C}_i$ and $\overline{C}_{i+1}$ (mod $h(C)$), a subset I_{i+1} of points of I appears in $H(S \setminus C)$. Either I_i is empty or it consists of consecutive points of P (see Figure 1). Let $\overline{I}$ be the set $I \setminus (I_1 \cup I_2 \cup \ldots \cup I_{h(C)})$ and let $\overline{C}$ be the set $Q \setminus C$. Observe that proving ii) is equivalent to proving $|\overline{C}| \leq |\overline{I}| + h(C)$.

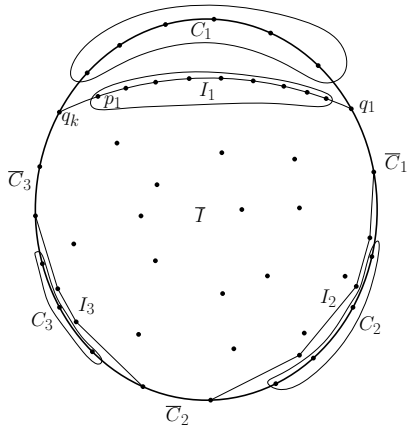


Figure 1: Illustration of Lemma 2.

Suppose that $I_1 = \{p_1, \ldots, p_{k'}\}$ is nonempty and let q_1 and q_k be the points of $\overline{C}$ placed on the boundary of $CH(S \setminus C)$ just after and before the points of I_1, respectively (see Figure 1). Without loss of generality, we can assume that $G(S)$ is a triangulation. So, each point of I_1 must be connected in $G(S)$ to some point of C_1 and, since $G(S)$ is 4-connected, they cannot be connected to a point of $\overline{C}$, except for the edges $p_{k'}q_1$ and $q_k p_1$. Therefore, for an edge linking a point of $\overline{C}$ with an interior point (at least $2|\overline{C}|$ edges), the interior endpoint must be in $I \setminus I_1$, except for the two mentioned edges $p_{k'}q_1$ and $q_k p_1$. We can repeat the same reasoning for every subset I_i, obtaining that there must be at least $2|\overline{C}| - 2h(C)$ edges with an endpoint in $\overline{C}$ and the other one in $\overline{I}$. On the other hand, as before, there are at most $2|\overline{I}|$ connections of this type, so it follows that $2|\overline{C}| - 2h(C) \leq 2|\overline{I}|$. □

Observe that if $|Q| = 3$, then $|I|+1 = n-2$. Hence, part ii) can only fail if we remove one point q of Q and the boundary of $CH(S \setminus q)$ contains all the remaining points. In [2], this is the condition that is proved to be necessary and sufficient to build a 4-connected geometric graph (in fact a triangulation) on S.

Lastly, let us point out that, given S, checking whether S satisfies the U-condition or not can be done in $O(|Q| + |P|)$ steps, after calculating Q and P. The algorithm is based in the observation (not easy to prove) that it is not necessary to compute $CH(S \setminus C)$ for all the possible subsets C, but only for a linear number of them.

2 Some 4-connectible sets

In this section, we will give some sets of points for which the U-condition is sufficient. In particular, we will see that if $Q \cup P$ satisfies the U-condition for a set of points S, then S is 4-connectible. We will use $\overline{Q}$ ($\overline{P}$) to refer to the convex polygon defined by the points of Q (P).

Let us start with the case in which S is precisely $Q \cup P$ and $|Q| = |P|$.

Lemma 3 *Let $Q = \{q_1, \ldots, q_n\}$ be a set of points in convex position and let $P = \{p_1, \ldots, p_n\}$ be another set of points in convex position such that $\overline{P}$ is inside $\overline{Q}$. Suppose that the set of points $S = Q \cup P$ satisfies the U-condition. Then S is 4-connectible.*

Proof. Let M be the region $CH(Q) \setminus CH(P)$. To prove the lemma, it is enough to obtain a crossing free zig-zag cycle $Z = p_i q_j p_{i+1} q_{j+1} \ldots p_{i-1} q_{j-1} p_i$ such that its edges are in M, because then the edges of Z and the edges of $\overline{Q}$ and $\overline{P}$ define a 4-connected graph (see Figure 2).

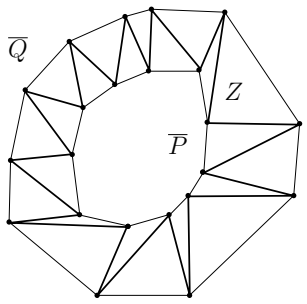


Figure 2: A 4-connected geometric graph when $S = Q \cup P$.

We will say that a triangle $q_j p_i p_{i+1}$ is legal if it is contained in region M. Proving the lemma is equivalent to proving that there is a sequence of n

consecutive legal triangles $q_j p_i p_{i+1}$, $q_{j+1} p_{i+1} p_{i+2}, \ldots,$ $q_{j+n-1} p_{i+n-1} p_{i+n}$, where points with equal subscripts modulo n are considered identical.

Let us assume that $\{q_1, \ldots, q_m\}$ is the set of clockwise points of Q on the left of the line $p_1 p_2$. The following algorithm computes a sequence of n consecutive legal triangles.

Begin
Do $i = j = 2$
While $(i \leq n)$ **Do**
(* **Invariant (1)**: Triangles of the sequence $q_{j-1} p_{i-1} p_i, q_{j-2} p_{i-2} p_{i-1}, \ldots, q_{j-(i-1)} p_1 p_2$ are legal.*)
If (Triangle $q_j p_i p_{i+1}$ is legal) **then**
Do $\{i = i + 1; j = j + 1\}$
Else
(* **Invariant (2)**: q_j is between q_{j-1} and the first crossing of line $p_i p_{i+1}$ with $\overline{Q}$ *)
Do $j = j + 1$
End of While
(* After finishing the algorithm, all the triangles of the sequence $q_{j-n} p_1 p_2, q_{j-(n-1)} p_2 p_3, \ldots, q_{j-1} p_n p_1$ are legal.*)

Let us see that assertions (1) and (2) are always true, so they are invariant in the algorithm.

Trivially, assertion (1) is true the first time because $q_1 p_1 p_2$ is legal by hypothesis. Now, suppose that assertions (1) and (2) are true in the iterations $1, 2, \ldots, k$ of the loop and let us prove that (1) is still true in the following iteration. If we begin the $k+1$ iteration after exploring a legal triangle in the iteration k, then clearly (1) is still true (because we are adding the last explored triangle to a previous legal sequence). If we begin iteration $k+1$ after exploring an illegal triangle $q_j p_i p_{i+1}$ in iteration k, then we need to check that the new sequence $ST = q_j p_{i-1} p_i, \ldots, q_{j-i+2} p_1 p_2$ of triangles (where j has been increased by one) is legal. Assume to the contrary that in this sequence ST a first illegal triangle $q_{j-h} p_{i-h-1} p_{i-h}$ appears, so q_{j-h} is the first clockwise point of Q on the right of line $p_{i-h-1} p_{i-h}$. By removing the points of Q from q_{j+1} to q_{j-h-1} (see Figure 3 left), then the points of P from p_i to p_{i-h} ($n-h+1$ points) and the points of Q from q_{j-h} to q_j ($h+1$ points) appear in the boundary of the new convex hull, contradicting the U-condition (at most $n+1$ points can appear in the boundary of the new convex hull). Therefore (1) is invariant.

For assertion (2), the first time that the algorithm goes to the else branch, we are exploring the illegal triangle $q_j p_j p_{j+1}$, being the triangles $q_{j-1} p_{j-1} p_j, \ldots, q_1 p_1 p_2$ legal. If q_j was on the right side of $p_i p_{i+1}$ and after the second crossing point of that line with $\overline{Q}$, then, by removing the points of Q from q_1 to q_{j-1} (remember that q_1 is the first point of Q to the left of $p_1 p_2$), the U-condition is contradicted because in the boundary of the new convex hull $n+2$ points appear (the points of P from p_1 to p_{j+1} and the points of Q from q_j to q_n). Therefore, in the first visit to the else branch assertion (2) is true.

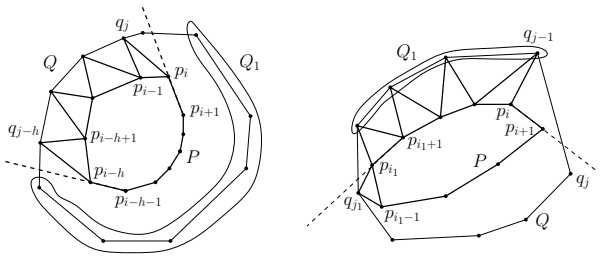


Figure 3: Illustration of Lemma 3. Left: Supposing q_{j-h} is on the right of $p_{i-h-1} p_{i-h}$, the U-condition fails if Q_1 is removed. Right: Supposing q_j is on the right of $p_i p_{i+1}$, the U-condition fails if Q_1 is removed.

Suppose that the last illegal triangle explored is $q_{j_1} p_{i_1} p_{i_1+1}$ and the algorithm is exploring a new illegal triangle $q_j p_i p_{i+1}$. As (2) was true in the previous iterations, point q_{j_1} has to be placed between q_{j_1-1} and the first crossing of line $p_{i_1} p_{i_1+1}$ with $\overline{Q}$. After exploring this triangle, subscript j_1 is increased by one, and then h operations (perhaps $h = 0$) of increasing both subscripts (i and j) are done. Therefore, it must be $j = j_1 + h + 1$ and $i = i_1 + h$, for some $h \geq 0$. Since (1) is invariant, the $h+1$ triangles $q_{j-1} p_{i-1} p_i, q_{j-2} p_{i-2} p_{i-1}, \ldots, q_{j_1} p_{i_1-1} p_{i_1}$ are legal. Therefore, if q_j is placed after the second crossing of line $p_i p_{i+1}$ with $\overline{Q}$, then, by removing the h points of Q from q_{j_1+1} to q_{j-1}, the $h+2$ points of P from p_{i_1+1} to p_{i+1} appear in the boundary of the new convex hull, contradicting the U-condition (see Figure 3 right). Hence, (2) is invariant.

Lastly, since (1) is invariant and the last triangle, $q_{j-(i-1)} p_1 p_2$, is legal, then the subscript $j - i + 1$ has to be between 1 and m. This implies that $j \leq i + m - 1$ in the algorithm. Hence, the algorithm can go to the else branch a maximum of $m - 1$ times, and finishes in a maximum number of $n + m - 1$ steps. When the algorithm finishes, then $i = n + 1$ and the triangles of the sequence $q_{j-1} p_n p_{n+1}, q_{j-2} p_{n-1} p_n, \ldots, q_{j-n} p_1 p_2$ are legal because (1) is invariant. □

Now, assume that $|Q| = |P|$, $Q \cup P$ satisfies the U-condition, there are more points inside $\overline{P}$ and that $|Q| > 3$ (the case $|Q| = 3$ was solved in [2]). To get a 4-connected geometric graph, we proceed as follows. First draw the zig-zag including alternatively the points of P and Q, according to the previous lemma. Then, take a diagonal of $\overline{P}$, for example diagonal $p_1 p_3$. This diagonal divides $\overline{P}$ into two subpolygons $P_1 = \{p_1, p_3, \ldots, p_n, p_1\}$ and $P_1' = \{p_1, p_2, p_3, p_1\}$. If the interior $I(P_1)$ of P_1 is nonempty, then let P_2 be the convex polygon defined by the points $H(I(P_1) \cup p_1, p_3)$. If the interior $I(P_2)$ of this polygon is again nonempty,

then we define P_3 as the convex polygon defined by the points $H(I(P_2) \cup p_1, p_3)$, and so on, until we obtain an empty convex polygon P_h. Thus, we have a sequence $P_h \subset P_{h-1} \subset \ldots \subset P_2 \subset P_1$ of nested polygons (see Figure 4 left). The same process can be done starting at P'_1, obtaining another sequence $P'_{h'} \subset P'_{h'-1} \subset \ldots \subset P'_2 \subset P'_1$ of nested polygons. Observe that the region M_i (M'_i) bounded by the consecutive polygons P_{i+1} and P_i (P'_{i+1} and P'_i) has the shape of a "half-moon". It is not difficult to prove that M_i (M'_i) can be triangulated such that points p_1 and p_3 are not used, and each one of the added edges has an endpoint in P_i (P'_i) and the other one in P_{i+1} (P'_{i+1}). Then, we triangulate all the half-moons in this way (using edges connecting points placed in different polygons) and we triangulate the convex polygon P_f, formed by concatenating P_h and $P'_{h'}$, such that the only points with degree two are p_1 and p_3. It can be easily checked that the triangulation obtained in this way is 4-connected (see Figure 4 right).

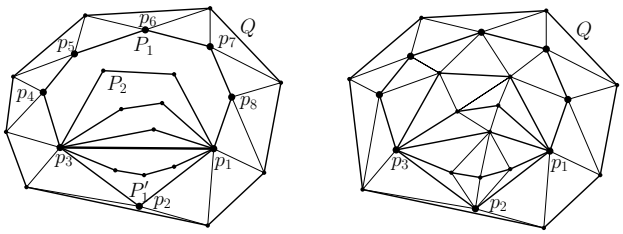


Figure 4: The general construction when $|Q| = |P|$ and there are points inside $\overline{P}$.

The case in which $|Q| < |P|$ and $Q \cup P$ satisfies the U-condition is solved in a similar way. First a graph based on a zig-zag is built, although this starting graph cannot be a zig-zag as in the previous case, because $|Q| < |P|$. Now, the starting graph is a zig-zag connecting the points of Q to some points of P plus some additional edges connecting the points of P not belonging to the zig-zag to some points of Q (bold edges in Figure 5 left). After building this starting graph, we take a diagonal of $\overline{P}$ connecting two points of P, consecutive in the zig-zag but not consecutive in P (points p_{i_1} and p_{i_2} in Figure 5), and we proceed as in the previous case, adding the triangulations of the different half-moons and the final triangulation of the convex polygon P_f (see Figure 5 right). The resulting triangulation is 4-connected.

The U-condition is the key to finding this starting graph (the zig-zag plus some additional edges), although proving the existence of such a graph is not obvious. Due to space limitations, we do not include this proof.

Therefore, we have proved the following theorem.

Theorem 4 *Let S be a set of points. If $Q = H(S)$, $P = H(S \setminus Q)$ and $Q \cup P$ satisfies the U-condition, then S is 4-connectible.*

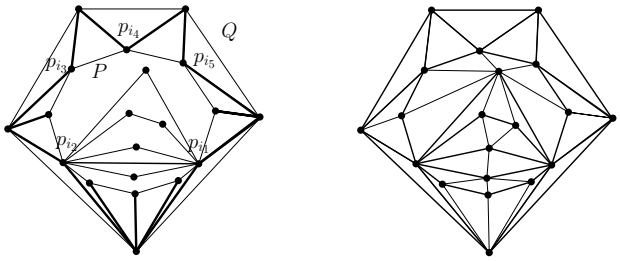


Figure 5: The general construction when $|Q| < |P|$ and there are points inside $\overline{P}$.

3 Conclusions

In this paper, we have defined a condition, the U-condition, that any set of points S must satisfy to be 4-connectible. Moreover, we have proved that the U-condition is also sufficient for sets of points in which $Q \cup P$ satisfies the U-condition.

In [2], the case $|Q| = 3$ is completely solved. Given a set S of points such that $|Q| = 3$, the authors show how to build a 4-connected triangulation on S, except for a particular configuration of points. This particular configuration is precisely the only one not satisfying the U-condition, among all the configurations of points such that $|Q| = 3$.

Using different techniques not included in this paper, we can extend the family of 4-connectible sets. For any set S of points satisfying the U-condition (it is not required that $Q \cup P$ satisfies the U-condition) such that $|P| = 3$ or $|P| = 4$, we can built a 4-connected geometric graph on S.

Finally, we conclude with the following conjecture.

Conjecture 1 *If a set of points S satisfies the U-condition, then S is 4-connectible.*

References

[1] P. Brass, W. Moser and J. Pach, *Research Problems in Discrete Geometry*, Springer-Verlag, Berlin, 2005.

[2] T.K. Dey, M.B. Dillencourt, S.K. Ghosh and J.M. Cahill, Triangulating with high connectivity, *Comput. Geom. Theory Appl.* **8** (1997), 39–56.

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

[4] J. Pach (ed.), *Thirty Essays on Geometric Graph Theory*, Springer Science+Business Media, New York, 2013.

Monotone crossing number of complete graphs

Martin Balko*[1], Radoslav Fulek†[1], and Jan Kynčl‡[2]

[1]Department of Applied Mathematics, Charles University, Faculty of Mathematics and Physics, Malostranské nám. 25, 118 00 Praha 1, Czech Republic;
[2]Department of Applied Mathematics and Institute for Theoretical Computer Science, Charles University, Faculty of Mathematics and Physics, Malostranské nám. 25, 118 00 Praha 1, Czech Republic;

Abstract

In 1958, Hill conjectured that the minimum number of crossings in a drawing of K_n is exactly $Z(n) = \frac{1}{4}\lfloor\frac{n}{2}\rfloor\lfloor\frac{n-1}{2}\rfloor\lfloor\frac{n-2}{2}\rfloor\lfloor\frac{n-3}{2}\rfloor$. Generalizing the result by Ábrego et al. for 2-page book drawings, we prove this conjecture for plane drawings in which edges are represented by x-monotone curves. In fact, our proof shows that the conjecture remains true for x-monotone drawings in which adjacent edges do not cross and we count only pairs of edges which cross odd number of times. We also discuss a combinatorial characterization of these drawings.

1 Introduction

Let G be a graph with no loops and multiple edges. In a *drawing D of a graph G* in the plane, the vertices are represented by distinct points and each edge is represented by a simple continuous arc connecting the images of its endpoints. As usual, we identify the vertices and their images, as well as the edges and the arcs representing them. It is required that the edges pass through no vertices other than their endpoints. We also assume for simplicity that any two edges have only finitely many points in common, no two edges *touch* at an interior point and no three edges meet at a common interior point.

A *crossing in D* is a common interior point of two edges where they properly cross. The *crossing number* $\mathrm{cr}(D)$ *of a drawing D* is the number of crossings in D. The *crossing number* $\mathrm{cr}(G)$ *of a graph G* is the minimum of $\mathrm{cr}(D)$, taken over all drawings D of G. A drawing D is called *simple* if no two adjacent edges cross and no two edges have more than one common crossing. It is well known and easy to see that every drawing of G which minimizes the crossing number is simple.

According to the famous conjecture of Hill [7, 8] (also known as Guy's conjecture), the crossing number of the complete graph K_n on n vertices satisfies $\mathrm{cr}(K_n) = Z(n)$, where

$$Z(n) = \frac{1}{4}\left\lfloor\frac{n}{2}\right\rfloor\left\lfloor\frac{n-1}{2}\right\rfloor\left\lfloor\frac{n-2}{2}\right\rfloor\left\lfloor\frac{n-3}{2}\right\rfloor.$$

This conjecture has been verified for $n \leq 12$ [12] and for each n, there are drawings of K_n with $Z(n)$ crossings [6, 7, 8, 9].

A curve α in the plane is *x-monotone* if every vertical line intersects α in at most one point. A drawing of a graph G in which every edge is represented by an x-monotone curve and no two vertices share the same x-coordinate is called *x-monotone* (or *monotone*, for short). The *monotone crossing number mon-cr(G) of a graph G* is the minimum of $\mathrm{cr}(D)$, taken over all monotone drawings D of G.

The *rectilinear crossing number* $\overline{\mathrm{cr}}(G)$ *of a graph G* is the smallest number of crossings in a drawing of G where every edge is represented by a straight-line segment. Since every rectilinear drawing of G in which no two vertices share the same x-coordinate is x-monotone, we have $\mathrm{cr}(G) \leq \text{mon-cr}(G) \leq \overline{\mathrm{cr}}(G)$ for every graph G.

We call a drawing of a graph *semisimple* if adjacent edges do not cross but independent edges may cross more than once. The *monotone semisimple odd crossing number* of G (called *monotone odd* + by Schaefer [14]), denoted by $\text{mon-ocr}_+(G)$, is the smallest number of pairs of edges that cross an odd number of times in a monotone semisimple drawing of G. Clearly, $\text{mon-ocr}_+(G) \leq \text{mon-cr}(G)$.

The monotone crossing number has been introduced by Valtr [15] and recently further investigated by Pach and Tóth [11], who showed that $\text{mon-cr}(G) < 2\mathrm{cr}(G)^2$ holds for every graph G. On the other hand,

*Email: balko@kam.mff.cuni.cz. Research supported by the grant GACR GIG/11/E023 GraDR in the framework of ESF EUROGIGA program, by the Grant Agency of the Charles University, GAUK 1262213, and by the grant SVV-2013-267313 (Discrete Models and Algorithms).

†Email: radoslav@kam.mff.cuni.cz. Research supported by the grant GACR GIG/11/E023 GraDR in the framework of ESF EUROGIGA program.

‡Email: kyncl@kam.mff.cuni.cz. Research supported by the grant GACR GIG/11/E023 GraDR in the framework of ESF EUROGIGA program, by the Grant Agency of the Charles University, GAUK 1262213, and by the grant SVV-2013-267313 (Discrete Models and Algorithms).

they showed that the monotone crossing number and the crossing number are not always the same: there are graphs G with arbitrarily large crossing numbers such that $\text{mon-cr}(G) \geq \frac{7}{6}\text{cr}(G) - 6$.

We study the monotone crossing numbers of complete graphs. The drawings of complete graphs with $Z(n)$ crossings obtained by Blažek and Koman [6] (see also [9]) are 2-*page book* drawings, which may be considered as a strict subset of x-monotone drawings. Thus we have $\text{mon-cr}(K_n) \leq Z(n)$. Ábrego et al. [1] recently proved Hill's conjecture for 2-page book drawings of complete graphs. We generalize their techniques and show that Hill's conjecture holds for all x-monotone drawings of complete graphs, even for the monotone semisimple odd crossing number.

Theorem 1 *For every $n \in \mathbb{N}$, we have*

$$\text{mon-ocr}_+(K_n) = \text{mon-cr}(K_n) = Z(n).$$

The rectilinear crossing number of K_n is known to be asymptotically larger than $Z(n)$: this follows from the best current lower bound $\overline{\text{cr}}(K_n) \geq (277/729)\binom{n}{4} - O(n^3)$ [3, 5] and from the simple upper bound $Z(n) \leq \frac{3}{8}\binom{n}{4} + O(n^3)$.

See a recent survey by Schaefer [14] for an encyclopedic treatment of all known variants of crossing numbers.

After submitting this extended abstract, we were informed that the authors of [1] achieved the result $\text{mon-cr}(K_n) = Z(n)$ already during discussions after their presentation at SoCG 2012, and that it will appear in the proceedings of LAGOS 2013 [2].

2 Monotone Crossing Number

To prove the upper bound on the 2-page crossing number of K_n, Ábrego et al. [1] generalized the notion of k-edges to arbitrary simple drawings of complete graphs. They also introduced the notion of $\leq\leq k$-edges. These capture the essential properties of 2-page book drawings better than $\leq k$-edges, which had been successfully used before for rectilinear and pseudolinear drawings [10, 4, 3]. We show that the approach using $\leq\leq k$-edges can be generalized to arbitrary semisimple x-monotone drawings.

For a semisimple drawing D of K_n and distinct vertices u and v of K_n, let γ be the oriented arc representing the edge $\{u, v\}$. If w is a vertex of K_n different from u and v, then we say that w is *on the left (right) side of* γ if the topological triangle uvw with vertices u, v and w traced in this order is oriented counterclockwise (clockwise, respectively). This generalizes the definition introduced by Ábrego et al. [1] for simple drawings. However, we were not able to find a meaningful generalization of this notion to drawings that are not semisimple, where the edges of the triangle uvw can cross several times.

A *k-edge* is an edge $\{u, v\}$ of D that has exactly k points on the same side (left or right). Since every k-edge has $n-2-k$ points on the other side, every k-edge is also an $(n-2-k)$-edge and so every edge of D is a k-edge for some integer k where $0 \leq k \leq \lfloor n/2 \rfloor - 1$.

An i-edge with $i \leq k$ is called a *$\leq k$-edge*. Let $E_i(D)$ be the number of i-edges and $E_{\leq k}(D)$ the number of $\leq k$-edges of D. Clearly, $E_{\leq k}(D) = \sum_{i=0}^{k} E_i(D)$. Similarly, the number of $\leq\leq k$-edges of D, $E_{\leq\leq k}(D)$, is defined by the following identity.

$$E_{\leq\leq k}(D) = \sum_{j=0}^{k} E_{\leq j}(D) = \sum_{i=0}^{k} (k+1-i) E_i(D) \quad (1)$$

Considering the only three different simple drawings of K_4 up to a homeomorphism of the plane, Ábrego et al. [1] showed that the number of crossings in a simple drawing D of K_n can be expressed in terms of the number of k-edges in the following way.

Lemma 2 ([1]) *For every simple drawing D of K_n we have*

$$\text{cr}(D) = 3\binom{n}{4} - \sum_{k=0}^{\lfloor n/2 \rfloor - 1} k(n-2-k) E_k(D), \quad (2)$$

which can be equivalently rewritten as

$$\text{cr}(D) = 2 \sum_{k=0}^{\lfloor n/2 \rfloor - 2} E_{\leq\leq k}(D) - \frac{1}{2}\binom{n}{2}\left\lfloor \frac{n-2}{2} \right\rfloor - \frac{1}{2}\left(1 + (-1)^n\right) E_{\leq\leq \lfloor n/2 \rfloor - 2}(D).$$

In fact, Lemma 2 can be easily generalized to semisimple drawings of K_n where $\text{cr}(D)$ is replaced by $\text{ocr}(D)$, which counts the number of pairs of edges that cross an odd number of times in D. The main reason is that the cycle C_4 cannot be drawn in the plane in such a way that both its pairs of opposite edges cross oddly while adjacent edges do not cross.

By Lemma 2, lower bounds on $E_k(D)$ imply lower bounds on $\text{cr}(D)$ and $\text{ocr}(D)$. Considering $\leq k$-edges, Ábrego and Fernández-Merchant [4] and Lovász et al. [10] proved that for rectilinear drawings of K_n, the inequality $E_{\leq k} \geq 3\binom{k+2}{2}$ together with (2) gives $\overline{\text{cr}}(G) \geq Z(n)$. However, there are simple x-monotone (even 2-page) drawings of K_n where $E_{\leq k} < 3\binom{k+2}{2}$ for $k = 1$ [1]. Ábrego et al. [1] showed that similar inequality for $\leq\leq k$-edges is satisfied by all 2-page book drawings. We show that the same inequality is satisfied by all x-monotone semisimple drawings of K_n.

Let $\{v_1, v_2, \ldots, v_n\}$ be the vertex set of K_n. Note that we can assume that all vertices in an x-monotone

drawing lie on the x-axis. We also assume that the x-coordinates of the vertices satisfy $x(v_1) < x(v_2) < \cdots < x(v_n)$.

Observation 3 *Let D be a semisimple drawing of K_n, not necessarily x-monotone. Let v be a vertex incident with the outer face of D and let γ_i be the ith edge incident with v in the counter-clockwise cyclic order such that γ_1 and γ_{n-1} are incident with the outer face in a small neighborhood of v. Let v_{k_i} be the other endpoint of γ_i. Then for every i, j, $1 \le i < j \le n-1$, the triangle $v_{k_i}vv_{k_j}$ is oriented clockwise. Consequently, for every k, $1 \le k \le (n-1)/2$, the edges γ_k and γ_{n-k} are $(k-1)$-edges. For even n, the edge $\gamma_{n/2}$ is a halving edge.* □

For an x-monotone drawing D of K_n, we use Observation 3 directly for the vertex v_n and then for each i, for the vertex v_i and the subgraph induced by $v_i, v_{i+1}, \ldots, v_n$.

The following definitions were introduced by Ábrego et al. [1] for 2-page book drawings. Let D be a semisimple x-monotone drawing of K_n and let D' be a drawing obtained from D by deleting the vertex v_n together with its adjacent edges. A k-edge in D is a *(D, D')-invariant k-edge* if it is also a k-edge in D'. It is easy to see that every $\le k$-edge in D' is also a $\le(k+1)$-edge in D. If $0 \le j \le k \le \lfloor n/2 \rfloor - 1$, then a (D, D')-invariant j-edge is called a *(D, D')-invariant $\le k$-edge*. Let $E_{\le k}(D, D')$ denote the number of (D, D')-invariant $\le k$-edges.

For $i < j$, the edge v_iv_j is called the *right edge at* v_i. The right edges at v_i have a natural vertical order.

Lemma 4 *Let k be a fixed integer such that $0 \le k \le (n-3)/2$. For every $i \in \{1, 2, \ldots, k+1\}$, the $k+2-i$ bottommost and the $k+2-i$ topmost right edges at v_i are $\le k$-edges in D. Moreover, at least $k+2-i$ of these $\le k$-edges are (D, D')-invariant $\le k$-edges.*

Proof. The first part of the lemma follows directly from Observation 3. If the edge v_iv_n is one of the $k+2-i$ topmost right edges at v_i, then the $k+2-i$ bottommost right edges at v_i are (D, D')-invariant $\le k$-edges. Otherwise the $k+2-i$ topmost right edges at v_i are (D, D')-invariant $\le k$-edges. □

Corollary 5 *We have*

$$E_{\le k}(D, D') \ge \sum_{i=1}^{k+1}(k+2-i) = \binom{k+2}{2}.$$

□

The following theorem gives the lower bound on the number of $\le\le k$-edges. The proof is essentially the same as in [1], we only extracted Lemma 4, which needed to be generalized. Together with Lemma 2, Theorem 6 yields Theorem 1.

Theorem 6 *Let $n \ge 3$ and let D be a semisimple x-monotone drawing of K_n. Then for every k, $0 \le k < n/2 - 1$, we have $E_{\le\le k}(D) \ge 3\binom{k+3}{3}$.*

Proof. The proof proceeds by induction on n where the case $n = 3$ is trivially true. Let $n \ge 4$ and let D be a semisimple x-monotone drawing of K_n. For the induction step we remove the point v_n together with its adjacent edges to obtain a drawing D' of K_{n-1}, which is also semisimple and x-monotone.

Using Observation 3 we see that for $0 \le i \le k < n/2 - 1$ there are two i-edges adjacent to v_n in D and together they contribute with $2\sum_{i=0}^{k}(k+1-i) = 2\binom{k+2}{2}$ to $E_{\le\le k}(D)$ by (1).

Let γ be an i-edge in D'. Then γ contributes by $(k-i)$ to the sum $E_{\le\le k-1}(D') = \sum_{i=0}^{k-1}(k-i)E_i(D')$. We already observed that γ is either an i-edge or an $(i+1)$-edge in D. If γ is also an i-edge in D (that is, γ is a (D, D')-invariant i-edge), then it contributes by $(k+1-i)$ to $E_{\le\le k}(D)$. This is a gain of $+1$ towards $E_{\le\le k-1}(D')$. If γ is an $(i+1)$-edge in D, then it contributes only $(k-i)$ to $E_{\le\le k}(D)$. Therefore we have

$$E_{\le\le k}(D) = 2\binom{k+2}{2} + E_{\le\le k-1}(D') + E_{\le k}(D, D').$$

By the induction hypothesis we know that $E_{\le\le k-1}(D') \ge 3\binom{k+2}{3}$ and thus we obtain

$$E_{\le\le k}(D) \ge 3\binom{k+3}{3} - \binom{k+2}{2} + E_{\le k}(D, D').$$

The theorem follows by plugging the lower bound from Corollary 5. □

3 Combinatorial Description

In this section we develop a combinatorial characterization of x-monotone drawings which is based on the signature function introduced by Peters and Szekeres [13] for describing order types of points sets. Let T_n be the set of ordered triples (i, j, k) of the set $[n] = \{1, 2, \ldots, n\}$ and let Σ_n be the set of *signature functions* $\sigma\colon T_n \to \{-, +\}$.

Let D be an x-monotone drawing of the complete graph $K_n = (V, E)$ with vertices $v_1, v_2, \ldots, v_n$ such that their x-coordinates satisfy $x(v_1) < x(v_2) < \cdots < x(v_n)$. We assign a signature function $\sigma \in \Sigma_n$ to the drawing D according to the following rule. For each $e = \{v_i, v_k\} \in E$ and every integer j, $i < j < k$, let $\sigma(i, j, k) = -$ if the point v_j lies above the arc representing the edge e and $\sigma(i, j, k) = +$ otherwise.

Note that if the drawing D is also semisimple, then a triangle $v_iv_kv_j$, $j \in (i, k)$, is oriented counterclockwise (clockwise) if and only if $\sigma(i, j, k) = -$ ($\sigma(i, j, k) = +$, respectively). It is easy to see that

for every signature function $\sigma \in \Sigma_n$ there is an x-monotone drawing D which induces σ. However, such a drawing does not have to be semisimple. We show a characterization of simple and semisimple x-monotone drawings by small forbidden configurations in the signature functions.

For $a, b, c, d \in [n]$ with $a < b < c < d$ and a signature function $\sigma \in \Sigma_n$ we say that the 4-tuple (a, b, c, d) *is of the form* $\xi_1\xi_2\xi_3\xi_4$ *in* σ if $\sigma(a, b, c) = \xi_1, \sigma(a, b, d) = \xi_2, \sigma(a, c, d) = \xi_3$ and $\sigma(b, c, d) = \xi_4$.

Theorem 7 *A signature function $\sigma \in \Sigma_n$ can be realized by a semisimple x-monotone drawing if and only if each ordered 4-tuple of indices is of one of the forms $++++$, $----$, $++--$, $--++$, $-++-$, $+--+$, $---+$, $+++-$, $+---$, $-+++$ in σ. The signature function σ can be realized by a simple x-monotone drawing if, in addition, there is no 5-tuple (a, b, c, d, e), $a < b < c < d < e$, with*

$$\sigma(a, b, e) = \sigma(a, d, e) = \sigma(b, c, d) = -\sigma(a, c, e).$$

Note that in a simple x-monotone drawing of K_n the crossings can appear only between edges whose endpoints induce a 4-tuple of one of the forms $++++, ----, ++--, --++, -++-, +--+$. Analogously to a similar correspondence in rectilinear drawings of K_n, we may call these 4-tuples *convex*. Then for a simple x-monotone drawing D of K_n the crossing number of D equals the number of convex 4-tuples.

A similar notion of convexity for general k-tuples was used by Peters and Szekeres [13]. This description of crossings is convenient for computer calculations. Using it, we have obtained a complete list of optimal x-monotone drawings of K_n for $n \leq 10$.

4 Concluding remarks

It is an interesting direction of further research to see if similar techniques can be helpful in proving Hill's conjecture for general drawings of complete graphs. We note that the same approach does not generalize to all drawings: for example, a particular planar realization of the so-called *cylindrical drawing* [7, 8] of K_{10}, with crossing number $Z(10)$, does not satisfy the lower bound on $\leq\leq$1-edges in Theorem 6. It would also be interesting to further generalize Theorem 1 to monotone drawings where also adjacent edges are allowed to cross.

Acknowledgments

We would like to thank Pavel Valtr for initializing the research which led to this problem and Marek Eliáš for developing visualization tools that were helpful during the research.

References

[1] B. M. Ábrego, O. Aichholzer, S. Fernández-Merchant, P. Ramos, and G. Salazar, The 2-page crossing number of K_n, preprint, 2012, arXiv:1206.5669.

[2] B. M. Ábrego, O. Aichholzer, S. Fernández-Merchant, P. Ramos, and G. Salazar, More on the crossing number of K_n: Monotone drawings, Proceedings of the VII Latin-American Algorithms, Graphs and Optimization Symposium (LAGOS) 2013, Playa del Carmen, Mexico (to appear).

[3] B. M. Ábrego, M. Cetina, S. Fernández-Merchant, J. Leaños, and G. Salazar, On $\leq k$-edges, crossings, and halving lines of geometric drawings of K_n, *Discrete Comput. Geom.* **48** (2012), 192–215.

[4] B. M. Ábrego and S. Fernández-Merchant, A lower bound for the rectilinear crossing number, *Graphs Combin.* **21** (2005), 293–300.

[5] B. M. Ábrego, S. Fernández-Merchant, J. Leaños, and G. Salazar, A central approach to bound the number of crossings in a generalized configuration, in: *The IVLatin-American Algorithms, Graphs, and Optimization Symposium*, Electron. Notes Discrete Math., 30, Elsevier Sci. B. V., Amsterdam, 2008, 273–278.

[6] J. Blažek and M. Koman, A minimal problem concerning complete plane graphs, in: *Theory of Graphs and its Applications*, Proc. Sympos. Smolenice, 1963, Publ. House Czechoslovak Acad. Sci., Prague, 1964, 113–117.

[7] R. K. Guy, A combinatorial problem, *Nabla (Bull. Malayan Math. Soc)* **7** (1960), 68–72.

[8] F. Harary and A. Hill, On the number of crossings in a complete graph, *Proc. Edinburgh Math. Soc. (2)* **13** (1963), 333–338.

[9] H. Harborth, Special numbers of crossings for complete graphs, *Discrete Mathematics* **244** (2002), 95–102.

[10] L. Lovász, K. Vesztergombi, U. Wagner, and E. Welzl, Convex quadrilaterals and k-sets, in: *Towards a theory of geometric graphs*, Contemp. Math., 7034, Amer. Math. Soc., Providence, RI, 2004, 139–148.

[11] J. Pach and G. Tóth, Monotone Crossing Number, in: *Graph drawing*, Lecture Notes in Comput. Sci., 7034, Springer, Berlin Heidelberg, 2012, 278–289.

[12] S. Pan and B. R. Richter, The crossing number of K_{11} is 100, *J. Graph Theory* **56** (2007), 128–134.

[13] G. Szekeres and L. Peters, Computer solution to the 17-point Erdős-Szekeres problem, *ANZIAM J.* **48** (2006), 151–164.

[14] M. Schaefer, The Graph Crossing Number and its Variants: A Survey, *Electronic Journal of Combinatorics*, Dynamic Survey 21 (2013).

[15] P. Valtr, On the pair-crossing number, in: *Combinatorial and computational geometry*, Math. Sci. Res. Inst. Publ., 52, Cambridge Univ. Press, Cambridge, 2005, 569–575.

Flips in combinatorial pointed pseudo-triangulations with face degree at most four (extended abstract)

Oswin Aichholzer*[1], Thomas Hackl*[1], David Orden†[2], Alexander Pilz*[1], Maria Saumell‡[3], and Birgit Vogtenhuber*[1]

[1]Institute for Software Technology, Graz University of Technology, Austria.
[2]Departamento de Física y Matemáticas, Universidad de Alcalá, Spain.
[3]Département d'Informatique, Université Libre de Bruxelles, Belgium.

Abstract

In this paper we consider the flip operation for combinatorial pointed pseudo-triangulations where faces have size 3 or 4, so-called *combinatorial 4-PPTs*. We show that every combinatorial 4-PPT is stretchable to a geometric pseudo-triangulation, which in general is not the case if faces may have size larger than 4. Moreover, we prove that the flip graph of combinatorial 4-PPTs with triangular outer face is connected and has diameter $O(n^2)$.

1 Introduction

Given a graph of a certain class, a *flip* is the operation of removing one edge and inserting a different one such that the resulting graph is again of the same class. For the class of maximal planar (simple) graphs, any combinatorial embedding (clockwise order of edges around each vertex) has only faces of size 3 and hence is called a *combinatorial triangulation*. Flips in combinatorial triangulations remove the common edge of two triangular faces and replace it by the edge between the two vertices not shared by the faces, provided that these two vertices where not already joined by an edge. Combinatorial triangulations have a geometric counterpart in triangulations of point sets in the plane, which are maximal plane geometric (straight-line) graphs with predefined vertex positions. In this geometric setting there is also a flip operation, for which a different restriction applies: An edge can be flipped if and only if the two adjacent triangles form a convex quadrilateral (otherwise the new edge would create a crossing).

Flips in (combinatorial) triangulations have been thoroughly studied. See [4] for a survey. A prominent question about flips is to study the *flip graph*. This is an abstract graph whose vertices are the members of the same graph class having the same number of vertices, and in which two graphs are neighbors if and only if one can be transformed into the other by a single flip. For both, combinatorial triangulations and triangulations (with fixed vertex positions), the flip graph is connected. However, the different settings imply linear and quadratic diameter, respectively (see [4] for references).

Triangulations have a natural generalization in pseudo-triangulations. They have become a popular structure in Computational Geometry within the last two decades, with applications in, e.g., rigidity theory and motion planning. See [7] for a survey. A *pseudo-triangle* is a simple polygon in the plane with exactly three convex vertices (i.e., vertices whose interior angle is smaller than π). A *pseudo-triangulation* $\mathcal{T}$ of a finite point set S in the plane is a partition of the convex hull of S into pseudo-triangles such that the union of the vertices of the pseudo-triangles is exactly S. Triangulations are a particular type of pseudo-triangulations, actually the ones with the maximum number of edges. Those with the minimum number of edges are the so-called *pointed pseudo-triangulations*, in which every vertex is *pointed*, i.e., incident to a reflex angle (an angle larger than π).

Flips can also be defined for the class of pseudo-triangulations of point sets in the plane. The flip graph for general pseudo-triangulations is known to be connected, as well as the subgraph induced by pointed pseudo-triangulations. The currently best known bound on the diameter is $O(n \log n)$ for both

*Email: [oaich|thackl|apilz|bvogt]@ist.tugraz.at. Research of OA and BV partially supported by the ESF EUROCORES programme EuroGIGA – CRP 'ComPoSe', Austrian Science Fund (FWF): I648-N18. Research of TH supported by the Austrian Science Fund (FWF): P23629-N18 'Combinatorial Problems on Geometric Graphs'. AP is a recipient of a DOC-fellowship of the Austrian Academy of Sciences.

†Email: david.orden@uah.es. Research partially supported by MICINN Project MTM2011-22792, ESF EUROCORES programme EuroGIGA - ComPoSe IP04 - MICINN Project EUI-EURC-2011-4306 and Junta de Castilla y León Project VA172A12-2.

‡Email: maria.saumell.m@gmail.com. Research supported by ESF EuroGIGA project ComPoSe as F.R.S.-FNRS - EUROGIGA NR 13604 and by ESF EuroGIGA project GraDR as GAČR GIG/11/E023.

flip graphs [2, 3].

In a pseudo-triangulation, the pseudo-triangles can have linear size. Hence, in contrast to triangulations, the flip operation can no longer be computed in constant time. This fact led to the consideration of pseudo-triangulations in which the size of the pseudo-triangles is bounded by a constant. Kettner et al. [5] showed that every point set admits a pointed pseudo-triangulation with face degree at most four (except, maybe, for the outer face). We call such a pseudo-triangulation a *4-PPT*.

On the one hand, 4-PPTs behave nicely for problems which are hard for general pseudo-triangulations. For instance, they are always properly 3-colorable, while 3-colorability is NP-complete to decide for general pseudo-triangulations [1]. On the other hand, known properties of general pseudo-triangulations remain open for 4-PPTs. For instance, it is not known whether the flip graph of 4-PPTs is connected, even for the basic case of a triangular convex hull.

The aim of this paper is to make a step towards answering this last question, by considering the combinatorial counterpart of 4-PPTs.

A *combinatorial pseudo-triangulation* [6] is a topological embedding of a planar simple graph together with an assignment of labels *reflex/convex* to its angles such that (1) every interior face has exactly three angles labeled convex, (2) all the angles of the outer face are labeled reflex, and (3) no vertex is incident to more than one reflex angle.

Note that this labeling fulfills the same properties as actual reflex/convex angles in a (geometric) pseudo-triangulation. This analogy with the geometric case goes on by calling *pointed* vertices in a combinatorial pseudo-triangulation those which, indeed, are incident to one angle labeled reflex. Then, *combinatorial pointed pseudo-triangulations* are those in which every vertex is pointed. Combinatorial pointed pseudo-triangulations with face degree at most four (except, maybe, for the outer face), will be called *combinatorial 4-PPTs*.

2 Properties

Lemma 1 *Let G be a combinatorial 4-PPT and H be a subgraph of G with $|V(H)| \geq 3$. Then H has at least 3 vertices whose reflex angle is contained in the outer face of H (corners of first type in [6]).*

Corollary 2 *In any combinatorial 4-PPT of the interior of a simple cycle with b vertices, of which c have the reflex angle inside the cycle, the number t of triangular faces is given by $t = b - 2c - 2$.*

A combinatorial pseudo-triangulation has the *generalized Laman property* if every subset of x non-pointed vertices and y pointed vertices, where $x+y \geq 2$, induces a subgraph with at most $3x + 2y - 3$ edges. Both this property and the number of reflex angles from Lemma 1 are related to the stretchability of a combinatorial pseudo-triangulation into a geometric one. A face of a combinatorial pseudo-triangulation is called *degenerate* if it contains edges which appear twice on the boundary of this face.

Proposition 3 *[6, Corollary 2] The following properties are equivalent for a combinatorial pseudo-triangulation G: (1) G can be stretched to become a pseudo-triangulation. (2) G has the generalized Laman property. (3) G has no degenerate faces and every subgraph of G with at least three vertices has at least three corners of first type.*

Since, by definition, combinatorial 4-PPTs have no degenerate faces, we can use Proposition 3 to conclude the following.

Theorem 4 *Every combinatorial 4-PPT can be stretched to become a 4-PPT with the given assignment of angles. Furthermore, combinatorial 4-PPTs have the generalized Laman property.*

Note that there exist non-stretchable combinatorial pointed pseudo-triangulations with faces of size at most 5. See Figure 1. There and in the forthcoming figures, arcs denote angles labeled as reflex.

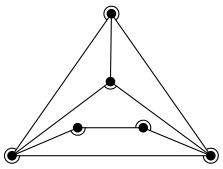


Figure 1: A non-stretchable combinatorial pointed pseudo-triangulation [6].

3 Flips

In the following we focus on combinatorial 4-PPTs with a fixed triangular outer face. For such a combinatorial 4-PPT, Corollary 2 implies that there is only one interior triangular face. Before defining flips between combinatorial 4-PPTs, we make some observations about their geometric counterpart.

Geometric 4-PPTs with triangular convex hull also have only one interior triangle. Furthermore, every edge of the triangle (except for those being part of the convex hull) is flippable [7]. Observe that the removal of the edge e to be flipped merges the triangle and the 4-face adjacent at e into a 5-face, which might be degenerate if the triangle and the 4-face share two edges. See Figure 2. Note that this is the only case in which the triangle and the 4-face can share three vertices, as there are no multiple edges in geometric graphs.

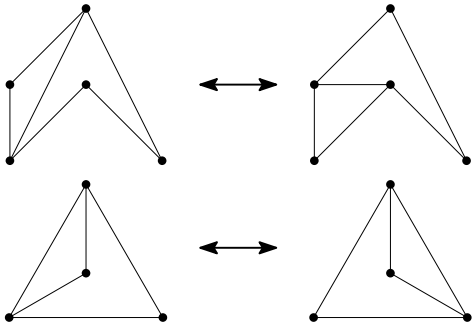


Figure 2: Geometric flip of an edge of a triangle. In the lower case, removal of the flipped edge gives a degenerate 5-face.

Similar to the geometric case, we consider flips of an edge e of the (unique) interior triangular face T in a combinatorial 4-PPT (with triangular outer face): Consider the 4-face F sharing e with T. A *flip* of e consists in replacing e by another edge e' such that (1) e' splits $(T \cup F) \backslash e$ into a triangular face T' and a 4-face F' and (2) the result is a combinatorial 4-PPT. In particular, and in contrast to the geometric case, in the combinatorial setting we have to explicitly avoid multiple edges and thus to ensure that the edge e' we insert is not already contained in the combinatorial 4-PPT (as an edge outside $T \cup F$). The following lemma shows that every interior edge of the interior triangular face can be flipped.

Lemma 5 *In a combinatorial 4-PPT, every edge e of an interior triangular face that is not an edge of the outer face is flippable. Furthermore: (1) If the removal of e results in a degenerate 5-face, then there is a unique valid flip for e. (2) If removing e results in a non-degenerate 5-face, then there are at least two valid flips for e.*

Observe that, given a combinatorial flip between two combinatorial 4-PPTs, by Theorem 4 we know that both of them can be stretched into geometric 4-PPTs with straight edges. However, it might not be possible to use the same geometric embedding for the vertices in both of them.

4 Flip graph connectivity

Lemma 6 *For a given combinatorial 4-PPT with triangular outer face and for any edge b of this outer face, there is a sequence of flips resulting in a combinatorial 4-PPT whose interior triangular face is incident to b.*

Once the interior triangular face is incident to an edge b of the outer face, the next step will be flipping away interior edges incident to one endpoint of b.

Lemma 7 *Given a combinatorial 4-PPT with triangular outer face, in which the interior triangular face T is incident to the edge b of the outer face, there is a sequence of flips resulting in a combinatorial 4-PPT in which the endpoint v of $b = uv$ has no interior incident edges.*

Proof. We describe a flip sequence that flips all inner edges incident to v. This flip sequence can be partitioned into two phases and some cases. Let the vertices neighbored to the vertex v be ordered radially around v, starting with u. In each case, let the vertices in that order be $u = w_0, \ldots, w_k$.

Phase 1: During this phase, the inner triangular face T has uv as a side, i.e., $T = vuw_1$. We distinguish three different cases:

Case 1: vw_1 is the only inner edge incident to v, i.e., $k = 2$. If T is incident to only one 4-face F (i.e., $T \cup F$ is degenerate), we can flip the edge vw_1 and are done. Otherwise, let the 4-face F incident to vw_1 be vw_1sw_2. See Figure 3. The reflex angle inside F is either at s or w_1. If it is at s, we flip vw_1 to w_0s, obtaining the 4-face vw_0sw_2. Otherwise, the reflex angle is at w_1 and we flip vw_1 to w_1w_2, obtaining the 4-face $vw_0w_1w_2$. Either way, the degree of v is 2 and we are done.

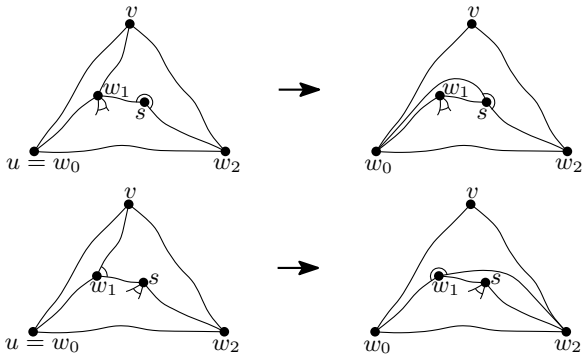


Figure 3: Phase 1, Case 1: Only one interior edge is incident to v.

Case 2: at least two inner edges are incident to v and there does not exist an edge w_0w_2. See Figure 4. Since the reflex angle of v is at the outer face we can replace the edge vw_1 by w_0w_2. This reduces the degree of v by one. The inner triangular face is again adjacent to w_0v, and we remain in Phase 1.

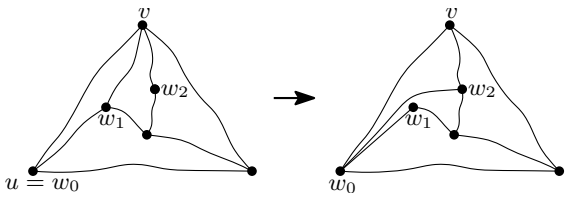


Figure 4: Phase 1, Case 2: Several interior edges are incident to v and w_0w_2 does not exist.

Case 3: at least two inner edges are incident to v and there exists an edge w_0w_2. See Figure 5. If the two inner edges of T are incident to a single 4-face, we have a degenerate case; we flip the edge w_0w_1 to w_1w_2, making vw_1w_2 the inner triangular face. Otherwise, let the 4-face F incident to vw_1 be vw_1sw_2; we flip vw_1 to vs (this is possible since if vs already existed, it would have to cross the cycle uw_2sw_1). Either way, the flip does not reduce the degree of v, but the inner triangular face is now inside the 3-cycle vw_0w_2. We switch to Phase 2.

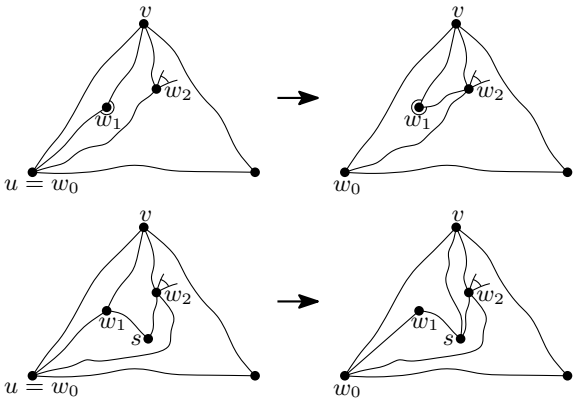


Figure 5: Phase 1, Case 3: The possible transitions to Phase 2.

Phase 2: During this phase, the inner triangular face is vw_1w_2, and w_1 stays fixed for the whole phase. Further, we know that w_1 was enclosed by a 3-cycle (at the transition to this phase), which implies that there are no edges from w_1 to w_i for any $i \geq 2$. We decrease the degree of v in the following manner.
Case 1: there is a 4-face F incident to vw_2. There cannot be an edge w_1w_3 since w_1 was enclosed by a 3-cycle. Further, the reflex angle of F is not at v. Hence, we can flip vw_2 to w_1w_3, which reduces the degree of v and we remain in Phase 2, with vw_1w_3 being the new inner triangular face.
Case 2: there is no 4-face incident to vw_2, i.e., $k = 2$. This case is symmetric to Case 1 of Phase 1. The edge vw_1 is flipped in one of the two described ways, reducing the degree of v to 2 and thus ending the process. □

Theorem 8 *The graph of flips in combinatorial 4-PPTs with n vertices and triangular outer face is connected and has diameter $O(n^2)$.*

Proof. Given such a combinatorial 4-PPT, follow the steps in Lemmas 6 and 7, then use induction for the combinatorial 4-PPT obtained by removing v. This leads to the unique *canonical* combinatorial 4-PPT with triangular outer face, where two of the vertices in the outer face are adjacent to all other vertices, while the third one has degree 2. See Figure 6.

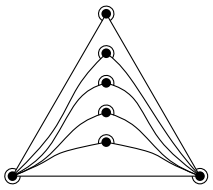


Figure 6: A canonical combinatorial 4-PPT.

Furthermore, the number of flips needed in Lemmas 6 and 7 is at most linear in the number of vertices of the combinatorial 4-PPT. □

The presented basic case of combinatorial 4-PPTs with triangular outer face is extendible to an arbitrarily sized outer face, to labeled vertices, and also to the general case of combinatorial 4-PPTs with an arbitrarily sized outer face on labeled vertices. Elaborating on these extensions would go beyond the scope of this extended abstract, though. Details (and omitted proofs) can be found in a forthcoming full version.

Acknowledgments. This work was initiated during the 9th European Research Week on Geometric Graphs and Pseudo-Triangulations, held May 14–18, 2012 in Alcalá de Henares, Spain. We thank Vincent Pilaud, Pedro Ramos, and André Schulz for helpful comments.

References

[1] O. Aichholzer, F. Aurenhammer, T. Hackl, C. Huemer, A. Pilz, and B. Vogtenhuber. 3-colorability of pseudo-triangulations. In *EuroCG 2010*, pages 21–24, 2010.

[2] O. Aichholzer, F. Aurenhammer, H. Krasser, and P. Braß. Pseudotriangulations from surfaces and a novel type of edge flip. *SIAM J. Comput.*, 32(6):1621–1653, 2003.

[3] S. Bereg. Transforming pseudo-triangulations. *Inf. Process. Lett.*, 90(3):141–145, 2004.

[4] P. Bose and F. Hurtado. Flips in planar graphs. *Comput. Geom.*, 42(1):60–80, 2009.

[5] L. Kettner, D. Kirkpatrick, A. Mantler, J. Snoeyink, B. Speckmann, and F. Takeuchi. Tight degree bounds for pseudo-triangulations of points. *Comput. Geom.*, 25(1–2):3–12, 2003.

[6] D. Orden, F. Santos, B. Servatius, and H. Servatius. Combinatorial pseudo-triangulations. *Discrete Math.*, 307(3–5):554–566, 2007.

[7] G. Rote, F. Santos, and I. Streinu. *Pseudo-triangulations — a survey.* Contemp. Math. AMS, 2008.

Recent developments on the crossing number of the complete graph

Pedro Ramos*

Department of Physics and Mathematics, University of Alcalá, Alcalá de Henares, Spain.

Introduction

There are two well defined periods in the history of the problem of finding the crossing number of the complete graph. Before 2004, the lack of tools to study the problem restricted the developments to the study configurations with a small number of points, the finding of conjectured optimal solutions for arbitrary n, and some fairly basic counting strategies. [4] is an excellent survey for the main results in this period, including the early history of the problem.

In 2004, and independently, Ábrego and Fernández-Merchant [1] and Lovász *et al* [5] discovered a very strong relation between the number of crossings of a rectilinear drawing of the complete graph and another well known object in combinatorial geometry: the number of j-edges of the set of vertices. This lead to a series of improvements on the known lower bounds and, although still only one very basic property of optimal configurations is known (the set of vertices has three extreme points), the gap in the leading term of the known lower and upper bounds has been greatly reduced. A very recent survey of all this developments can be found in [2].

Recently [3] the concept of j-edge has been generalized to topological drawings of the complete graph, and the relation between crossings and j-edges has emerged as a promising tool for the general problem. The conjectured bound has already proven to be optimal for some families of drawings, including 2-page drawings and monotone drawings.

References

[1] B. M. Ábrego and S. Fernández-Merchant. A lower bound for the rectilinear crossing number. *Graphs and Combinatorics*, 21:293–300, 2005.

[2] B. M. Ábrego and S. Fernández-Merchant, and G. Salazar. The rectilinear crossing number of Kn: Closing in (or are we?). In Thirty essays in geometric graph theory, J. Pach (Ed), Springer, XVI, pp 504, 2013.

[3] B.M. Ábrego, O. Aichholzer, S. Fernández-Merchant, P. Ramos, and G. Salazar. The 2-page crossing number of K_n. *Discrete and Computational Geometry*, to appear.

[4] L. Beineke and R. Wilson. The early history of the brick factory problem. *Math. Intelligencer*, 32:41–48, 2010.

[5] L. Lovász, K. Vesztergombi, U. Wagner, and E. Welzl. Convex quadrilaterals and k-sets. In J. Pach, editor, *Contemporary Mathematics Series, 342, AMS 2004*, volume 342, pp. 139–148. American Mathematical Society, 2004.

*Email: pedro.ramos@uah.es. Partially supported by MEC grant MTM2011-22792 and by the ESF EUROCORES programme EuroGIGA, CRP ComPoSe, under grant EUI-EURC-2011-4306.

Index of Authors