technische universität
dortmund

26th European Workshop on Computational Geometry

March 22-24, 2010

Department of Computer Science
Technische Universität Dortmund
Dortmund, Germany

Workshop Proceedings

Jan Vahrenhold (Ed.)

Introduction

This volume contains the abstracts of the talks given at the 26th European Workshop on Computational Geometry (EuroCG'10). The workshop was held at the Technische Universität Dortmund, Germany, on March 22–24, 2010. EuroCG is renowned as a friendly workshop, where researchers across Europe and the world can meet and exchange ideas in a delightful atmosphere.

The workshop received 85 submissions. There was a limited refereeing process to make sure that the presented papers were in scope and met some minimal standards. Since the abstracts were not peer-reviewed in a formal manner, it is expected that many of them will appear in formally reviewed conference proceedings or journals. The reviewing process was synchronized using the EasyChair conference management system, and we thank the EasyChair team for providing this tremendously helpful service to the community.

In addition to the accepted contributions, this volume also contains the abstracts of the invited lectures by Timothy M. Chan (Waterloo), Markus Gross (ETH Zurich), and János Pach (EPFL Lausanne and Rényi Institute Budapest).

We would like to thank all the authors who responded to the call for papers, the invited speakers, the members of the program committee, as well as the external referees and the organizing committee members.

The conference was made possible in part by generous financial support of the "Alumni der Informatik Dortmund e.V.", the Chair for Efficient Algorithms and Complexity Theory and the Chair for Algorithm Engineering at the Faculty of Computer Science, Technische Universität Dortmund.

March 2010 Jan Vahrenhold

Table of Contents

Instance-Optimal Geometric Algorithms*

Timothy M. Chan†

Abstract

Planar convex hull is undisputably one of the most basic problems in computational geometry. Many $O(n \log n)$-time algorithms have been discovered, and this bound is worst-case optimal under a standard algebraic decision tree model. However, some input point sets are "easier" than others. For example, for point sets with output size h, $O(n \log h)$ algorithms are known; for point sets under various distributions, $O(n)$ algorithms are known. In this talk, I will present arguably the ultimate result on planar convex hull: there is an algorithm that provably has running time as good as any other algorithms (within a general class, up to constant factors) on *every* point set. Such an algorithm achieves so-called *instance optimality* (in an order-oblivious sense). Similar instance-optimal results are possible for 3D convex hull and several other fundamental geometric problems, such as 2D and 3D maxima, orthogonal line segment intersection, and planar point location.

The talk will describe an elegant theory that touches on many interesting threads—output-sensitive, adaptive, and average-case algorithms, partition trees, entropy, distribution-sensitive data structures, decision-tree lower bounds, and a new simple adversary argument.

*Joint work with Peyman Afshani and Jeremy Barbay (work appeared in FOCS 2009).

†Department of Computer Science, University of Waterloo
tmchan@uwaterloo.ca

Point Samples for Surface Representation and Geometry Processing

Markus Gross*

Abstract

Over the past decade point primitives have received a growing attention in Computer Graphics and Geometry Processing. There are two main reasons for this new interest in points: On one hand, we have witnessed a dramatic increase in the polygonal complexity of computer graphics models. The overhead of managing, processing, and manipulating very large polygonal meshes has led many researchers to question the future utility of polygons as the fundamental graphics primitive. On the other hand, modern 3D digital photography and 3D scanning systems facilitate the ready acquisition of complex, real-world objects. These techniques generate huge volumes of point samples and create the need for advanced point processing.

In this presentation I will discuss the utility and versatility of point primitives for surface representation and geometric modeling, and I will present a survey the latest research results in this area. I will review novel concepts for the mathematical representation of point-sampled shapes with a focus on moving least squares, spherical MLS, and robust statistics. Furthermore, I will address efficient algorithms for digital geometry processing and modeling of point models, including filtering, resampling, spectral processing, and deformation. In last part, I will discuss how point based representations can help to bridge the gap between numerical simulations and interactive graphics, and I will demonstrate their potential for a fusion of both.

*Department of Computer Science ETH Zurich

Touching points

János Pach*

We say that two simple *curves* in the plane *touch* or are *tangent to* each other if they have precisely one point in common (which is their point of tangency) and at this point one curve does not pass from one side of the other curve to the other. Two *Jordan regions* are said to *touch* if their boundary curves touch. We say that two curves *cross* or properly cross at a point p if p belongs to both of them and in a small neighborhood of p one curve passes from from one side of the other curve to the other.

Estimating the maximum number of tangencies between noncrossing circles was initiated by de Rocquigny [19] at the end of the 19th century. The problem was forgotten for three quarters of a century, until similar questions were asked and answered for "Apollonian arrangements" [10]. It follows immediately from Euler's polyhedral formula that among any $n > 2$ pairwise disjoint simply connected Jordan regions in the plane there are at most $3n - 6$ "touching" (tangent) pairs. This bound is tight. A family of closed curves in the plane is said to form a set of *pseudo-circles* if any two of them are disjoint, or tangent to each other in one point, or cross in precisely two points.

Theorem 1 (Erdős-Grünbaum) *Any set of $n > 2$ pairwise noncrossing pseudo-circles in the plane determines at most $3n - 6$ points of tangencies. This bound is tight.*

Note that at each of these "touching points" (points of tangencies) several curves may touch one another.

Erdős's famous unsolved question [6] on the maximum number of unit distance pairs among n points in the plane can also be formulated as a problem about tangencies: What is the maximum number $u(n)$ of tangencies among n (possibly overlapping) disks of unit diameter in the plane? The answer is superlinear in n.

Theorem 2 (Erdős, Spencer-Szemerédi-Trotter [21]) *The maximum number of tangencies among n unit circles in the plane satisfies*

$$n^{1+c/\log\log n} < u(n) < c' n^{4/3},$$

for suitable constants $c, c' > 0$.

Equivalently, one can ask: What is the maximum number of incidences between n unit circles and n points in the plane?

It was first observed by Tamaki and Tokuyama [22] that in order to obtain an upper bound on the number of incidences between a family $\mathcal{C}$ of curves and a set of points, it is sufficient to estimate the minimum number of points needed to cut the curves in $\mathcal{C}$ into "pseudo-segments," that is, smaller pieces such that any pair of them are either disjoint or cross precisely once. Obviously, this number is at least as large as the number of tangencies between the members of $\mathcal{C}$, and in most cases these two quantities do not differ too much. For many applications, this approach leads to the best known upper bounds for the number of incidences between curves and points [1], [5], [12].

Theorem 3 (Marcus-Tardos) *Any family of n pseudo-circles can be cut into $O(n^{3/2}\log n)$ pseudo-segments.*

A natural generalization of the notion of tangency among closed curves is that of a "lens," i.e., a face of the arrangement bounded by precisely two arcs belonging to different curves. In his thesis, Rom Pinchasi [16], in connection with a conjecture of Bezdek [4], proved the following remarkable result: Any family of n pairwise intersecting circles in the plane determines at most n lenses. This result was extended to pseudo-circles [1] (see also [2]).

Theorem 4 (Agarwal et al.) *Any family $\mathcal{C}$ of pairwise intersecting pseudo-circles, no three of which pass through the same point, determine at most $O(n)$ tangencies.*

As is shown by Theorem 2, this statement does not remain true if we drop the condition that the curves are pairwise intersecting. However, if we count only those tangencies that do not belong to the interior of any member of $\mathcal{C}$, then we can again obtain a linear upper bound [11], [15]. If we also drop the condition that the curves are pseudo-circles, then even the number of tangencies not contained in the interior of a third curve can be as large as $\Omega(n^{4/3})$. It was proved in [8] that this bound is not far from being optimal, provided that no pair of curves is allowed to cross in more than a fixed number s of times.

If we do not assume that any two curves cross a bounded number of times, then it is easy to construct

*EPFL, Lausanne and Rényi Inst., Budapest pach@cims.nyu.edu

a family of n simple closed curves, no *three* of which pass through the same point, with a quadratic number of touching pairs. In fact, there are two families of size $n/2$ such that each member of the first family is tangent to every member of the second. However, in this case, the total number of crossings between the curves must be quite large.

Theorem 5 [FFPP10] *Let $\mathcal{A}$ and $\mathcal{B}$ be two families of x-monotone curves with a total of n members, no three of which pass through the same point, such that no curve in $\mathcal{A}$ properly crosses any curve in $\mathcal{B}$. If m denotes the number of pairs of touching curves (α, β) with $\alpha \in \mathcal{A}$ and $\beta \in \mathcal{B}$, the total number of crossing points in $\mathcal{A} \cup \mathcal{B}$ is $\Omega(m \log m)$.*

Consequently, if $m > \varepsilon n^2$ for some $\varepsilon > 0$, then the total number of crossing points in $\mathcal{A} \cup \mathcal{B}$ is superquadratic in n.

This result seems to support the following conjecture (see also [13], [20]).

Conjecture 1 (Richter-Thomassen [18]) *Any intersecting family of n closed curves, no three of which pass through the same point, determines a total of at least $(1 - o(1))n^2$ intersection points.*

In a forthcoming paper [14], we study the structure of tangencies between two families of closed Jordan regions, each consisting of n pairwise disjoint members. It was shown by Pinchasi and Ben-Dan [3], using the proof idea of Theorem 3 that the maximum number of such tangencies is $O(n^{3/2} \log n)$.

Theorem 6 [PST10] *The number of tangencies between two families of convex bodies in the plane, each consisting of $n > 2$ disjoint members, cannot exceed $6n - O(1)$. This bound is asymptotically tight.*

Corollary 7 [PST10] *Let $\mathcal{C}$ be a family of n convex bodies in the plane, which can be decomposed into k subfamilies consisting of disjoint bodies. The total number of tangencies between members of $\mathcal{C}$ is $O(kn)$. This bound is asymptotically tight.*

Conjecture 2 [PST10] *For every fixed integer $k > 2$, the number of tangencies in any n-member family of convex bodies, no k of which are pairwise intersecting, is at most $O_k(n)$.*

Somewhat surprisingly, the above results are no longer true if we replace the assumption that the sets are convex by the weaker one that they are *vertically convex*, that is, they are closed connected sets and every vertical line misses them or intersects them in a connected set (interval).

Theorem 8 [PST10] *Let $f(n)$ denote the maximum number of tangencies between two n-member families of disjoint, vertically convex bodies in the plane. Then we have*

$$\Omega(n \log n) \leq f(n) \leq O(n \log^2 n).$$

Acknowledgment

Research supported by Grants from NSF, NSA, PSC-CUNY, BSF, OTKA, and SNF.

References

[1] P.K. Agarwal, E. Nevo, J. Pach, R. Pinchasi, M. Sharir, and S. Smorodinsky. Lenses in arrangements of pseudo-circles and their applications. *J. ACM* **51** (2004), 139–186.

[2] N. Alon, H. Last, R. Pinchasi, and M. Sharir. On the complexity of arrangements of circles in the plane. *Discrete Comput. Geom.* **26** (2001), 465–492.

[3] I. Ben-Dan and R. Pinchasi. Personal communication, November 2007.

[4] A. Bezdek. Incidence problems for points and unit circles. In: *Paul Erdős and His Mathematics* (A. Sali, M. Simonovits and V. T. Sós, Eds.), J. Bolyai Math. Soc., Budapest, 1999, 33–36.

[5] T.M. Chan. On levels in arrangements of curves. II. A simple inequality and its consequences. *Discrete Comput. Geom.* **34** (2005), no. 1, 11–24.

[6] P. Erdős. On sets of distances of n points. *Amer. Math. Monthly* **53** (1946), 248–250.

[7] P. Erdős and B. Grünbaum. Osculation vertices in arrangements of curves. *Geometriae Dedicata* **1** (1973), 322–333.

[8] E. Ezra, J. Pach, and M. Sharir. On regular vertices of the union of planar convex objects. *Discrete Comput. Geom.* **41** (2009), 216–231.

[9] J. Fox, F. Frati, J. Pach, and R. Pinchasi. Crossings between curves with many tangencies. In: *Proc. WALCOM: Workshop on Algorithms and Computation, Lecture Notes in Computer Science,* Springer-Verlag, to appear.

[10] B. Grünbaum. *Arrangements and Spreads. Regional Conference Series in Mathematics, No.* **10**, Am. Math. Soc., Providence (1972).

[11] K. Kedem, R. Livne, J. Pach, and M. Sharir. On the union of Jordan regions and collision-free translational motion amidst polygonal obstacles. *Discrete Comput. Geom.* **1** (1986), 59–71.

[12] A. Marcus and G. Tardos. Intersection reverse sequences and geometric applications. *J. Combin. Theory Ser. A* **113** (2006), 675–691.

[13] D. Mubayi. Intersecting curves in the plane *Graphs Combin.* **18** (2002), no. 3, 583–589.

[14] J. Pach, A. Suk, and M. Treml. *Tangencies between families of disjoint regions in the plane.* Unpublished manuscript.

[15] J. Pach and M. Sharir. On the boundary of the union of planar convex sets. *Discrete Comput. Geom.* **21** (1999), no. 3, 321–328.

[16] R. Pinchasi. *Problems in Combinatorial Geometry in the Plane.* Ph.D. Thesis, Hebrew University, Jerusalem, 2001.

[17] R. Pinchasi and R. Radoičić. On the number of edges in geometric graphs with no self-intersecting cycle of length 4. In: *Towards a Theory of Geometric Graphs, Contemporary Mathematics* (J. Pach, ed.), **342**, American Mathematical Society, Providence, RI, 2004.

[18] R. B. Richter and C. Thomassen. Intersection of curves systems and the crossing number of $C_5 \times C_5$, *Discrete Comput. Geometry* **13** (1995), 149–159.

[19] G. de Rocquigny. Questions 1179 et 1180. *Intermed. Math.* **4** (1897), 267 and **15** (1908), 169.

[20] G. Salazar. On the intersections of systems of curves. *J. Combin. Theory Ser. B* **75** (1999), 56–60.

[21] J. Spencer, E. Szemerédi, and W. Trotter, Jr. Unit distances in the Euclidean plane. In: *Graph Theory and Combinatorics (Cambridge, 1983)*, Academic Press, London, 1984, 293–303.

[22] H. Tamaki, T. Tokuyama. How to cut pseudoparabolas into segments. *Discrete Comput. Geom.* **19** (1998), no. 2, 265–290.

Proximity Graphs inside Large Weighted Graphs

Bernardo M. Ábrego* Ruy Fabila-Monroy† Silvia Fernández-Merchant* David Flores-Peñaloza‡
Ferran Hurtado§ Henk Meijer¶ Vera Sacristán§ Maria Saumell§

Abstract

Given a large weighted graph $G = (V, E)$ and a subset U of V, we define several graphs with vertex set U in which two vertices are adjacent if they satisfy some prescribed proximity rule. These rules use the shortest path distance in G and generalize the proximity rules that generate some of the most common proximity graphs in Euclidean spaces. We prove basic properties of the defined graphs and provide algorithms for their computation.

1 Introduction

In Euclidean spaces, proximity graphs are a key tool to obtain neighborhood relations in a given set of points [5]. They have been intensively explored in the contexts of spacial distribution analysis [9] and graph drawing [7], among others.

In non-Euclidean settings, the Delaunay graph and its relatives have found applications in the analysis of networks that model real connection nets. A prominent example is the network Voronoi diagram (see Section 3.8 in [9]).

Here we deal with a complex graph G with a large number of vertices and edges, in which it is difficult to distinguish which are the relations of proximity among a subset of the vertices. The edges of the graph come with an associated positive weight. We study relations of proximity based on shortest paths along $G = (V, E)$ among the vertices of a subset $U \subseteq V$, which might represent the schools in the map of a city, the corresponding stations in a huge transportation net, etc. We consider generalizations of some well-known proximity graphs. This appears to be a natural method to provide notions of closeness.

*Department of Mathematics, California State University, Northridge, CA, {bernardo.abrego,silvia.fernandez}@csun.edu.

†Departamento de Matemáticas, CINVESTAV, Mexico DF, Mexico, ruyfabila@math.cinvestav.edu.mx.

‡Instituto de Matemáticas, Universidad Nacional Autónoma de México, dflores@math.unam.mx.

§Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, Barcelona, Spain, {ferran.hurtado,vera.sacristan,maria.saumell}@upc.edu.
Partially supported by projects MTM2009-07242 and Gen. Cat. DGR 2009SGR1040.
¶Science Department, Roosevelt Academy, Middelburg, The Netherlands, h.meijer@roac.nl.

The natural and important question of defining suitable notions of closeness among vertices of a graph has found different kinds of answers in the literature. However, we are only aware of one approach that uses proximity graphs (see [6, 11]). The graphs considered there are clearly different from ours, as proximity is constructed by adopting a notion whose universe is a given geometric graph, but where the relations are given by the full Euclidean plane.

Let us mention that the set U together with the shortest-path distance constitutes a finite metric space, so some of the proximity graphs we consider are not new because they can be seen as a particular case of proximity graphs defined on general metric spaces. Even though there exists some literature on proximity graphs in metric spaces, to the best of our knowledge this topic has not been deeply investigated, as only some definitions and basic properties have been given (see Section 4.5 in [12], and also [4]). The sphere-of-influence graph has been further studied [3, 8], but it is out of the scope of our work.

When using empty regions as proximity criteria in G, such as disks, two main variations arise, since we might allow these disks to be centered at any point in G, or we might restrict their centers to lie only on vertices of the graph, as in [3, 1]. Moreover, the definition of certain regions of interference might depend on the multiplicity of paths or distances in G. Degeneracies that occur in the standard geometric case also generate several possibilities. For the sake of clarity we first present the situation where there are essentially no degeneracies (Sections 2–5). In Section 6 we drop the non-degeneracy assumptions and extend our results to the general setting.

Proofs and descriptions of the algorithms will be given in the full-version of this paper.

2 Definitions and Notation

We deal with a connected and edge-weighted graph $G = (V, U, E)$, where $U \subseteq V$ and all edges have positive real weights assigned to them. We assume that it is possible to consider points in the edges of G; more precisely, for every edge $e = (v_1, v_2)$ with weight $w(e)$ and every $r \in (0, w(e))$, we assume that there exists a point p in e and paths from both v_1 and v_2 to p such that the weight of the path from v_1 to p is r,

and the weight of the path from v_2 to p is $w(e) - r$ (if G is embedded in the plane, these paths are simply portions of the edges). We say that p is a *point of* G if p is either a vertex of G, or a point in an edge of G. The *distance* $d_G(p, q)$ between two points p and q in G is defined as the minimum total weight of any path connecting p and q in G. The *closed disk* $D_G(p, r)$ is defined as the set of points q of G for which $d_G(p, q) \leq r$. We say that $u_i \in U$ is a *nearest neighbor* of $u_j \in U$ with $i \neq j$ if $d_G(u_j, u_i) \leq d_G(u_j, u_k)$ for all vertices $u_k \neq u_j, u_i \in U$. A *midpoint* of two points p and q of G is a point m on one of the shortest paths from p to q such that $d_G(m, p) = d_G(m, q)$. We denote the set of midpoints of p and q by $M_G(p, q)$. For the remainder of this paper, we define $|V| = m$, $|U| = n$, and $|E| = e$.

We first consider the case where the following non-degeneracy assumptions hold: (A1) for all $u_i, u_j \in U$, the shortest path connecting u_i and u_j is unique; (A2) there do not exist three distinct vertices $u_i, u_j \in U$, $v \in V - U$ such that $d_G(v, u_i) = d_G(v, u_j)$; (A3) there do not exist vertices $v_i, v_j \in V$, $u_i, u_j \in U$ such that $d_G(v_i, u_i) = d_G(v_j, u_j)$ with $v_i \neq u_i$; (A4) all paths in G between distinct nodes in V have different lengths.

Obviously, the previous assumptions are not independent, but considering them separately allows to clarify and provide a more precise description of the scenario. In Section 6, we extend the results from Sections 3–5 to the general case where A1–A4 are not necessarily satisfied.

We now adapt several known definitions to proximity structures in graphs $G = (V, U, E)$.

Definition 1 *The* nearest neighbor graph *of* $G = (V, U, E)$, *denoted by* $\mathrm{NNG}(G)$, *is the graph* $H = (U, F)$ *such that* $(u_i, u_j) \in F$ *if* u_j *is one of the nearest neighbors of* u_i *in* G.

Definition 2 *A* minimal spanning tree *of* $G = (V, U, E)$ *is a tree* $T = (U, F)$ *such that the sum of* $d_G(u_i, u_j)$ *over all edges* $(u_i, u_j) \in F$ *is minimal. The* union of the minimal spanning trees *of* G, *denoted by* $\mathrm{UMST}(G)$, *is the graph consisting of all the edges included in any of the minimal spanning trees of* G.

If A3 holds, each vertex in U has exactly one nearest neighbor and the minimal spanning tree of G, denoted by $\mathrm{MST}(G)$, is unique.

Definition 3 *The* relative neighborhood graph *of* $G = (V, U, E)$, *denoted by* $\mathrm{RNG}(G)$, *is the graph* $H = (U, F)$ *such that* $(u_i, u_j) \in F$ *if there exists no vertex* $u_k \in U$ *such that* $d_G(u_k, u_i) < d_G(u_i, u_j)$ *and* $d_G(u_k, u_j) < d_G(u_i, u_j)$.

Definition 4 *The* free Gabriel graph *of* $G = (V, U, E)$, *denoted by* $\mathrm{GG_f}(G)$, *is the graph* $H = (U, F)$ *such that* $(u_i, u_j) \in F$ *if there exists no vertex* $u_k \in U$ $(u_k \neq u_i, u_j)$ *such that* $d_G(p, u_k) \leq d_G(p, u_i)$, *where* p *is the midpoint of* u_i *and* u_j.

If A1 holds, there exists only one midpoint of u_i and u_j, thus the previous graph is well-defined.

Definition 5 *The* constrained Gabriel graph *of* $G = (V, U, E)$, *denoted by* $\mathrm{GG_c}(G)$, *is the graph* $H = (U, F)$ *such that* $(u_i, u_j) \in F$ *if the smallest closed disk centered at a vertex in* V *enclosing* u_i *and* u_j *does not contain any other vertex from* U.

The previous graph is well-defined if A3 holds.

Definition 6 *The* Voronoi region *of a vertex* $u_i \in U$ *is the set of points* p *of* G *such that* $d_G(p, u_i) \leq d_G(p, u_j)$ *for all vertices* $u_j \in U$ *different from* u_i. *The* Voronoi diagram *of* $G = (V, U, E)$, *denoted by* $\mathrm{VD}(G)$, *is the Voronoi diagram of the vertex set* U *for the distance* d_G.

Definition 7 *The* free Delaunay graph *of* $G = (V, U, E)$, *denoted by* $\mathrm{DG_f}(G)$, *is the graph* $H = (U, F)$ *such that* $(u_i, u_j) \in F$ *if there exists a closed disk* $D_G(p, r)$, *where* p *is a point of* G, *enclosing* u_i *and* u_j *and no other vertex from* U.

Definition 8 *The* constrained Delaunay graph *of* $G = (V, U, E)$, *denoted by* $\mathrm{DG_c}(G)$, *is the graph* $H = (U, F)$ *such that* $(u_i, u_j) \in F$ *if there exists a closed disk* $D_G(v, r)$, *with* $v \in V$, *enclosing* u_i *and* u_j *and no other vertex from* U.

3 Inclusion Sequence

The graphs just defined satisfy some inclusion relations. In this section we show which proximity graphs are subgraphs of which other proximity graphs assuming A1, A2, and A3.

Theorem 1 *The relations of containment among all classes of proximity graphs are shown in Table 1. The symbol* $\subseteq$ *means that the inclusion is satisfied for all graphs* G, *and* $\not\subseteq$ *means that there are graphs* G *for which the inclusion is not satisfied.*

All inclusions in the table are proper, in the sense that there exists a graph G for which the corresponding proximity subgraph does not coincide with its supergraph.

4 Geometric and Combinatorial Properties

We define the *dual graph of the Voronoi diagram* of $G = (V, U, E)$ as the graph with vertex set U and edges connecting two vertices if their Voronoi regions share some point in G that does not belong to the Voronoi region of any other element in U.

Table 1: Relations of containment among proximity graphs in the non-degenerate case.

	MST	RNG	GG_c	GG_f	DG_c	DG_f
NNG	$\subseteq$	$\subseteq$	$\not\subseteq$	$\subseteq$	$\subseteq$	$\subseteq$
MST		$\subseteq$	$\not\subseteq$	$\subseteq$	$\not\subseteq$	$\subseteq$
RNG			$\not\subseteq$	$\subseteq$	$\not\subseteq$	$\subseteq$
GG_c				$\not\subseteq$	$\subseteq$	$\subseteq$
GG_f					$\not\subseteq$	$\subseteq$
DG_c						$\subseteq$

Table 2: Running times of the algorithms to compute the proximity graphs on G.

proximity graph	running time
NNG	$O(e + (m-n)\log(m-n))$
MST	$O(e\,\alpha(e,n) + (m-n)\log(m-n))$
RNG	$O(\mathrm{APSP}(G) + \min\{n^2, e\}n)$
GG_c	$O(\mathrm{APSP}(G) + \min\{n^2, e\}m)$
GG_f	$O\left(\mathrm{APSP}(G) + \min\{n^2, e\}m\right)$
DG_c	$O(e + m\log m)$
DG_f	$O(e + (m-n)\log(m-n))$

Proposition 2 *Let $G = (V, U, E)$ be a graph. Then $DG_f(G)$ is the dual graph of $VD(G)$.*

The previous proposition allows to draw the first analogy between the usual proximity graphs and these new proximity structures on graphs. Moreover, it is a key tool to prove the following result:

Corollary 3 *Let $G = (V, U, E)$ be a graph. The number of edges of $NNG(G)$, $MST(G)$, $RNG(G)$, $GG_c(G)$, $GG_f(G)$, $DG_c(G)$, and $DG_f(G)$ is at most e.*

This bound is tight up to a constant factor:

Proposition 4 *There exists a graph $G = (V, U, E)$ such that $RNG(G) = GG_f(G) = DG_f(G) = G$. There also exists a graph $G' = (V', U', E')$ such that the number of edges of $GG_c(G')$ and $DG_c(G')$ is $e'/2$. Furthermore, all of these graphs have $\Theta(n^2)$ edges.*

In the following theorems we show that the proximity graphs inherit planarity and acyclicity from the original graph.

Theorem 5 *Let $G = (V, U, E)$ be a planar graph. Then $NNG(G)$, $MST(G)$, $RNG(G)$, $GG_c(G)$, $GG_f(G)$, $DG_c(G)$, and $DG_f(G)$ are planar.*

Theorem 6 *Let $G = (V, U, E)$ be a tree. Then $GG_c(G)$ and $DG_c(G)$ are forests, and $RNG(G) = GG_f(G) = DG_f(G) = MST(G)$.*

Next we give complete characterizations for those graphs that are isomorphic to a certain proximity graph of some other graph.

Proposition 7 *If $G = (V, E)$ is a graph, there exists a graph $\bar{G} = (\bar{V}, \bar{U}, \bar{E})$ such that $G \cong NNG(\bar{G})$ if and only if G is acyclic and does not contain isolated vertices.*

Proposition 8 *If $G = (V, E)$ is a graph, there exists a graph $\bar{G} = (\bar{V}, \bar{U}, \bar{E})$ such that $G \cong MST(\bar{G})$ if and only if G is a tree.*

Proposition 9 *If $G = (V, E)$ is a graph, there exists a graph $\bar{G} = (\bar{V}, \bar{U}, \bar{E})$ such that $G \cong RNG(\bar{G})$ if and only if G is triangle-free.*

Proposition 10 *Let $G = (V, E)$ be a graph. There exists a graph $\bar{G} = (\bar{V}, \bar{U}, \bar{E})$ such that $G \cong GG_c(\bar{G}) = GG_f(\bar{G}) = DG_c(\bar{G}) = DG_f(\bar{G})$.*

5 Algorithms

We have derived algorithms to compute each of the proximity graphs we have studied. Due to lack of space, we omit the description of the algorithms and only give their running times.

In some cases the algorithm computes the shortest paths between all pairs of vertices in U. If G is a sparse graph, we use the algorithm in [10], which runs in $O(m \log m + ne \log \alpha(m, e))$ time. If G is dense, we use the algorithm in [2], which runs in $O\left(m^3 \log^3 \log m / \log^2 m\right)$ time. We define $\mathrm{APSP}(G) = \min\{m \log m + ne \log \alpha(m, e), m^3 \log^3 \log m / \log^2 m\}$.

Theorem 11 *For each graph $G = (V, U, E)$, the proximity graphs on G can be computed in the number of steps indicated in Table 2.*

6 Presence of Degeneracies

In this section we generalize our results to the case in which degeneracies arise.

First of all, we look through the definitions. The graphs $NNG(G)$, $UMST(G)$, $RNG(G)$, $DG_f(G)$, and $DG_c(G)$ are well-defined regardless of the properties of G, although, in contrast to the non-degenerate case, a vertex in U might have several nearest neighbors.

In the general case there might be more than one shortest path between two vertices of U. This gives rise to two definitions of free Gabriel graphs:

Definition 9 *The* free-one Gabriel graph *of $G = (V, U, E)$, denoted by $GG_{f1}(G)$, is the graph $H = (U, F)$ such that $(u_i, u_j) \in F$ if there exists $p \in$*

Table 3: Relations of containment among all classes of proximity graphs in the general case.

	UMST	RNG	GG_{ca}	GG_{c1}	GG_{fa}	GG_{f1}	DG_c	DG_f
NNG	$\subseteq$	$\subseteq$	$\not\subseteq$	$\not\subseteq$	$\not\subseteq$	$\not\subseteq$	$\not\subseteq$	$\not\subseteq$
UMST		$\subseteq$	$\not\subseteq$	$\not\subseteq$	$\not\subseteq$	$\not\subseteq$	$\not\subseteq$	$\not\subseteq$
RNG			$\not\subseteq$	$\not\subseteq$	$\not\subseteq$	$\not\subseteq$	$\not\subseteq$	$\not\subseteq$
GG_{ca}				$\subseteq$	$\not\subseteq$	$\not\subseteq$	$\subseteq$	$\subseteq$
GG_{c1}					$\not\subseteq$	$\not\subseteq$	$\subseteq$	$\subseteq$
GG_{fa}						$\subseteq$	$\not\subseteq$	$\subseteq$
GG_{f1}							$\not\subseteq$	$\subseteq$
DG_c								$\subseteq$

$M_G(u_i, u_j)$ such that no vertex $u_k \in U$ ($u_k \neq u_i, u_j$) satisfies $d_G(p, u_k) \leq d_G(p, u_i)$.

Definition 10 *The* free-all Gabriel graph *of $G = (V, U, E)$, denoted by $GG_{fa}(G)$, is the graph $H = (U, F)$ such that $(u_i, u_j) \in F$ if, for each $p \in M_G(u_i, u_j)$, no vertex $u_k \in U$ ($u_k \neq u_i, u_j$) satisfies $d_G(p, u_k) \leq d_G(p, u_i)$.*

Analogously, the definition of the constrained Gabriel graph must be replaced by the following variants:

Definition 11 *The* constrained-one Gabriel graph *of $G = (V, U, E)$, denoted by $GG_{c1}(G)$, is the graph $H = (U, F)$ such that $(u_i, u_j) \in F$ if there exists a closed disk $D_G(v, r)$, with $v \in V$ and $r = \min_{v \in V}\{r \mid D_G(v, r)$ contains both u_i and $u_j\}$, enclosing u_i and u_j and no other vertex from U.*

Definition 12 *The* constrained-all Gabriel graph *of $G = (V, U, E)$, denoted by $GG_{ca}(G)$, is the graph $H = (U, F)$ such that $(u_i, u_j) \in F$ if every closed disk $D_G(v, r)$ containing both u_i and u_j, and where $v \in V$ and $r = \min_{v \in V}\{r \mid D_G(v, r)$ contains both u_i and $u_j\}$, does not contain any other vertex of U.*

Now we may go through the inclusion relations of the proximity graphs.

Theorem 12 *If degenerate situations are allowed, the relations of containment among all classes of proximity graphs are shown in Table 3. Furthermore, all classes of proximity graphs are different.*

To conclude this section, we focus on the most important properties presented in Section 3.

The fact that $DG_f(G)$ is the dual graph of the Voronoi diagram of G holds in all cases. On the other hand, if A2 is not satisfied, some of the proximity graphs might have more edges than the original graph:

Theorem 13 *Let $G = (V, U, E)$ be a graph. The number of edges of $GG_{ca}(G)$, $GG_c(G)$, $GG_{fa}(G)$, $GG_f(G)$, $DG_c(G)$, and $DG_f(G)$ is at most e. The number of edges of NNG(G), UMST(G), and RNG(G) may be greater than e.*

Finally, we check whether all proximity graphs inherit the property of being planar or acyclic in the degenerate case.

Theorem 14 *Let $G = (V, U, E)$ be a planar graph. Then the graphs $GG_{ca}(G)$, $GG_{c1}(G)$, $GG_{fa}(G)$, $GG_{f1}(G)$, $DG_c(G)$, and $DG_f(G)$ are planar, whereas NNG(G), UMST(G), and RNG(G) may not be.*

Theorem 15 *Let $G = (V, U, E)$ be a tree. Then the graphs $GG_{ca}(G)$, $GG_{c1}(G)$, $GG_{fa}(G)$, $GG_{f1}(G)$, $DG_c(G)$, and $DG_f(G)$ are acyclic, whereas NNG(G), UMST(G), and RNG(G) may not be.*

The algorithms in the preceding section can be adapted to run under the presence of degeneracies yet we omit here further details.

References

[1] M. Abellanas and F. Harary. *Delaunay Graphs on a Prescribed Graph.* Proc. EuroCG'99, pp. 101–103, 1999.

[2] T.M. Chan. *More Algorithms for All-Pairs Shortest Paths in Weighted Graphs.* Proc. STOC'07, pp. 590–598, 2007.

[3] F. Harary, M.S. Jacobson, M.J. Lipman, and F.R. McMorris. *Abstract Sphere-of-Influence Graphs.* Math. Comput. Modelling 17(11):77–83, 1993.

[4] J.W. Jaromczyk and M. Kowaluk. *A Note on Relative Neighborhood Graphs.* Proc. SoCG'87, pp. 233–241, 1987.

[5] J.W. Jaromczyk and G.T. Toussaint. *Relative Neighborhood Graphs and Their Relatives.* Proc. IEEE 80(9):1502–1517, 1992.

[6] S. Kapoor and X.-Y. Li. *Proximity Structures for Geometric Graphs.* Proc. WADS'03, pp. 365–376, 2003.

[7] G. Liotta. *Proximity Drawings.* In: Handbook of Graph Drawing and Visualization. CRC Press, to appear.

[8] T.S. Michael and T. Quint. *Sphere of Influence Graphs in General Metric Spaces.* Math. Comput. Modelling 29(7):45–53, 1999.

[9] A. Okabe, B. Boots, K. Sugihara, and S.N. Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams.* John Wiley & Sons, 2000.

[10] S. Pettie and V. Ramachandran. *A Shortest Path Algorithm for Real-Weighted Undirected Graphs.* SIAM J. Comput. 34(6):1398–1431, 2005.

[11] R. Pinchasi and S. Smorodinsky. *On Locally Delaunay Geometric Graphs.* Proc. SoCG'04, pp. 378–382, 2004.

[12] H. Samet. *Foundations of Multidimensional and Metric Data Structures.* Morgan Kaufmann, 2006.

Computing the discrete Fréchet distance with imprecise input*

Hee-Kap Ahn † Christian Knauer ‡ Marc Scherfenberg‡ Lena Schlipf‡ Antoine Vigneron §

Abstract

We consider the problem of computing the discrete Fréchet distance between two polygonal curves when their vertices are imprecise. An imprecise point is given by a region and this point could lie anywhere within this region. By modelling imprecise points as balls in arbitrary fixed dimension, we present an algorithm for this problem that returns in time $O(m^2n^2\log^2(mn))$ the Fréchet distance lower bound between two imprecise polygonal curves with n and m vertices, respectively. We give an improved algorithm for the planar case with running time $O(mn\log^2(mn)+(m^2+n^2)\log mn)$. In the d-dimensional orthogonal case, where points are modelled as axis-parallel boxes, and we use the L_∞ distance, we give an $O(dmn\log(dmn))$-time algorithm.

1 Introduction

Shape matching is an important ingredient in a wide range of computer applications such as computer vision, computer–aided design, robotics, medical imaging, and drug design. In shape matching, we are given two geometric objects and we compute their distance according to some geometric similarity measure. The Fréchet distance [1] is a natural distance function for continuous shapes such as curves and surfaces, and is defined using reparameterizations of the shapes.

The discrete Fréchet distance is a variant of the Fréchet distance in which we only consider vertices of polygonal curves. Given two polygonal curves with n and m vertices, respectively, there is a dynamic programming algorithm that computes the discrete Fréchet distance between them in $\Theta(mn)$ time [6], and no subquadratic algorithm is known yet.

*Work by Ahn was supported by the Korea Research Foundation Grant funded by the Korean Government(KRF-2008-614-D00008). Work by Scherfenberg was supported by the Deutsche Forschungsgemeinschaft, grant AL 253/5-1. Work by Schlipf was supported by the Deutsche Forschungsgemeinschaft within the research training group 'Methods for Discrete Structures'(GRK 1408).

†Department of Computer Science and Engineering, POSTECH, Pohang, Korea. heekap@postech.ac.kr

‡Institut für Informatik, Freie Universität Berlin, Germany, {knauer,scherfen,schlipf}@mi.fu-berlin.de

§INRA, UR 341 Mathématiques et Informatique Appliquées, 78352 Jouy-en-Josas, France. antoine.vigneron@jouy.inra.fr

Most of previous works on the Fréchet distance assume that the input curves are given precisely. The input curve, however, could be only an approximation; In many cases, geometric data comes from measurements of continuous real-world phenomenons, and the measuring devices have finite precision.

Imprecise data can be modelled in different ways. One possible model, for data that consists of points, is to assign each point to a region, typically a disk or a square. In this case, existing algorithms for computing the Fréchet distance could be too sensitive to the precision of the measurements, and they may return a solution without providing any guarantee on its correctness or preciseness. One solution to this problem is to take the impreciseness of the input into account in the design of algorithms, so that they return a solution with some additional information on its quality.

Our results. We study the problem of computing the discrete Fréchet distance between two polygonal curves, where the vertices of a polygonal curve are imprecise. Each point belongs to a region, which is either a Euclidean ball or an axis-parallel box in $\mathbb{R}^d$. We consider two cases: the orthogonal case and the Euclidean case. In the orthogonal case, the regions are boxes, and we use the L_∞ distance. In the Euclidean case, the regions are balls and we use the Euclidean distance.

Given two imprecise sequences of n and m points, respectively, we give algorithms for computing the Fréchet distance lower bound between these two sequences. In the orthogonal case, our algorithm runs in $O(dmn\log dmn)$ time. In the Euclidean case, we give an $O(m^2n^2\log^2 mn)$-time algorithm for arbitrary fixed dimension. We also give an improved algorithm for the planar Euclidean case with running time $O(mn\log^2(mn)+(m^2+n^2)\log mn)$.

2 Notation and preliminaries

We work in $\mathbb{R}^d$, and we use a metric $\mathrm{dist}(\cdot,\cdot)$ which is either the Euclidean distance, or the L_∞ distance. Let $A=a_1,\ldots,a_n$ and $B=b_1,\ldots,b_m$ denote two sequences of points in $\mathbb{R}^d$. A *coupling* is a sequence of ordered pairs $(\alpha_1,\beta_1),\ldots,(\alpha_c,\beta_c)$ such that:

- $\alpha_1=1$, $\beta_1=1$, $\alpha_c=n$ and $\beta_c=m$.

- for each $1 \leqslant k < c$, one of the three statements below is true:
  - $\alpha_{k+1} = \alpha_k + 1$ and $\beta_{k+1} = \beta_k + 1$.
  - $\alpha_{k+1} = \alpha_k + 1$ and $\beta_{k+1} = \beta_k$.
  - $\beta_{k+1} = \beta_k + 1$ and $\alpha_{k+1} = \alpha_k$

The *discrete Fréchet distance* $\mathrm{F}(A, B)$ is the minimum, over all couplings, of $\max_{1 \leqslant k \leqslant c} \mathrm{dist}(a_{\alpha_k}, b_{\beta_k})$.

In what follows, we consider the case where the two point sequences A and B are *imprecise*. So, instead of knowing the position of each a_i, b_j, we are given two sequences of regions of $\mathbb{R}^d$ denoted by $H = h_1, \ldots, h_n$ and $V = v_1, \ldots, v_m$. These regions will be either Euclidean balls, or axis-aligned boxes. They specify where the points a_i, b_j lie, and thus for each i, j, we have $a_i \in h_i$ and $b_j \in v_j$. For all $i \leqslant n$, we denote by H_i the subsequence $h_1, \ldots, h_i$, and for all $j \leqslant m$, we denote $V_j = v_1, \ldots, v_j$.

We will consider two different cases. In the *Euclidean case*, the regions are Euclidean balls in $\mathbb{R}^d$ and we use the Euclidean distance. In the *orthogonal case*, the regions are axis-aligned boxes and the distance we use is the L_∞ metric. In the Euclidean case, we will assume that we are in fixed dimension, that is, we assume that $d = O(1)$.

A *realization* of the region sequence H is a point sequence $A = a_1, \ldots, a_n$ such that $a_i \in h_i$ for all $1 \leqslant i \leqslant n$. Similarly, a realization of the region sequence V is a point sequence $B = b_1, \ldots, b_m$ such that $b_j \in v_j$ for all $1 \leqslant j \leqslant m$. We denote by $A \in_R H$ and $B \in_R V$ the fact that A is a realization of H, and B is a realization of V, respectively. When $A \in_R H$ and $B \in_R V$, we will say that (A, B) is a realization of (H, V). This will be denoted as $(A, B) \in_R (H, V)$. For two region sequences H and V, the *Fréchet distance lower bound* $\mathrm{F}^{\min}(H, V)$ is the minimum, over all realizations (A, B) of (H, V), of the discrete Fréchet distance $\mathrm{F}(A, B)$:

$$\mathrm{F}^{\min}(H, V) = \min_{(A,B) \in_R (H,V)} \mathrm{F}(A, B).$$

3 Computing the Fréchet distance lower bound $\mathrm{F}^{\min}$

In this section, we give algorithms for computing $\mathrm{F}^{\min}(H, V)$. We first give a decision algorithm that, given a real number $\delta \geqslant 0$, decides whether $\mathrm{F}^{\min}(H, V) \leqslant \delta$. Then we give an improved decision algorithm for the Euclidean case. Based on these decision algorithms, we finally give optimization algorithms, which compute $\mathrm{F}^{\min}(H, V)$ in the orthogonal case and in the Euclidean case.

We denote by h_i^δ (resp. v_j^δ) the set of points that are at distance at most δ from h_i (resp. v_j). In the Euclidean case, where h_i is a ball with radius r, the set h_i^δ is the concentric ball with radius $r + \delta$. In the orthogonal case, if $h_i = [x_1, y_1] \times \cdots \times [x_d, y_d]$, we have $h_i^\delta = [x_1 - \delta, y_1 + \delta] \times \cdots \times [x_d - \delta, y_d + \delta]$.

3.1 Decision algorithm for the orthogonal case

Our decision algorithm is based on dynamic programming. In each cell of an array with n rows and m columns, we will store two *feasibility regions* $\mathrm{FH}_\delta(i, j) \subset \mathbb{R}^d$ and $\mathrm{FV}_\delta(i, j) \subset \mathbb{R}^d$. The ith row represents the region H_i, and the jth column represents V_j. We will compute these fields row by row, from $i = 1$ to $i = n$.

As we shall see in Lemma 1, the feasibility region $\mathrm{FH}_\delta(i, j)$ represents the possible locations of a_i, where (A_i, B_j) is a realization of (H_i, V_j), and there exists a coupling that achieves $\mathrm{F}(A_i, B_j) \leqslant \delta$ whose last two pairs are not $(i - 1, j), (i, j)$. The other feasibility region $\mathrm{FV}_\delta(i, j)$ represents the possible locations of b_j, when there is such a coupling whose last two pairs are not $(i, j - 1), (i, j)$.

The pseudocode of our decision algorithm *DecideFréchetMin* is given below. Lines 1 to 8 initialize some of the fields of our array for the first row and column, as well as an extra zeroth column and row. It allows boundary cases when $i = 1$ and $j = 1$ to be handled correctly in the main loop. The main loop is from line 9 to 15.

Algorithm *DecideFréchetMin*
Input: Two sequences of regions $H = h_1, \ldots, h_n$ and $V = v_1, \ldots, v_m$, and a value $\delta \geqslant 0$.
Output: TRUE when $\mathrm{F}^{\min}(H, V) \leqslant \delta$, and FALSE otherwise.

```
1.  for i ← 1 to n
2.          FH_δ(i, 0) ← ∅
3.          FV_δ(i, 0) ← ∅
4.  for j ← 1 to m
5.          FH_δ(0, j) ← ∅
6.          FV_δ(0, j) ← ∅
7.  FH_δ(0, 0) ← ℝ^d
8.  FV_δ(0, 0) ← ℝ^d
9.  for i ← 1 to n
10.         for j ← 1 to m
11.                 if FH_δ(i − 1, j − 1) = ∅ and
                    FV_δ(i − 1, j − 1) = ∅
12.                   then FH_δ(i, j) ← FH_δ(i, j −
                            1) ∩ v_j^δ
13.                        FV_δ(i, j) ← FV_δ(i −
                            1, j) ∩ h_i^δ
14.                   else FH_δ(i, j) ← h_i ∩ v_j^δ
15.                        FV_δ(i, j) ← h_i^δ ∩ v_j
16. if FH_δ(n, m) = ∅ and FV_δ(n, m) = ∅
17.   then return FALSE
18.   else return TRUE
```

In order to prove that our decision algorithm *DecideFréchetMin* is correct, we need the following

lemma.

Lemma 1 *For any $2 \leqslant i \leqslant n$, $2 \leqslant j \leqslant m$, we have $\mathrm{F}^{\min}(H_i, V_j) \leqslant \delta$ if and only if $\mathrm{FH}_\delta(i,j) \neq \emptyset$ or $\mathrm{FV}_\delta(i,j) \neq \emptyset$. More precisely, for any $x, y \in \mathbb{R}^d$, we have:*

(a) $x \in \mathrm{FH}_\delta(i,j)$ if and only if there exists $(A_i, B_j) \in_R (H_i, V_j)$ such that $a_i = x$, and such that there exists a coupling achieving $\mathrm{F}(A_i, B_j) \leqslant \delta$ whose last two pairs are not $(i-1,j),(i,j)$.

(b) $y \in \mathrm{FV}_\delta(i,j)$ if and only if there exists $(A_i, B_j) \in_R (H_i, V_j)$ such that $b_j = y$, and such that there exists a coupling achieving $\mathrm{F}(A_i, B_j) \leqslant \delta$ whose last two pairs are not $(i,j-1),(i,j)$.

We now prove Lemma 1 when $i, j \geqslant 3$. The boundary cases where $i = 2$ or $j = 2$ can be easily checked. We only prove Lemma 1(a); the proof of (b) is similar. Our proof is done by induction on (i,j), so we assume that Lemma 1 is true for all the cells that have been handled before cell (i,j) by our algorithm; in particular, it is true for all cells $(i',j') \neq (i,j)$ such that $i' \leqslant i$ and $j' \leqslant j$.

We first assume that $x \in \mathrm{FH}_\delta(i,j)$, and we want to prove that there exists $(A_i, B_j) \in_R (H_i, V_j)$ such that $a_i = x$, and such that there exists a coupling achieving $\mathrm{F}(A_i, B_j) \leqslant \delta$ whose last two pairs are not $(i-1,j),(i,j)$. We distinguish between two cases:

- First case: $\mathrm{FH}_\delta(i-1,j-1) \neq \emptyset$ or $\mathrm{FV}_\delta(i-1,j-1) \neq \emptyset$. Then, by induction, there exists $(A_{i-1}, B_{j-1}) \in_R (H_{i-1}, V_{j-1})$ such that $\mathrm{F}(A_{i-1}, B_{j-1}) \leqslant \delta$. We also know that $\mathrm{FH}_\delta(i,j)$ was set to $h_i \cap v_j^\delta$ at line 14. In other words, $x \in h_i$, and there exists $y' \in v_j$ such that $\mathrm{dist}(x,y') \leqslant \delta$. So we extend A_{i-1} and B_{j-1} by choosing $a_i = x$ and $b_j = y'$. We extend a coupling achieving $\mathrm{F}(A_{i-1}, B_{j-1}) \leqslant \delta$ with the pair (i,j), and obtain a coupling achieving $\mathrm{F}(A_i, B_j) \leqslant \delta$ whose last two pairs are $(i-1,j-1),(i,j)$.

- Second case: $\mathrm{FH}_\delta(i-1,j-1) = \emptyset$ and $\mathrm{FV}_\delta(i-1,j-1) = \emptyset$. Then $\mathrm{FH}_\delta(i,j)$ was set to $\mathrm{FH}_\delta(i,j-1) \cap v_j^\delta$ at line 12. Thus $x \in \mathrm{FH}_\delta(i,j-1)$, so by induction, there exists $(A_i, B_{j-1}) \in_R (H_i, V_{j-1})$ such that $a_i = x$ and $\mathrm{F}(A_i, B_{j-1}) \leqslant \delta$. Since $x \in v_j^\delta$, there exists $y' \in v_j$ such that $\mathrm{dist}(x,y') \leqslant \delta$. So we extend B_{j-1} by choosing $b_j = y'$. We extend a coupling achieving $\mathrm{F}(A_i, B_{j-1}) = \delta$ with the pair (i,j), and we obtain a coupling achieving $\mathrm{F}(A_i, B_j) \leqslant \delta$ whose last two pairs are $(i,j-1),(i,j)$.

Now we assume that there exists $(A_i, B_j) \in_R (H_i, V_j)$ such that there exists a coupling $\mathcal{C}$ achieving $\mathrm{F}(A_i, B_j) \leqslant \delta$ whose last two pairs are not $(i-1,j),(i,j)$. We want to prove that $a_i \in \mathrm{FH}_\delta(i,j)$. We distinguish between two cases:

- First case: $\mathrm{FH}_\delta(i-1,j-1) \neq \emptyset$ or $\mathrm{FV}_\delta(i-1,j-1) \neq \emptyset$. It implies that $\mathrm{FH}_\delta(i,j)$ was set to $h_i \cap v_j^\delta$ at line 14. Since $A_i \in_R H_i$, we have $a_i \in h_i$. Since $B_j \in_R V_j$ and $\mathrm{F}(A_i, B_j) \leqslant \delta$, it follows that $\mathrm{dist}(a_i, b_j) \leqslant \delta$, and thus $a_i \in v_j^\delta$. Thus, $a_i \in \mathrm{FH}_\delta(i,j)$.

- Second case: $\mathrm{FH}_\delta(i-1,j-1) = \emptyset$ and $\mathrm{FV}_\delta(i-1,j-1) = \emptyset$. Then, by induction, we have $\mathrm{F}^{\min}(H_{i-1}, V_{j-1}) > \delta$, which implies that $\mathrm{F}(A_{i-1}, B_{j-1}) > \delta$, so the pair $(i-1,j-1)$ cannot appear in $\mathcal{C}$. It follows that the last three pairs of $\mathcal{C}$ can only be $(i,j-2),(i,j-1),(i,j)$ or $(i-1,j-2),(i,j-1),(i,j)$. So, by induction, we have $a_i \in \mathrm{FH}_\delta(i,j-1)$. Since $\mathrm{F}(A_i, B_j) \leqslant \delta$, we have $a_i \in v_j^\delta$. As $\mathrm{FH}_\delta(i-1,j-1) = \emptyset$ and $\mathrm{FV}_\delta(i-1,j-1) = \emptyset$, the value of $\mathrm{FH}_\delta(i,j)$ was set to $\mathrm{FH}_\delta(i,j-1) \cap v_j^\delta$ at line 14, so we have $a_i \in \mathrm{FH}_\delta(i,j)$.

This completes the proof of Lemma 1. It follows immediately from Lemma 1 that Algorithm *DecideFréchetMin* decides correctly whether $\mathrm{F}^{\min}(H, V) \leqslant \delta$. We still need to analyze this algorithm. In the orthogonal case, lines 12–15 consist in intersecting two axis-aligned boxes in fixed dimension; it can be done in $O(d)$ time. Thus, we obtain the following result:

Theorem 2 *In the d-dimensional orthogonal case, given $\delta \geqslant 0$, and given two imprecise sequences H and V of n and m points, respectively, we can decide in $O(dmn)$ time whether $\mathrm{F}^{\min}(H, V) \leq \delta$.*

3.2 Decision algorithm for the Euclidean case

In this section, we give an efficient algorithm for the Euclidean case. We will need the following result:

Lemma 3 *We can decide in $O(k)$ time whether k balls in fixed-dimensional Euclidean space have an empty intersection.*

Proof. We consider a collection of k balls in $\mathbb{R}^d$, with $d = O(1)$. We use the standard lifting-map [5, Section 1.2], which maps any point $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$ to the point $\hat{x} = \left(x_1, \ldots, x_d, \sum_{i=1}^d x_i^2\right)$. Then a ball $\mathcal{B} \subset \mathbb{R}^d$ can be mapped to an affine hyperplane $\mathcal{H} \subset \mathbb{R}^{d+1}$ such that $x \in \mathcal{B}$ if and only if $\hat{x}$ is below $\mathcal{H}$. Thus, deciding whether k balls have a non-empty intersection reduces to deciding whether there is a point x such that $\hat{x}$ is below all the corresponding hyperplanes. To do this, it suffices to decide whether there is a point x below all these hyperplanes and such that $\sum_{i=1}^d x_i^2 \leqslant x_{d+1}$. It can be done in $O(k)$ time

using an algorithm of Dyer [4] for some generalized linear programs in fixed dimension; in our case, the linear constraints for Dyer's algorithm are given by our set of hyperplanes, and the convex function we use is $(x_1, \ldots, x_{d+1}) \mapsto -x_{d+1} + \sum_{i=1}^{d} x_i^2$. □

We now explain how we implement line 13 in amortized $O(\log n)$ time. We fix the value of j, and we show how to build an incremental data structure that decides in amortized $O(\log n)$ time whether $\mathrm{FV}_\delta(i, j) = \emptyset$. To achieve this, we do not maintain the region $\mathrm{FV}_\delta(i, j)$ explicitly: we only maintain an auxiliary data structure that allows us to decide quickly whether it is empty or not. During the course of Algorithm *DecideFréchetMin*, the region $\mathrm{FV}_\delta(i, j)$ can be reset to $h_i^\delta \cap v_j$ at line 15, and otherwise, it is the intersection of $\mathrm{FV}_\delta(i-1, j)$ with h_i^δ. So at any time, we have $\mathrm{FV}_\delta(i, j) = h_{i_0}^\delta \cap h_{i_0+1}^\delta \cdots \cap h_i^\delta \cap v_j$ for some $1 \leqslant i_0 \leqslant i$.

So our auxiliary data structure needs to perform three types of operations:

1. Set $\mathcal{S} = \emptyset$.
2. Insert the next ball into $\mathcal{S}$.
3. Decide whether the intersection of the balls in $\mathcal{S}$ is empty.

When we run Algorithm *DecideFréchetMin* on column j, the sequence of n balls $h_1^\delta, \ldots, h_n^\delta$ is known in advance, but not the sequence of operations. So this is the assumption we make for our auxiliary data structure: we know in advance the sequence of balls, but the sequence of operations is given online. A trivial implementation using Lemma 3 requires $O(n)$ time per operation. Using exponential and binary search [8], we will show how to do it in amortized $O(\log n)$ time per operation.

Operation 1 is trivial to implement. To implement operation 2, suppose that, before we perform this operation, the cardinality $|\mathcal{S}|$ of $\mathcal{S}$ is $s = 2^\ell$, for some integer ℓ. Then, using Lemma 3, we check whether the intersection of the balls in $\mathcal{S}$ and the next s balls is empty. If so, we find by binary search the first subsequence of balls, starting at the balls of $\mathcal{S}$, whose intersection is empty. By Lemma 3, it can be done in $O(s \log s)$ time. Then we can perform in constant time each operation of type 2 or 3 until the next time operation 1 is performed. On the other hand, if the intersection of the balls in $\mathcal{S}$ and the next s balls is not empty, we record this fact. Then, until the cardinality of S reaches $2s = 2^{\ell+1}$, or we perform operation 1, we can perform each operation of type 2 or 3 in constant time.

This data structure needs only amortized $O(\log n)$ time per operation. Keeping one such data structure for each value of j, we can perform line 13 of Algorithm *DecideFréchetMin* in amortized $O(\log n)$ time. Similarly, we can implement line 12 in amortized $O(\log m)$ time. Overall, we obtain the following result:

Theorem 4 *In the fixed-dimensional Euclidean case, given $\delta \geqslant 0$, and given two imprecise sequences H and V of n and m points, respectively, we can decide in $O(mn \log mn)$ time whether $\mathrm{F}^{\min}(H, V) \leq \delta$.*

3.3 Optimization algorithms

We obtain algorithms for computing the Fréchet distance lower bound based on the decision algorithms above, and two standard optimization technique: We use the monotone matrix searching technique by Frederickson and Johnson [2, 7] in the orthogonal case, and we use parametric search [2, 3] in the Euclidean case. The results are summarized in the theorem below, whose proof is omitted, due to the space limit.

Theorem 5 *Given two imprecise sequences H and V of n and m points, respectively, we can compute $\mathrm{F}^{\min}(H, V)$ in $O(dmn \log dmn)$ time in the d-dimensional orthogonal case. The running time of our algorithm is $O(mn \log^2(mn) + (m^2 + n^2) \log mn)$ in the planar Euclidean case, and $O(m^2 n^2 \log^2(mn))$ in the fixed-dimensional Euclidean case.*

References

[1] H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry and Applications*, 5:75–91, 1995.

[2] P. Agarwal and M. Sharir. Efficient algorithms for geometric optimization. *Computing Surveys*, 30, 1998.

[3] P. Agarwal, M. Sharir, and S. Toledo. Applications of Parametric Searching in Geometric Optimization. In Proc. *3rd ACM-SIAM Symposium on Discrete Algorithms*, pages 72–82, 1992.

[4] M. Dyer. A Class of Convex Programs with Applications to Computational Geometry. In *Proc. 8th Annual Symposium on Computational Geometry*, pages 9–15, 1992.

[5] E. Edelsbrunner. *Geometry and Topology for Mesh Generation*. Cambridge University Press, 2001.

[6] T. Eiter and H. Mannila. Computing discrete Fréchet distance. *Technical Report*, CD-TR 94/64, Christian Doppler Laboratory for Expert Systems, TU Vienna, Austria, 1994.

[7] G. Frederickson and D. Johnson. Generalized selection and ranking: Sorted matrices. *SIAM Journal on Computing*, 13(1):14–30, 1984.

[8] A. Moffat and A. Turpin. *Compression and Coding Algorithms*. Kluwer Academic Publishers, 2002.

Arc Triangulations*

Oswin Aichholzer† Wolfgang Aigner†§ Franz Aurenhammer‡ Kateřina Čech Dobiášová§
Bert Jüttler§

Abstract

The quality of a triangulation is, in many practical applications, influenced by the angles of its triangles. In the straight line case, angle optimization is not possible beyond the Delaunay triangulation. We propose and study the concept of circular arc triangulations, a simple and effective alternative that offers flexibility for additionally enlarging small angles. We show that angle optimization and related questions lead to linear programming problems, and we define unique flips in arc triangulations. Moreover, applications of certain classes of arc triangulations in the areas of finite element methods and graph drawing are sketched.

1 Introduction

Geometric graphs and especially triangulations are an ubiquitous tool in geometric data processing [2, 8, 13]. The quality of a given triangular mesh naturally depends on the size and shape, in particular the angles, of its composing triangles. In practice, quite often the Delaunay triangulation (see, e.g., [8]) is the mesh of choice, because it maximizes the smallest angle over all possible triangulations of a given finite set of points in the plane. Still, the occurrence of 'poor' triangles cannot be avoided sometimes, especially near the boundary of the input domain, or due to the presence of mesh vertices of high edge degree.

The situation becomes different (and interesting again) if the requirement that triangulation edges be straight is dropped. In applications like finite element methods or graph drawing, the numerical and optical benefits of a graph that potentially grants nice angles can be exploited fully only if curved edges are admitted. In this paper, we try to encourage the use of so-called *arc triangulations*, which are triangulations whose edges are circular arcs. Modeling triangulations this way bears several advantages if angles are to be optimized. Small angles at the boundary can be enlarged by optimizing the arc curvatures for the given triangulation. Situations with vertices of high degree can be faced by applying angle-improving flips in arc triangles that reduce the vertex degree.

*Supported by FWF NRN 'Industrial Geometry' S92

†Institute for Software Technology, Graz University of Technology, Austria, {oaich,waigner}@ist.tugraz.at

‡Institute for Theoretical Computer Science, Graz University of Technology, Austria, auren@igi.tugraz.at

§Institute of Applied Geometry, Johannes Kepler University Linz, Austria, {Bert.Juettler, Katerina.Cech-Dobiasova}@jku.at

Maximizing the smallest angle in a combinatorially fixed arc triangulation of a point set can be formulated as a linear program. This guarantees a fast solution of this optimization problems for arc triangulations in practice. Moreover, the linear program will tell us whether a given domain admits an arc triangulation of a pre-specified combinatorial type, by checking whether its feasible region is void. In particular, flips for arcs can be defined, via optimization after the flip has been applied combinatorially. If we want to optimize equiangularity in an arc triangulation (i.e., maximize the sorted angle vector lexicographically) then we can do so as well.

We believe that arc triangulations constitute a useful tool in several important areas, including finite element methods or especially graph drawing. In view of the latter application [5, 6], it is desirable to extend our approach to optimizing angles in general plane graphs. As our simple optimization method works only for full triangulations, we complete the graphs to suitable triangulations (e.g., the constrained Delaunay [11, 4]) and treat the newly obtained angles in concatenation. In several applications, the boundary of the underlying domain will be given as a polynomial spline curve. Such domains can be approximated in a convenient way using circular biarc splines [1], and thus are naturally suited to triangulation by circular arcs.

2 Angle Optimization

Consider a straight line triangulation, $\mathcal{T}$, in a given domain D of the plane. No restrictions on D are required but, for the ease of presentation, let D be simply connected and have piecewise circular (or linear) boundary. In general, $\mathcal{T}$ will use vertices in the interior of D. We are interested in the following optimization problem: Replace each interior (i.e., non-boundary) edge of $\mathcal{T}$ by some circular arc, in a way such that the smallest angle in the resulting arc triangulation is maximized. To see that this problem is well defined, notice that the optimal solution, call it $\mathcal{T}^*$, cannot contain negative angles: The smallest angle between arcs has to be at least as large as the smallest angle that arises in $\mathcal{T}$. As a consequence, for each vertex in S, the order of its incident arcs in $\mathcal{T}^*$ coincides with the order of its incident edges in the input triangulation $\mathcal{T}$. In other words, each arc triangle in $\mathcal{T}^*$ is *well-oriented*, i.e., it has the same orientation as its straight line equivalent. Therefore, if each angle is less than π, no overlap of arcs or arc triangles in $\mathcal{T}^*$

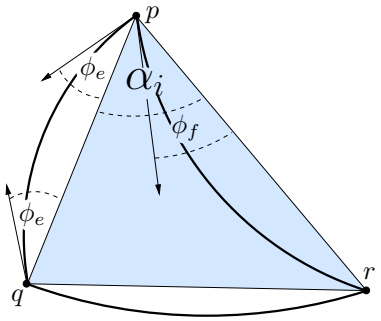


Figure 1: Angles of deviation

can occur. Interestingly, this is a specialty of triangulations; the last conclusion remains no longer true if faces with more than three arcs are present. We postulate for the rest of this paper that arc triangles be well-oriented.

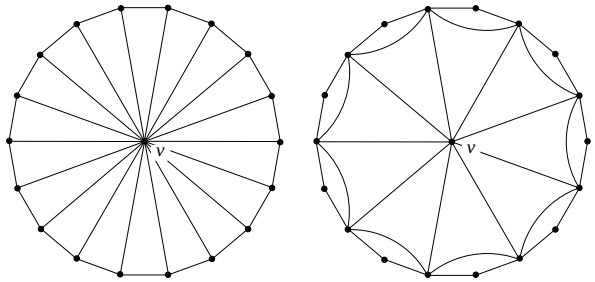
We now formulate the angle optimization problem as a linear program. For each interior edge $e = \overline{pq}$ in the triangulation $\mathcal{T}$ we introduce one variable, ϕ_e, describing the angle at which the circular arc $\widehat{pq}$ deviates from the straight connection of p and q (at these very points). Figure 1 offers an illustration. Note that ϕ_e may take on positive or negative values, depending on the sidedness of $\widehat{pq}$ with respect to e. For each edge e' of $\mathcal{T}$ on the input boundary ∂D, we fix $\phi_{e'}$ to the value $d_{e'}$ given by ∂D.[1] The inequalities for the linear program now stem from the angles α_i arising in $\mathcal{T}$. If e and f are the two edges of $\mathcal{T}$ that define α_i, we consider the angle between the two respective circular arcs, $\beta_i = \phi_e + \alpha_i + \phi_f$, and we put

$$\varepsilon \leq \beta_i \ .$$

The linear objective function L, which is to be maximized, is just $L = \varepsilon$, what clearly maximizes the smallest angle $\beta_{\min}$ in the arc triangulation. There are precisely $3 \cdot (2n-h-2)$ inequalities and $3n-2h-3$ variables, if n is the total number of vertices, and h among them are situated on ∂D.

Sometimes the objective is to optimize not only the smallest angle, but rather to maximize lexicographically the sorted list of all arising angles, as is guaranteed by the Delaunay triangulation in the straight line case. This can be achieved by repeatedly solving the linear program above, keeping angles that have been optimized already as constants. (This is a nontrivial task. Depending on the solver, minimum angles do typically occur at several places, and the optimal ones among them have to be singled out.) By modifiying or adding constraints the results may be adapted to various needs, as avoiding angles larger than π or obtaining arc triangles 'as equilateral as possible'. We consider the flexibility of our simple approach as an important feature in practice.

[1] We have $d_{e'} = 0$ if e' is a line segment. However, we can keep $\phi_{e'}$ variable and bound it from above by some threshold $t > d_{e'}$.



(a) Delaunay triangulation (b) Arc triangulation

Figure 2: Flip-optimized arc triangulation starting from a Delaunay triangulation.

3 Flipping in Arc Triangles

The fact that every simple polygon can be triangulated with straight line segments is folklore. Again, a domain D with piecewise circular boundary need not admit *any* triangulation, even if circular arcs may be used. It is known that a linear number of Steiner points is required in the worst case to ensure an arc triangulation [1].

One of the arising questions is: Given the domain D and a (combinatorial) triangulation $\mathcal{T}_c$ in D (possibly with interior points), can $\mathcal{T}_c$ be realized by circular arcs? For deciding this, we can now utilize the linear program formulated in Section 2. A realizing arc triangulation exists if and only if the feasible region of the linear program is nonempty. As a particularly nice feature, this enables us to define flip operations in arc triangulations, as is described below.

Consider some arc triangulation $\mathcal{A}$ in the domain D. Each interior arc $\widehat{pq}$ of $\mathcal{A}$ lies on the boundary of two arc triangles. Let r and s be the two vertices of these arc triangles different from p and q. Flipping $\widehat{pq}$ by definition means removing $\widehat{pq}$ from $\mathcal{A}$, establishing an arc between r and s combinatorially, and optimizing over the resulting triangulation. The new arc triangulation, if it exists, will contain a unique circular arc between r and s. In case of nonexistence, we declare the arc $\widehat{pq}$ as not flippable. Observe that an arc flip may change various circular arcs geometrically (by optimizing over their curvature), whereas only a single arc is exchanged combinatorially. An arc flip thus is a geometrically global operation which is combinatorially local.

Optimizing angles with arc flips is a powerful (though maybe costly) tool. We demonstrate the positive effect of sequences of such flips with Figures 2a and 2b. A significant improvement over the Delaunay triangulation becomes possible (in fact, the smallest angle is doubled in this example) by reducing the degree of a particular vertex, v. In general, we observe that small angles in a straight line triangulation stem from one of two reasons: (1) The geometry of the underlying domain D (plus its vertex set) forces slim

triangles in the vicinity of ∂D. These 'boundary effects' can usually be mildened by mere geometric optimization of the corresponding arc triangulation. (2) Vertices of degree k naturally impose an upper bound of $\frac{2\pi}{k}$ on the smallest arising angle. This situation can be remedied only with combinatorial changes, and in contrast to the straight edge case, this is indeed possible for arc triangulations. (For straight edges, the combinatorics of the Delaunay triangulation is already optimal.)

4 Special Arc Triangles

An arc triangle ∇ is termed a *π-triangle* if the sum of its interior angles is π. In the isoparametric approach to finite element methods [10], based on the fact that π-triangles are images of straight triangles under a Möbius transformation, an approximation by conformal (angle-preserving) Bézier patches can be obtained, when the sum of angles is optimized towards π. Moreover, if the angle sum is even equal to π, simple inverse geometry mappings can be constructed. We omit details (and proofs) in this version.

Property 1 *Let ∇ be some arc triangle. The following three properties are equivalent.*
(a) ∇ is a π-triangle.
(b) The three supporting circles of ∇ intersect in a common point exterior to ∇.
(c) ∇ is the image of a straight line triangle under a unique Möbius transformation.

Property 2 *Any π-triangle is contained in the circumcircle of its vertices.*

In view of the mentioned properties, it is worthwhile to study *π-triangulations*. Such triangulations will not always exist, depending on the boundary domain D, and in particular the sum of its inner angles, but they do, of course, if D is a simple polygon.

For the remainder of this section, let D be a simple polygon, and $\mathcal{T}$ be some straight line triangulation in D. The geometry of any arc triangulation $\mathcal{A}$ in D that is combinatorially equivalent to $\mathcal{T}$ is determined by the vector $\Phi(\mathcal{A})$ of deviation angles $\phi(a_i)$ for the interior arcs a_i of $\mathcal{A}$; see Section 2. Interpreting $\Phi(\mathcal{A})$ as a point in high dimensions, we can talk of the space of arc triangulations for $\mathcal{T}$. The next lemma is important in view of optimizing a given π-triangulation.

Lemma 1 *Let $\mathcal{T}$ have n vertices, h of which lie on the boundary of D. The dimension of the space of π-triangulations for $\mathcal{T}$ is $n-h$.*

Lemma 1 remains true if $\mathcal{T}$ is replaced by any π-triangulation of D. In practice, the input is most likely a straight line triangulation, which is to be optimized into a π-triangulation with maximum smallest angle. Figure 3 displays an example. The change

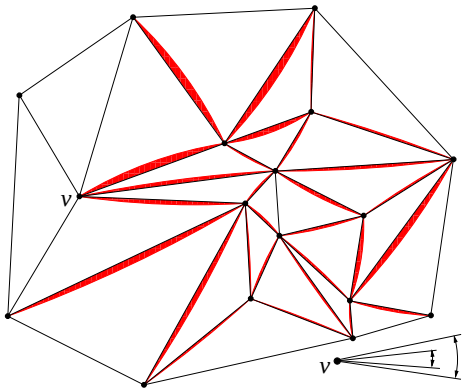


Figure 3: Straight line triangulation and its angle-maximized π-triangulation superimposed

angle sum	Delaunay min	min arc angle	gain
180°	18.03°	22.52°	25%
179° - 181°	-"-	22.92°	26%
175° - 185°	-"-	24.88°	38%
170° - 190°	-"-	27.53°	50%
160° - 200°	-"-	31.77°	72%

Table 1: Improvement of angles in (almost) π-triangulations

does not appear dramatic, but observe that the smallest angle (occurring at vertex v) almost doubles, from 9.7° to 19°. No arc flips have been applied. Table 1 shows experimental data for a larger input (500 random points, postprocessed to keep a certain interpoint distance as in realistic meshes). We see that the gain reduces for larger Delaunay meshes but is still significant, especially if the condition on the angle sum in the triangles is relaxed from π to a small intervall around that value.

5 Graph Drawing

Literature on drawing graphs nicely in the plane is large; see e.g. [5, 14]. Most algorithms take as input an abstract graph G and produce a layout of the vertices of G such that the resulting straight line (or orthogonal) drawing is aesthetically pleasing, and/or satisfies certain application criteria. On the theoretical side, bounds on the achievable angular resolution are known for various classes of graphs [7, 12], including planar graphs.

Results for curvilinear drawings of graphs are comparatively sparse. See, for example, [3, 9] and references therein, who give lower bounds and algorithms for drawing graphs on a grid with curved edges (including circular multiarcs), and [6] where a method based on physical simulation is proposed. To our knowledge, no algorithm has been given that draws a graph with (single) circular arcs under some optimization criterion. Here we actually consider a sim-

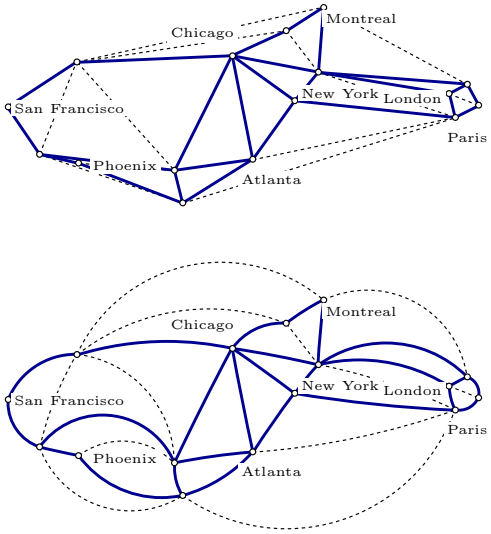


Figure 4: IP backbone graph, straight line and optimally redrawn.

pler setting, namely, for a given planar straight line embedding of a graph G, the problem of *redrawing* G with curved edges in an optimal way. In a redrawing, the positions of the vertices are kept fixed. This may be a natural demand, for instance, in certain geographical applications.

Let us describe how maximizing the smallest angle in a circular arc redrawing of G can be achieved. It is tempting to apply the linear optimization method from Section 2 to G directly. This, however, bears the risk of arc overlaps getting out of control. The way out is to embed G in some triangulation $\mathcal{T}$ first, and treat respective sums of angles as single entities to be optimized. More precisely, for each angle ϱ in G, given by the concatenation of angles $\alpha_1, \ldots, \alpha_k$, $k \geq 1$, in $\mathcal{T}$ we use the constraint

$$\varepsilon \leq \sum_{i=1}^{k} \beta_i$$

with each β_i expressed by the corresponding straight line triangulation angle α_i and its two deviation variables ϕ_e and ϕ_f as in Section 2. The quality of optimization depends on the chosen triangulation, which will be subject of future research (cf. Section 3). Also, the entire angle vector $\varrho_1, \ldots, \varrho_m$ for G can be optimized, in an iterative way as before. Additional restrictions may be posed, like $\varrho_j < \pi$ or $\varrho_j < \frac{\pi}{2}$, in order to preserve obtuse or sharp angles in G.

The adjacency graphs in Figure 4 exemplify the effect of our circular arc redrawing method. The results seem satisfactory, in spite of the fact that vertices are required not to move. Our results compare well to, e.g. [6], who use for optimization the additional freedom of placing vertices, though at a price of high computation cost. For our method, the number of vertices of the input graph is no limitation, as far as applications from graph drawing are concerned.

6 Future Work

Circular arc triangulations are a flexible and computationally controllable structure with potential impact but, so far, with lack of interest from computational geometry. They lead to simple and fast graph redrawing procedures, and bear novel aspects for finite element methods. Among the open questions raised are the convergence of the angle-increasing arc flipping process in Section 3, and an extension of the presented results to three dimensions, for tetrahedral volumes with spherical faces. We will elaborate on the properties of such 3D primitives and their meshes in a forthcoming paper.

References

[1] O. Aichholzer, F. Aurenhammer, T. Hackl, B. Juettler, M. Oberneder, Z. Sir. Computational and structural advantages of circular boundary representation. Int'l J. Computational Geometry & Applications, to appear.

[2] M. Bern, D. Eppstein. Mesh generation and optimal triangulation. In: D.-Z.Du, F.Hwang (eds), Computing in Euclidean Geometry, Lecture Notes Series on Computing 4, World Scientific, 1995, 47-123.

[3] C.C. Cheng, C.A. Duncan, M.T. Goodrich, S.G. Kobourov. Drawing planar graphs with circular arcs. Proc. 7th Int. Symposium on Graph Drawing, 1999, Springer LNCS 1771, 2000, 117–126.

[4] L.P. Chew. Constrained Delaunay triangulations. Algorithmica 4 (1989), 97-108.

[5] G. Di Battista, P. Eades, R. Tamassia, I.G. Tollis. *Graph Drawing - Algorithms for the Visualization of Graphs.* Prentice Hall, 1999.

[6] B. Finkel, R. Tamassia. Curvilinar graph drawing using the force-directed method. Proc. 12th Int. Symposium on Graph Drawing, 2004, Springer LNCS 3383, 2004, 448-453.

[7] M. Formann, T. Hagerup, J. Haralambides, M. Kaufmann, F.T. Leighton, A. Symvonis, E. Welzl, G. Wöginger. Drawing graphs in the plane with high resolution. SIAM J. Computing 22 (1993), 1035 - 1052.

[8] S. Fortune. Voronoi diagrams and Delaunay triangulations. In: D.-Z.Du, F.Hwang (eds), Computing in Euclidean Geometry, Lecture Notes Series on Computing 4, World Scientific, 1995, 225-265.

[9] M.I. Goodrich, C.G. Wagner. A framework for drawing planar graphs with curves and polylines. J. Algorithms 37 (2000), 399–421.

[10] T.J.R Hughes. The Finite Element Method – Linear Static and Dynamic Finite Element Analysis, Reprint, Dover Publications, New York.

[11] D.T. Lee, A.K. Lin. Generalized Delaunay triangulation for planar graphs. Discrete & Computational Geometry 1 (1986), 201-217.

[12] S. Malitz, A. Papakostas. On the angular resolution of planar graphs. Proc. 24th Ann. ACM Symp. on Theory of Computing, 1992, 527-538.

[13] J. Shewchuk. What is a good linear element? Interpolation, conditioning, and quality measures. Proc. 11th International Meshing Roundtable, 2002, 115-126.

[14] K. Sugiyama. *Graph Drawing and Applications for Software and Knowledge Engineers.* World Scientific, 2002.

3-Colorability of Pseudo-Triangulations*

Oswin Aichholzer† Franz Aurenhammer‡ Thomas Hackl† Clemens Huemer§ Alexander Pilz†
Birgit Vogtenhuber†

Abstract

Deciding 3-colorability for general plane graphs is known to be an NP-complete problem. However, for certain classes of plane graphs, like triangulations, polynomial time algorithms exist. We consider the family of pseudo-triangulations (a generalization of triangulations) and prove NP-completeness for this class. The complexity status does not change if the maximum face-degree is bounded to four, or pointed pseudo-triangulations with maximum face degree five are treated. As a complementary result, we show that for pointed pseudo-triangulations with maximum face-degree four, a 3-coloring always exists and can be found in linear time.

1 Introduction

The *chromatic number* of a graph is the smallest number of colors needed to color its vertices so that no two adjacent vertices share the same color. Graphs with chromatic number 3 are said to be *(vertex) 3-colorable*. Determining the chromatic number of a graph is known to be a computationally hard problem. Interestingly, deciding 3-colorability of a *plane* graph is still NP-complete [9]. For the class of triangulations, though, 3-colorability can be decided in linear time; it is necessary and sufficient that every interior (i.e., non-extreme) vertex has even degree. Alternatively, we can use the following constructive approach: Start with the three different colors of a single triangle. Then the color of the third vertex of each edge-adjacent triangle is determined. This process is iterated until either a contradiction occurs (an already colored vertex is forced to have a different color) or a proper coloring is obtained.

Also for some other types of graphs the decision problem can be solved efficiently. Beside (obvious) graph classes like paths, cycles, trees, and quadrangulations, the class of maximal outerplanar graphs (or, equivalently, triangulations of polygons, or of point sets in convex position) is also 3-colorable. Ellingham et al. [3], and in a different formulation Diks, Kowalik, and Kurowski [2], give a characterization of planar graphs with isolated non-triangular faces that are 3-colorable. Moreover, 3-colorability is linear-time decidable for general locally connected graphs [5]. See [8] for a survey on 3-colorability.

In the present work we consider the class of pseudo-triangulations, which generalize triangulations in several aspects. In fact, as we shall see, this class is rich enough to lead to a wide spectrum of coloring results. We show that deciding 3-colorability for pseudo-triangulations is NP-complete. In fact, any plane geometric graph can be reduced, with respect to 3-coloring, to a (pointed) pseudo-triangulation. For the special case of pointed pseudo-triangulations with constant maximum face-degree, the problem remains NP-complete if the degree bound is at least five. As a complementary result, we prove that for pointed pseudo-triangulations with maximum face-degree four, a 3-coloring always exists and can be found in linear time. Some intermediate results for a varying number of pointed vertices are given as well.

We assume that point sets that serve as vertex sets for geometric graphs are in general position, that is, no three points lie on a common straight line. For a point set S, let $n = |S|$, and denote with $|CH(S)|$ the number of extreme points of S.

2 (Pointed) Pseudo-Triangulations

Pseudo-triangulations are a versatile generalization of the well-known concept of (geometric) triangulations [7]. Instead of triangles, their faces are *pseudo-triangles*, that is, simple polygons with exactly three convex vertices. In a geometric straight-line graph G, a vertex v is called *pointed* if there exists a line through v such that all edges of G incident to v lie on one side. The *rank* of a pseudo-triangulation is its number of non-pointed vertices; see [1] for further details. Pseudo-triangulations with rank zero are called pointed. These structures are of particular interest, because they are planar Laman graphs, and are minimally rigid [10].

Lemma 1 *Any plane geometric graph $G(S)$ on S can be extended to a pseudo-triangulation, $T(S')$, such that:*

- $S \subseteq S'$ *and* $|S'| = \Theta(n)$

*This work was initiated during the Sixth European Pseudo-Triangulation Working Week in Ratsch an der Weinstraße, Austria, September 2009. O. A., F. A., T. H., A. P., and B. V. were supported by the FWF [Austrian Fonds zur Förderung der Wissenschaftlichen Forschung] under grant S9205-N12, NFN Industrial Geometry. Research of C. H. partially supported by projects MEC MTM2009-07242 and Gen. Cat. DGR 2009SGR1040.

†Institute for Software Technology, Graz University of Technology, Austria, [oaich|thackl|apilz|bvogt]@ist.tugraz.at

‡Institute for Theoretical Computer Science, Graz University of Technology, Austria, auren@igi.tugraz.at

§Departament de Matemàtica Aplicada IV, Universitat Politècnica de Catalunya, clemens.huemer@upc.edu

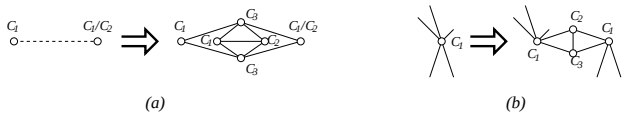


Figure 1: (a) Transforming a plane graph into a pseudo-triangulation. (b) Transforming a plane graph into a pointed plane graph. Note that these gadgets are arbitrarily flat. Colors C_1 to C_3 indicate possible color configurations.

- *$G(S)$ is 3-colorable if and only if $T(S')$ is 3-colorable*
- *the rank of $T(S')$ equals the number of non-pointed vertices in $G(S)$*

Proof. From [7, Theorem 2.6] it follows that, by adding a linear number of edges, any plane geometric graph $G(S)$ can be augmented in polynomial time to a pseudo-triangulation $T(S)$ without changing the pointedness of the underlying vertices. Instead of adding single edges, we use gadgets like in Figure 1(a) to connect two vertices in order to obtain a pseudo-triangulation $T(S')$.

Observe that $|S'| = \Theta(n)$ holds, as one gadget adds only a constant number of points. Also, the gadgets can be added in a way such that they do not change the pointedness of the involved vertices. (Under the general position assumption, the gadgets can be made sufficiently narrow.) As the additional vertices introduced with each gadget are all pointed, it follows that the number of non-pointed vertices remains unchanged.

Finally, adding the gadgets does not add additional coloring restrictions. The connected vertices might be colored arbitrarily (identically or differently), and still the added vertices of the gadget are 3-colorable. Thus $G(S)$ is 3-colorable if and only if $T(S')$ is 3-colorable. □

As planar graph 3-colorability is known to be NP-complete [9], the previous lemma already leads to the following NP-completeness result.

Theorem 2 *Deciding whether a pseudo-triangulation is 3-colorable is NP-complete.*

Proof. By Lemma 1, we can obtain a pseudo-triangulation T from each plane graph G such that G is 3-colorable if and only if T is 3-colorable. As the transformation can be done in polynomial time, and only a linear number of edges and vertices are added, the claimed NP-completeness result follows. □

Pointed pseudo-triangulations are an important subclass of pseudo-triangulations. They minimize the number of edges over all pseudo-triangulations and thus, in some way, also the number of color restrictions. Nevertheless, we will show that even for this restricted class, 3-colorability is NP-complete. To this end, we prove that pointed planar graph 3-colorability is NP-complete, from which NP-completeness of pointed pseudo-triangulation 3-colorability follows.

Lemma 3 *Deciding whether a pointed plane geometric graph is 3-colorable is NP-complete.*

Proof. We show how to transform a given plane straight-line graph G into a pointed plane straight-line graph G', such that G is 3-colorable if and only if G' is 3-colorable. W.l.o.g., assume that there are no horizontal edges in the given embedding of G, as otherwise we slightly rotate the plane. Now every non-pointed vertex v of G is replaced by two duplicates v_L and v_R of v. The two copies are placed sufficiently close to the left (v_L) and to the right (v_R) of v, respectively. All edges incident to v from above are now incident to v_L, and all edges incident to v from below are moved to v_R. In addition, v_L and v_R are connected by a small construction consisting of five edges, as shown in Figure 1(b).

By the general position assumption, the resulting graph G' is plane. Only a linear number of additional vertices has been added, and all the vertices are now pointed. Moreover, the gadget connecting v_L and v_R ensures that in a proper 3-coloring of G' both vertices have to get the same color. Thus G is 3-colorable if and only if G' is 3-colorable. NP-completeness follows as the transformation can be done in polynomial time. □

Combining Lemma 1 with Lemma 3 gives the following theorem.

Theorem 4 *Deciding whether a pointed pseudo-triangulation is 3-colorable is NP-complete.*

The last result gives rise to an interesting question. On the one hand, pointed pseudo-triangulations have rank 0 and, as shown in Theorem 4, it is NP-complete to decide their 3-colorability. On the other hand, triangulations have maximum rank $r_{max} = n - |CH(S)|$, i.e., all interior vertices are non-pointed, and, as already mentioned in the introduction, 3-colorability can be decided in linear time. So it is natural to ask for which rank the change from 'easy' to 'intractable' happens. With the next two theorems we make a first step towards answering this question. (In the following, several proofs are omitted due to space constraints.)

Theorem 5 *For all constants $c \geq 1$ and any $r \leq r_{max} - \Theta(\sqrt[c]{n})$ it is NP-complete to decide whether a pseudo-triangulation of rank r is 3-colorable.*

Theorem 6 *Whether a pseudo-triangulation $T(S)$ of rank $r \geq r_{max} - \Theta(\log n)$ is properly 3-colorable can be decided in polynomial time.*

3 Constant Maximum Face-Degree

In this section, we consider pseudo-triangulations with constant maximum face-degree, that is, pseudo-triangulations where each interior face is a pseudo-triangle with at most a (small) constant number of vertices. The following two statements will allow us to transform any pseudo-triangulation with high maximal face-degree into one with smaller maximal face-degree while keeping rank and colorability properties; cf. Figure 2.

Lemma 7 *Any pseudo-triangle with $k > 5$ vertices can be subdivided into two pseudo-triangles of sizes strictly less than k, by adding an interior vertex of degree two, such that the pointedness of the involved vertices persists.*

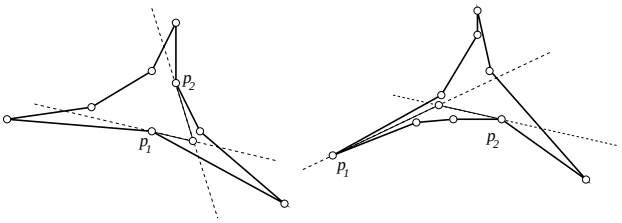


Figure 2: Subdividing large pseudo-triangles.

Corollary 8 *Every pseudo-triangulation $T(S)$ with maximum face-degree $k > 5$ can be transformed into a pseudo-triangulation $T'(S')$ with maximum face-degree five in polynomial time such that:*

- *$S \subseteq S'$ and $|S'| = \Theta(n)$*
- *$T(S)$ is 3-colorable if and only if $T'(S')$ is 3-colorable*
- *the rank of $T'(S')$ equals the rank of $T(S)$*

3.1 Face-Degree ≤ 5 Pseudo-Triangulations

By combining Corollary 8 with Theorem 4, we obtain a result for pointed pseudo-triangulations with maximum face-degree five.

Corollary 9 *Deciding whether a pointed pseudo-triangulation with maximum face-degree five is 3-colorable is NP-complete.*

A more general statement (that includes the previous corollary) is the following.

Theorem 10 *For all constants $c \geq 1$ and any rank $r \leq r_{max} - \Theta(\sqrt[c]{n})$ it is NP-complete to decide whether a pseudo-triangulation of rank r and with maximum face-degree five is 3-colorable.*

3.2 Face-Degree ≤ 4 Pseudo-Triangulations

Pseudo-triangles of size larger than five can always be subdivided as described in Lemma 7. This result cannot be extended to smaller pseudo-triangles. In fact, the situation changes completely if we bound the face-degree of a pseudo-triangulation by four.

Theorem 11 *Pointed pseudo-triangulations with maximum face-degree four are 3-colorable.*

Proof. To prove the theorem, we use the concept of combinatorial (pointed) pseudo-triangulations [6]. These are combinatorial embeddings of (pointed) pseudo-triangulations, where the edges need not be straight lines and pointedness is not a geometric property anymore. Instead, each pointed vertex has a mark in one incident face, namely the one where it is pointed to, and for each face all but three vertices (the corners) have marks in this face. The only exception is the outer face, where all (at least three) incident vertices have their mark in. Note that a given combinatorial (pointed) pseudo-triangulation can be embedded such that every angle with a mark is larger than π and all other angles are smaller than π [4, Section 5.2]. Thus, with respect to 3-colorability, combinatorial pointed pseudo-triangulations are equivalent to geometric pointed pseudo-triangulations.

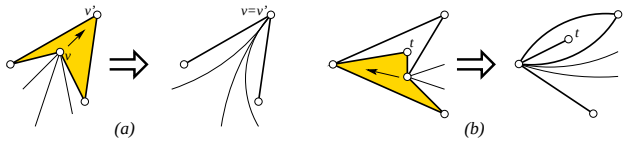


Figure 3: Move operation to collapse a pseudo-triangle (a), and a degenerate case (b).

For an interior vertex v we define a merge operation for the pseudo-triangle ∇ to which v is pointed. This operation identifies v with the antipodal vertex v' in ∇, by 'moving' v towards v', see Figure 3(a). In this way ∇ collapses, but the remaining graph is still a valid combinatorial pointed pseudo-triangulation with one vertex, one face, and two edges less.

We iterate this process as long as we have interior vertices. This can be done, as each such vertex is always pointed towards a pseudo-triangle of size four. Whenever there exist interior vertices of degree two, they are merged before other vertices, to avoid degenerate cases as shown in Figure 3(b). Such degeneracies can only happen if vertex t has degree two. Observe that all arguments also hold in the degenerate case, as we still have all relevant properties of combinatorial pointed pseudo-triangulations. However, for simplicity, we prefer to avoid degeneracies.

At the end of all merging steps, no interior vertices are left, and we obtain a (combinatorial) triangulation of a convex point set. Such triangulations are

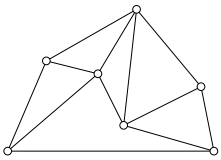


Figure 4: A pseudo-triangulation with rank 1 (one non-pointed vertex) and maximum face-degree four, which can not be 3-colored.

well known to be 3-colorable, and we can assign their colors in linear time.

We finally invert the above process and replicate, in each reversed merge step, the color of the original vertex for the duplicated vertex. This keeps the 3-coloring valid, as these vertices are not connected in the original graph. After all merge steps are undone, the given pointed pseudo-triangulation is 3-colored. □

Note that the above proof also provides a linear time algorithm to find a 3-coloring. The obtained 3-coloring is special in the sense that, for every interior face of size four of the pointed pseudo-triangulation, its reflex vertex has the same color as its antipodal vertex. In fact, up to permutation of the three colors, there is only one coloring with this property. This follows from the facts that (1) a 3-coloring of a triangulation of a convex point set is unique (up to permutation), and (2) the merge steps used in the above proof lead to a unique triangulation of the convex set, independent of the order they are carried out.

Pointed pseudo-triangulations with bounded face-degree four are a special structure concerning 3-colorability. Note that triangulations of convex point sets also fall into that category. Investigating the influence of the rank of a bounded face-degree four pseudo-triangulation on 3-colorability reveals that already a rank of 1 allows pseudo-triangulations which are not properly 3-colorable; see Figure 4. Note that all interior vertices in this example have even degree. So the parity property, which can be used to prove 3-colorability for triangulations, does not carry over to pseudo-triangulations of general rank. In addition, there exist 3-colorable examples with non-pointed interior vertices of odd degree.

In fact, we can prove NP-completeness for a wide range of ranks for maximum face-degree four pseudo-triangulations.

Theorem 12 *For all constants $c \geq 1$ and any r, $\Theta(\sqrt[c]{n}) \leq r \leq r_{max} - \Theta(\sqrt[c]{n})$, it is NP-complete to decide whether a rank r pseudo-triangulation with maximum face-degree four is 3-colorable.*

4 Final remarks

To summarize, we have the following results for pseudo-triangulations of maximum face-degree four:

- rank 0 (pointed pseudo-triangulations): always 3-colorable.
- rank r, $\Theta(\sqrt[c]{n}) \leq r \leq r_{max} - \Theta(\sqrt[c]{n})$: NP-complete.
- rank r, $r_{max} - \Theta(\log n) \leq r \leq r_{max}$: decidable in polynomial time.
- rank $r_{max} = n - |CH(S)|$ (triangulations): decideable in linear time.

For rank r pseudo-triangulations of maximum face-degree five, and rank r pseudo-triangulations without any face-degree bound, 3-colorability is NP-complete as long as $r \leq r_{max} - \Theta(\sqrt[c]{n})$. For both classes, 3-colorability is decidable in polynomial time if $r \geq r_{max} - \Theta(\log n)$. Where precisely do the changes between 'NP-complete' and 'polynomial time decidable' happen? What can be said if a pseudo-triangulation is 'almost pointed' (small constant rank)?

5 Acknowledgments

We would like to thank the participants of the 6^{th} European Workshop on Pseudo-Triangulations, held in Ratsch an der Weinstraße, Austria, September 2009, for stimulating discussions.

References

[1] O. Aichholzer, F. Aurenhammer, P. Brass, H. Krasser. *Pseudo-triangulations from surfaces and a novel type of edge flip.* SIAM Journal on Computing, 32, pp. 1621–1653, 2003.

[2] K. Diks, L. Kowalik, M. Kurowski. *A new 3-color criterion for planar graphs.* In: L. Kučera, editor, Graph-Theoretic Concepts in Computer Science, Lecture Notes in Computer Science, 2573, pp. 138–149, Springer, 2002.

[3] M. N. Ellingham, H. Fleischner, M. Kochol, E. Wenger. *Colorability of planar graphs with isolated nontriangular faces.* Graphs and Combinatorics, 20(4), pp. 443–446, 2004.

[4] R. Haas, D. Orden, G. Rote, F. Santos, B. Servatius, H. Servatius, D. Souvaine, I. Streinu, W. Whiteley. *Planar minimally rigid graphs and pseudo-triangulations.* Computational Geometry, Theory and Applications, 31, pp. 31–61, 2005.

[5] M. Kochol. *Linear algorithm for 3-coloring of locally connected graphs.* In: K. Jansen et al., editors, Experimental and Efficient Algorithms, Lecture Notes in Computer Science, 2647, pp. 191–194, Springer, 2003.

[6] D. Orden, F. Santos, B. Servatius, H. Servatius, *Combinatorial pseudo-triangulations.* Discrete Mathematics, 307, pp. 554–566, 2007.

[7] G. Rote, F. Santos, I. Streinu, *Pseudo-triangulations–a Survey.* Contemporary Mathematics, 453, pp. 343–410, 2008.

[8] R. Steinberg. *The state of the three color problem.* In: J. Gimbel, J. W. Kennedy, and L. V. Quintas, editors, Quo vadis, graph theory?: A source book for challenges and directions. Annnals of Discrete Mathematics, 55, pp. 211–248, North Holland, 1993

[9] L. Stockmeyer. *Planar 3-colorability is polynomial complete.* SIGACT News, 5, pp. 19–25, 1973.

[10] I. Streinu. *A combinatorial approach to planar non-colliding robot arm motion planning.* Proc. 41^{st} IEEE Symp. FOCS, pp. 443–453, 2000.

Connecting Obstacles in Vertex-Disjoint Paths

Marwan Al-Jubeh* Gill Barequet†* Mashhood Ishaque*
Diane L. Souvaine* Csaba D. Tóth‡* Andrew Winslow*

Abstract

Given a set of k disjoint convex polygonal obstacles inside a triangular container, we add straight-line noncrossing edges such that each obstacle has three vertex-disjoint paths to the container. We prove combinatorial bounds on the minimum number of edges that are always sufficient and sometimes necessary.

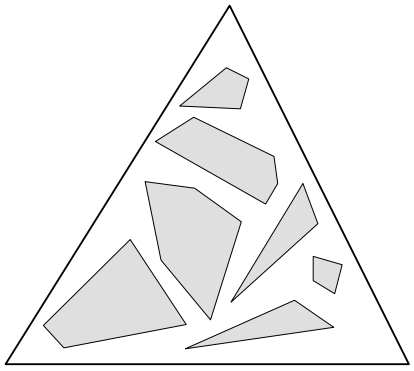


Figure 1: A triangular container with disjoint convex obstacles.

1 Introduction

A given graph is said to be k-connected if it remains connected upon deleting any $k-1$ vertices along with the incident edges. A k-connected graph has k vertex-disjoint paths between any two nodes. An important area of research in graph theory and computational geometry is the problem of connectivity augmentation. The k-connectivity augmentation problem asks for the minimum number of edges needed to augment a graph to make it k-connected. Edge-connectivity augmentation is defined analogously.

In abstract graphs, the connectivity augmentation problem can be solved in linear time for $k = 2$ [4, 6], and in polynomial time for any fixed k [5]. For a given planar graph, the augmentation that has to preserve graph planarity, is called *planarity-preserving* augmentation. Unfortunately, the problem is NP-hard even for $k = 2$ [3]. For a given planar graph that has already been embedded in the plane, if the augmentation has to respect the given embedding, the augmentation is said to be *embedding preserving*. For a planar straight-line graph, the minimum embedding-preserving augmentation using noncrossing straight-line edges is NP-Hard for any $2 \leq k \leq 5$ [7].

There are two possible approaches to get around the NP-Hardness of the augmentation problem: (*i*) approximation algorithms (e.g., there is a 2-approximation algorithm for planarity-preserving connectivity augmentation for $k = 2$, which runs in $O(n \log n)$ time [3]); and (*ii*) proving combinatorial bounds on the number of new edges in terms of the number of vertices (e.g., Al-Jubeh *et al.* [2] show that $2n-2$ new edges are always sufficient and sometimes necessary for the embedding-preserving 3-edge-connectivity augmentation of a planar straight line graph with n vertices if augmentation is possible). Tóth and Valtr [8] characterized the planar straight line graphs that can be augmented to 3-connectivity. These graphs are called *3-augmentable*. It remains an open problem what is the minimum number of new edges that are sufficient for the 3-connectivity augmentation of every 3-augmentable planar straight line graphs with n vertices.

In this paper we consider a special type of augmentation problem (see the formulation below) and provide combinatorial bounds on the minimum number of necessary and sufficient new edges.

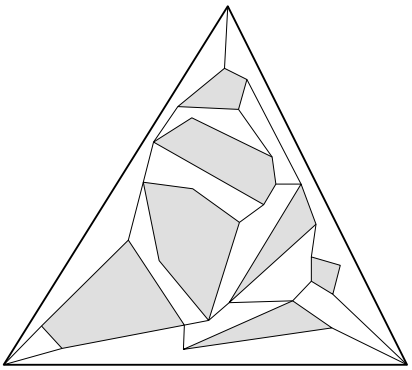


Figure 2: Adding noncrossing straight-line edges so as to make each obstacle connected by three vertex-disjoint paths to the triangular container.

*Dept. of Computer Science, Tufts University, Medford, MA. This material is based upon work supported by the National Science Foundation under Grant No. 0830734. {maljub01,barequet,mishaque,dls,awinsl02}@cs.tufts.edu
†Dept. of Computer Science, Technion, Haifa, Israel
‡Dept. of Mathematics, University of Calgary, AB, Canada. cdtoth@ucalgary.ca

1.1 Problem Definition

Given a set of k disjoint convex polygonal obstacles inside a triangular container, add straight-line non-crossing edges such that each obstacle has 3 vertex-disjoint paths to the three vertices of the container. The three paths should start at distinct vertices of the obstacle and end at distinct vertices of the container. They can use the edges of the obstacles arbitrarily.

1.2 When is Augmentation Possible?

If the obstacles are not convex, it might not be possible at all to add edges such that each obstacle has three vertex-disjoint paths to the container. In Figure 3 the inner-most obstacle "sees" only three other vertices, all of which belong to the same obstacle. Since it is not possible to route three vertex disjoint paths along the same obstacle without adding edges in the interior of the obstacle, this example is not augmentable.

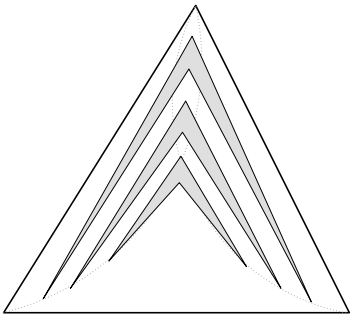


Figure 3: Non-convex obstacles may not be connected to the boundary by three vertex-disjoint paths.

For a set of disjoint convex obstacles inside the triangular container, every triangulation of the free space around the obstacles is a 3-connected graph [8]. It is easy to see that there are three vertex-disjoint paths from every obstacle to the container along the edges of a triangulation. For any particular obstacle, add a new internal node p and connect it to the boundary of the obstacle at three distinct vertices. Similarly, add a node q outside the triangular container and add the three edges connecting q to the corners of the container. It can be easily verified that the new graph is still 3-connected, which implies that there are three vertex-disjoint paths from p to q. Hence, there are three vertex-disjoint paths that start at distinct vertices of the obstacle and end at distinct vertices of the container. These three paths can be determined using any max-flow algorithm [1].

Although a triangulation contains the desired augmentation as a subgraph, it may contain too many edges. In this paper we show how to perform this augmentation by using much fewer edges.

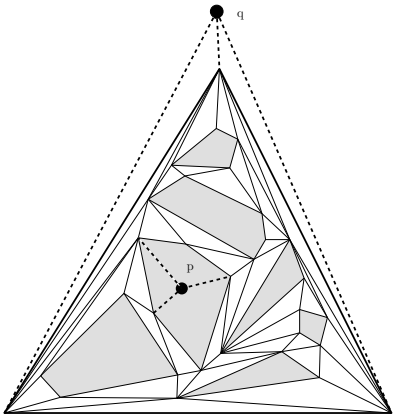


Figure 4: A triangulation of the free space around convex obstacles in a triangular container is a 3-connected graph, and it contains the desired augmentation as a subgraph.

1.3 Our Results

- For k convex obstacles, where k can be arbitrarily large, $3k-1$ edges are sometimes necessary (Section 2).
- For k convex obstacles, where k can be arbitrarily large but each obstacle has at most s sides, $3k - \frac{k-1}{s-1}$ edges are sometimes necessary (Section 2).
- For k convex obstacles, $3k$ edges are always enough (Section 3).

Once each obstacle has three vertex-disjoint paths to the container, we can get a 3-connected planar graph by adding an edge for each degree-2 vertex [2].

2 Lower Bound Constructions

When there is only one convex obstacle, three edges are obviously required (and sufficient) for connecting it to the container. However, for k (an arbitrarily-large k) convex obstacles, at least $3k-1$ edges are necessary in the worst case. Our lower bound construction is depicted in Figure 5. It includes one large convex obstacle which hides one small obstacle behind each side (except the base), such that each small obstacle can "see" only three different vertices (the top vertex of the container and two adjacent vertices of the large obstacle). Thus, we need three edges for each small obstacle and only two edges for the larger obstacle, connecting its two bottom vertices to the two endpoints of the base of the container.

The large obstacle in the above construction is a convex k-gon, and so the lower bound $3k-1$ does not hold if the every obstacle has at most s sides, for some fixed $3 \leq s < k$. In that case we use a similar construction, in which a big s-sided obstacle hides $s-1$ smaller obstacles behind all its sides except one, and the construction is repeated recursively. This construction corresponds to a complete tree with

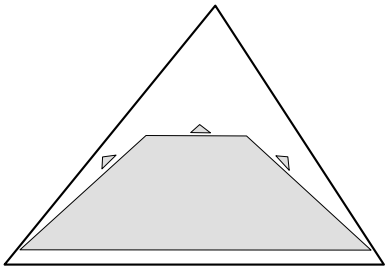


Figure 5: k convex obstacles, edges needed: $3k-1$.

branching factor $s-1$, in which the smaller obstacles are the children of a larger obstacle. For a fixed value of s, we set h as the height of the complete $(s-1)$-ary tree. Thus, the number of obstacles,

$$k = \frac{(s-1)^h - 1}{s-2}, \tag{1}$$

can be as high as we desire. The number of leaves in the tree is $(s-1)^{h-1}$. A simple manipulation of Equation 1 shows that this number equals $k - \frac{k-1}{s-1}$. Hence, the number of internal nodes in the tree is $\frac{k-1}{s-1}$. For the 3-vertex-disjoint path augmentation, each leaf obstacle needs three edges and each non-leaf obstacle needs two edges. The total number of edges required is, thus,

$$3\left(k - \frac{k-1}{s-1}\right) + 2\left(\frac{k-1}{s-1}\right) = 3k - \frac{k-1}{s-1},$$

which ranges from $\frac{5}{2}k + \frac{1}{2}$ to $3k-1$ for $3 \le s \le k$.

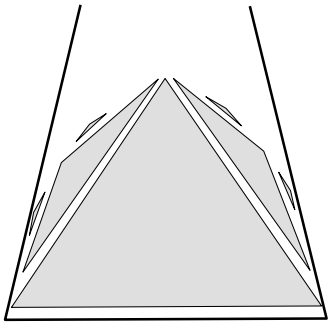


Figure 6: Construction for triangular obstacles.

3 The Upper Bound

We now prove that $3k$ edges are always sufficient for making the given set of obstacles O 3-vertex-connected to the triangular container C. Initially, there exists a triangulation T of the free space inside C that is the 3-connected, which is not always true for non-convex containers. The algorithm recurses such that each subproblem is on a polygonal container P with 3-connected triangulation (Lemmas 1 and 2).

We designate the three corners of the C with the colors red (v_R), green (v_G), and blue (v_B). Each obstacle is charged up to three times, once for each color. An obstacle is marked to indicate its connection to a particular colored corner of the container. If a path to a designated vertex goes through another obstacle, then the latter obstacle is charged for one of the edges. For each edge at least one obstacle is charged, and no obstacle is charged more than thrice, which implies that the entire process adds at most $3k$ edges. The procedure AUGMENT implements this process, which is invoked by a call AUGMENT(C, v_R, v_G, v_B).

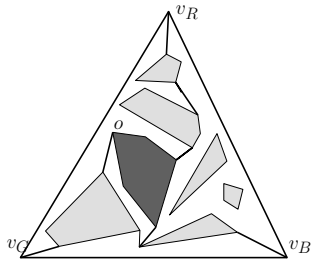


Figure 7: Vertex-disjoint paths from the obstacle o.

Algorithm 1 AUGMENT(P, v_R, v_G, v_B)

```
Pick an arbitrary obstacle o inside P.
Find three vertex-disjoint paths π_R, π_G, and π_B to
the vertices v_R, v_G, and v_B, respectively.
for all paths π_i, where i ϵ {R, G, B} do
  π_i = ShortenPath(π_i)
  for all edges e along the path π_i from o to v_i do
    Mark the obstacle incident to e for v_i
    if e is a part of some obstacle boundary then
      Go to next edge.
    else if e in incident to the boundary of P then
      Add the edge e and exit loop.
    else if e in incident to the vertex v_i then
      Add the edge e and exit loop.
    else if e is incident to an marked obstacle
    then
      Add the edge e and exit loop.
    else
      Add the edge e.
    end if
  end for
end for
HandleSubproblem(P, o, π_R, π_G)
HandleSubproblem(P, o, π_R, π_B)
HandleSubproblem(P, o, π_B, π_G)
```

Lemma 1 *For a polygon P such that every triangulation of P contains a 2-cut C_i then all the designated vertices on P are not on the same side of C_i.*

Proof. As a result of the subroutine SHORTENPATH, the polygonal boundary on the either side of any 2-cut cannot consist of only one path. Since there are always two vertex disjoint paths forming the polygonal

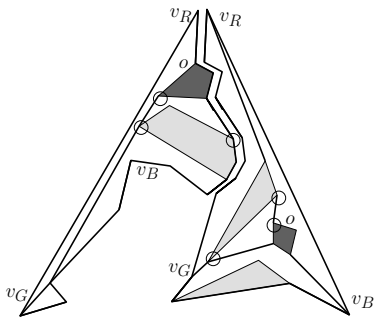


Figure 8: Recursing on the subproblems. Empty circles denote designated vertices in subproblems.

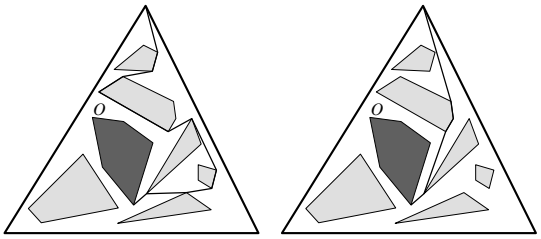


Figure 9: Shortening a vertex disjoint path.

Algorithm 2 ShortenPath(π)

Let $\{v_1, v_2, \ldots, v_m\}$ be the vertices in path π.
while for some $i < j - 1$, v_i and v_j see each other **do**
  Let P' be closed polygon formed by π and the line segment v_iv_j. Assume we are allowed to travel along π.
  Let $\pi_{i,j}$ be shortest geodesic path between v_i and v_j inside P'.
  Replace the portion of π between v_i and v_j by $\pi_{i,j}$.
  Exit loop when π stops changing.
end while
return π

boundary, there must a designated vertex or a vertex of the obstacle o present. □

Lemma 2 *Given three vertex-disjoint paths from an obstacle to v_R, v_G, and v_B, the path to v_R cannot touch the boundary of the polygon P between the vertices v_G and v_B.*

Proof. The lemma follows from the fact that the three paths are vertex disjoint. □

4 Open problems

- Close the gap between the lower and upper bounds. We conjecture that the lower bound is the correct one. Hence, give an augmentation algorithm that adds only $3k - \frac{k-1}{s-1}$ edges.

Algorithm 3 HandleSubproblem(P, o, π_i, π_j)

Obstacle o together with π_i and π_j creates a closed polygon P' inside P
Let v_i, v_j be the designated vertices of the paths π_i and π_j.
Let $l \in \{R, G, B\} \setminus \{i, j\}$.
Designate a vertex on the obstacle o as v_l.
if There is a 3-connected triangulation of P' **then**
  Augment(P', v_i, v_j, v_l)
else
  Let C_1 be the leftmost 2-cut.
  Let P_1 be the polygon created by C_1.
  Let v_R be one of the designated vertices the right of the C_1 (w.l.o.g).
  Designate the two vertices of the 2-cut as V_G and V_B
  Augment(P_1, v_R, v_G, v_B)
  HandleSubproblem($P \setminus P_1, C_1, \pi_i, \pi_j$)
end if

- Provide combinatorial bounds for 3-connectivity augmentation of *2-regular* graphs.
- Similarly, set combinatorial bounds for 3-connectivity augmentation of a set of line segments (a 1-regular graph).

Acknowledgements. We would like to thank anonymous reviewers for their useful suggestions.

References

[1] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, February 1993.

[2] Marwan Al-Jubeh, Mashhood Ishaque, Kristóf Rédei, Diane L. Souvaine, and Csaba D. Tóth. Tri-edge-connectivity augmentation for planar straight line graphs. In *ISAAC*, pages 902–912, 2009.

[3] P. O. Bex, Goos Kant, and Hans L. Bodlaender. Planar graph augmentation problems. In *WADS*, pages 286–298, 1991.

[4] Kapali P. Eswaran and Robert Endre Tarjan. Augmentation problems. *SIAM J. Comput.*, 5(4):653–665, 1976.

[5] Bill Jackson and Tibor Jordán. Independence free graphs and vertex connectivity augmentation. *J. Comb. Theory Ser. B*, 94(1):31–77, 2005.

[6] Ján Plesník. Minimum block containing a given graph. *Archiv der Mathematik*, 27(1):668–672, 1976.

[7] Ignaz Rutter and Alexander Wolff. Augmenting the connectivity of planar and geometric graphs. *Electronic Notes in Discrete Mathematics*, 31:53–56, 2008.

[8] Csaba D. Tóth and Pavel Valtr. Augmenting the edge connectivity of planar straight line graphs to three. In *XIII Spanish Meeting on Comput. Geom.*, 2009.

Blocking Coloured Point Sets*

Greg Aloupis† Brad Ballinger‡ Sébastien Collette§ Stefan Langerman¶ Attila Pór‖
David R. Wood**

1 Introduction

This paper studies problems related to visibility and blocking in sets of coloured points in the plane. A point x *blocks* two points v and w if x is in the interior of the line segment $\overline{vw}$. Let P be a finite set of points in the plane. Two points v and w are *visible* with respect to P if no point in P blocks v and w. The *visibility graph* of P has vertex set P, where two distinct points $v, w \in P$ are adjacent if and only if they are visible with respect to P. A point set B *blocks* P if $P \cap B = \emptyset$ and for all distinct $v, w \in P$ there is a point in B that blocks v and w. That is, no two points in P are visible with respect to $P \cup B$, or alternatively, P is an independent set in the visibility graph of $P \cup B$.

A set of points P is *k-blocked* if each point in P is assigned one of k colours, such that each pair of points $v, w \in P$ are visible with respect to P if and only if v and w are coloured differently. Thus v and w are assigned the same colour if and only if some other point in P blocks v and w. We say P is $\{n_1, \dots, n_k\}$-blocked if it is k-blocked and for some labelling of the colours by the integers $[k] := \{1, 2, \dots, k\}$, the i-th colour class has exactly n_i points, for each $i \in [k]$. Equivalently, P is $\{n_1, \dots, n_k\}$-blocked if the visibility graph of P is the complete k-partite graph $K(n_1, \dots, n_k)$. See Figure 1 for an example.

The following fundamental conjecture regarding k-blocked point sets is the focus of this paper.

Conjecture 1 *For each integer k there is an integer n such that every k-blocked set has at most n points.*

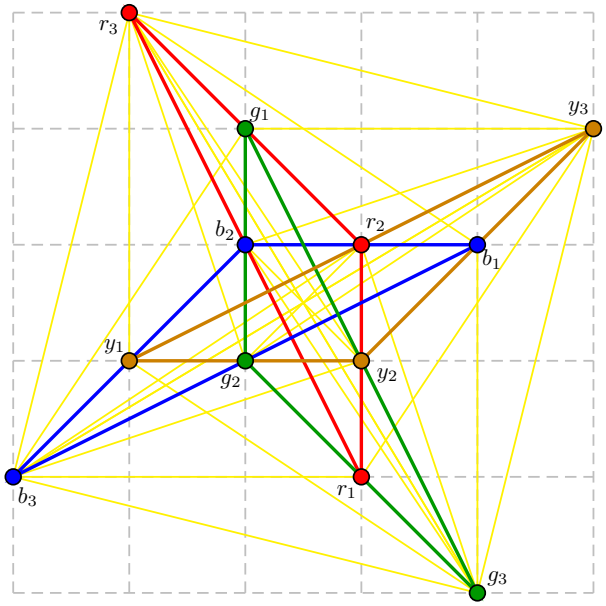


Figure 1: A $\{3, 3, 3, 3\}$-blocked point set.

A *k-set* is a multiset of k positive integers. A k-set $\{n_1, \dots, n_k\}$ is *representable* if there is an $\{n_1, \dots, n_k\}$-blocked point set. As illustrated in Figure 2, it follows from the characterisation of 2- and 3-colourable visibility graphs by Kára et al. [6] that $\{1, 1\}$ and $\{1, 2\}$ are the only representable 2-sets, and that $\{1, 1, 1\}$, $\{1, 1, 2\}$, $\{1, 2, 2\}$ and $\{2, 2, 2\}$ are the only representable 3-sets. In particular, every 2-blocked point set has at most 3 points, and every 3-blocked point set has at most 6 points. This proves Conjecture 1 for $k \leq 3$. Later we prove Conjecture 1 for $k = 4$.

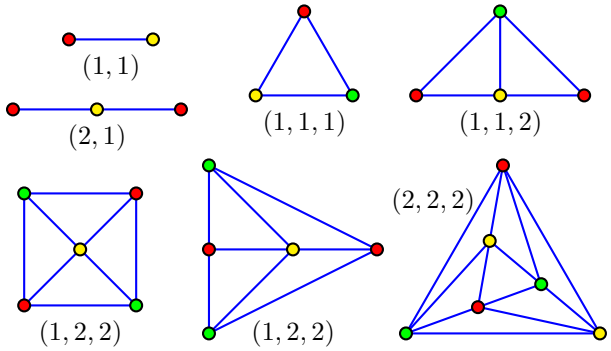


Figure 2: The 2-blocked and 3-blocked point sets.

*Initiated at the 2009 Bellairs Workshop on Computational Geometry. The authors are grateful to Godfried Toussaint and Erik Demaine for organising the workshop, and to the other participants for providing a stimulating working environment.

†Institute of Information Science, Academia Sinica, Taipei, Taiwan (aloupis.greg@gmail.com).

‡Department of Mathematics, Humboldt State University, Arcata, California, U.S.A (bjb86@humboldt.edu).

§Chargé de Recherches du F.R.S.-FNRS. Département d'Informatique, Université Libre de Bruxelles, Brussels, Belgium (sebastien.collette@ulb.ac.be).

¶Maître de Recherches du F.R.S.-FNRS, Département d'Informatique, Université Libre de Bruxelles, Brussels, Belgium (stefan.langerman@ulb.ac.be).

‖Department of Mathematics, Western Kentucky University, Bowling Green, Kentucky, U.S.A. (attila.por@wku.edu).

**Dept. of Mathematics & Statistics, The University of Melbourne, Melbourne, Australia (woodd@unimelb.edu.au). QEII Research Fellow supported by the Australian Research Council.

This paper makes the following contributions. Section 2 introduces some background motivation. Section 3 describes methods for constructing k-blocked sets from a given $(k-1)$-blocked set. These methods lead to a characterisation of representable k-sets when each colour class has at most three points. Section 4 studies the $k=4$ case in more detail. In particular, we characterise the representable 4-sets, and conclude that the example in Figure 1 is in fact the largest 4-blocked point set. Section 5 introduces a special class of k-blocked sets (so-called midpoint-blocked sets) that lead to a construction of the largest known k-blocked sets for infinitely many values of k.

Also note the following easily proved properties.

Lemma 1 ([2]) *At most three points are collinear in every k-blocked point set.*

Lemma 2 ([2]) *Each colour class in a k-blocked point set is in general position (no three collinear).*

2 Some Background Motivation

Much recent research on blockers began with the following conjecture by Kára et al. [6].

Conjecture 2 (Big-Line-Big-Clique Conjecture [6]) *For all integers t and ℓ there is an integer n such that for every finite set P of at least n points in the plane:*

- *P contains ℓ collinear points, or*
- *P contains t pairwise visible points (that is, the visibility graph of P contains a t-clique).*

Conjecture 2 is true for $t \leq 5$, but is open for $t \geq 6$ or $\ell \geq 4$; see [10, 1]. Jan Kára suggested the following weakening of Conjecture 2.

Conjecture 3 ([10]) *For all integers t and ℓ there is an integer n such that for every finite set P of at least n points in the plane:*

- *P contains ℓ collinear points, or*
- *the chromatic number of the visibility graph of P is at least t.*

Clearly Conjecture 2 implies Conjecture 3.

Proposition 3 *Conjecture 3 with $\ell = 4$ and $t = k+1$ implies Conjecture 1.*

Proof. Assume Conjecture 3 holds for $\ell = 4$ and $t = k+1$. Suppose P is a k-blocked set of at least n points. By Lemma 1, at most three points are collinear. Thus the first conclusion of Conjecture 3 does not hold. Since the visibility graph of P is k-colourable, the second conclusion of Conjecture 3 does not hold. This contradiction proves that every k-blocked set has less than n points, and Conjecture 1 holds. □

Since Conjecture 2 holds for $t \leq 5$, Conjecture 1 holds for $k \leq 4$. Let $b(n)$ be the minimum integer such that some set of n points in the plane in general position is blocked by some set of $b(n)$ points. Linear lower bounds on $b(n)$ are known [7, 3], but many authors have conjectured or stated as an open problem that $b(n)$ is super-linear.

Conjecture 4 ([7, 9, 3, 10]) $\frac{b(n)}{n} \to \infty$ *as* $n \to \infty$.

Pór and Wood [10] proved that Conjecture 4 implies Conjecture 3, and thus implies Conjecture 1. That Conjecture 1 is implied by a number of other well-known conjectures, yet remains challenging, adds to its interest.

3 k-Blocked Sets with Small Colour Classes

We now describe some methods for building blocked point sets from smaller blocked point sets.

Lemma 4 *Let G be a visibility graph. Let $i \in \{1,2,3\}$. Furthermore suppose that if $i \geq 2$ then $V(G) \neq \emptyset$, and if $i = 3$ then not all the vertices of G are collinear. Let G_i be the graph obtained from G by adding an independent set of i new vertices, each adjacent to every vertex in G. Then G_1, G_2, and G_3 are visibility graphs.*

Proof. For distinct points p and q, let $\overleftarrow{pq}$ denote the ray that is (1) contained in the line through p and q, (2) starting at p, and (3) not containing q. Let $\mathcal{L}$ be the union of the set of lines containing at least two vertices in G.

$i = 1$: Since $\mathcal{L}$ is the union of finitely many lines, there is a point $p \notin \mathcal{L}$. Thus p is visible from every vertex of G. By adding a new vertex at p, we obtain a representation of G_1 as a visibility graph.

$i = 2$: Let p be a point not in $\mathcal{L}$. Let v be a vertex of G. Each line in $\mathcal{L}$ intersects $\overleftarrow{vp}$ in at most one point. Thus $\overleftarrow{vp} \setminus \mathcal{L} \neq \emptyset$. Let q be a point in $\overleftarrow{vp} \setminus \mathcal{L}$. Thus p and q are visible from every vertex of G, but p and q are blocked by v. By adding new vertices at p and q, we obtain a representation of G_2 as a visibility graph.

$i = 3$: Let u, v, w be non-collinear vertices in G. Let p be a point not in $\mathcal{L}$ and not in the convex hull of $\{u, v, w\}$. Without loss of generality, $\overline{uv} \cap \overline{pw} \neq \emptyset$. There are infinitely many pairs of points $q \in \overleftarrow{up}$ and $r \in \overleftarrow{vp}$ such that w blocks q and r. Thus there are such q and r both not in $\mathcal{L}$. By construction, u blocks p and q, and v blocks p and r. By adding new vertices at p, q and r, we obtain a representation of G_3 as a visibility graph. □

We now characterise the representable (≥ 4)-sets, assuming each colour class has at most three points.

Proposition 5 *A k-set $\{n_1, \ldots, n_k\}$ is representable whenever $k \geq 4$ and each $n_i \leq 3$, except for $\{1,3,3,3\}$ which is not representable [2].*

Proof. We say $\{n_1, \ldots, n_k\}$ *contains* $\{n_1, \ldots, n_{i-1}, n_{i+1}, \ldots, n_k\}$ for each $i \in [k]$. We proceed by induction on k. If $\{n_1, \ldots, n_k\}$ contains a representable $(k-1)$-set, then $\{n_1, \ldots, n_k\}$ is also representable by Lemma 4. (Since $k \geq 4$ the assumptions in Lemma 4 hold.) Now assume that every $(k-1)$-set contained in $\{n_1, \ldots, n_k\}$ is not representable. By induction, we may assume that $k \leq 5$. Moreover, if $k = 5$ then $\{n_1, \ldots, n_5\}$ must contain $\{1,3,3,3\}$ (since by induction all other 4-sets are representable). Similarly, if $k = 4$ then $\{n_1, \ldots, n_4\}$ must contain $\{1,1,3\}$, $\{1,2,3\}$, $\{1,3,3\}$, $\{2,2,3\}$, $\{2,3,3\}$ or $\{3,3,3\}$ (since $\{1,1,1\}$, $\{1,1,2\}$, $\{1,2,2\}$ and $\{2,2,2\}$ are representable). The following table describes the construction in each case.

$\{1,1,1,x\}$	contains $\{1,1,1\}$
$\{1,1,2,x\}$	contains $\{1,1,2\}$
$\{1,1,3,3\}$	Figure 1 minus $\{r_1, g_3, r_3, g_1\}$
$\{1,2,2,x\}$	contains $\{1,2,2\}$
$\{1,2,3,3\}$	Figure 1 minus $\{g_1, g_3, r_3\}$
$\{2,2,2,x\}$	contains $\{2,2,2\}$
$\{2,2,3,3\}$	Figure 1 minus $\{g_3, r_3\}$
$\{2,3,3,3\}$	Figure 1 minus g_3
$\{1,1,3,3,3\}$	contains $\{1,1,3,3\}$
$\{1,2,3,3,3\}$	contains $\{1,2,3,3\}$
$\{1,3,3,3,3\}$	contains $\{3,3,3,3\}$

□

4 4-Blocked Point Sets

As shown in Section 2, Conjecture 1 holds for $k \leq 4$. That is, every 4-blocked set has bounded size. An explicit bound of 2^{790} follows from a result of Abel et al. [1], which can be improved to 2^{578} using a recent result by Dumitrescu et al. [3]; see [2]. Before characterising all representable 4-sets, we give a simple proof that every 4-blocked point set has bounded size.

Proposition 6 *Every 4-blocked set has at most 36 points.*

Proof. Let P be a 4-blocked set. Suppose that $|P| \geq 37$. Let S be the largest colour class. Thus $|S| \geq 10$. By Lemma 2, S is in general position. By a theorem of Harborth [4], some 5-point subset $K \subseteq S$ is the vertex-set of an empty convex pentagon $\text{conv}(K)$. Let $T := P \cap (\text{conv}(K) - K)$. Since $\text{conv}(K)$ is empty with respect to S, each point in T is not in S. Thus T is 3-blocked. K needs at least 8 blockers (5 blockers for the edges on the boundary of $\text{conv}(K)$, and 3 blockers for the chords of $\text{conv}(K)$). Thus $|T| \geq 8$. But every 3-blocked set has at most 6 points, which is a contradiction. Hence $|P| \leq 36$. □

Theorem 7 *A 4-set $\{a,b,c,d\}$ is representable if and only if:*

- $\{a,b,c,d\} = (4,2,2,1\}$, *or*
- $\{a,b,c,d\} = (4,2,2,2\}$, *or*
- *all of $a,b,c,d \leq 3$ except for $\{3,3,3,1\}$*

Proof Sketch. Figure 3 shows $\{4,2,2,1\}$-blocked and $\{4,2,2,2\}$-blocked point sets. When $a,b,c,d \leq 3$, the required constructions are described in Proposition 5. Now we prove the converse. Let P be a 4-blocked point set. We prove [2] that if some colour class S contains a 4-point subset K, such that $\text{conv}(K)$ is a convex quadrilateral that is empty with respect to S, then P is $\{4,2,2,1\}$-blocked. Moreover, if some colour class S has at least five points, then by Lemma 2 and a theorem of Esther Klein, S contains such a subset K—implying P is $\{4,2,2,1\}$-blocked, which is a contradiction. Hence each colour class has at most four points. Let S be a largest colour class. If S consists of four points in convex position, then P is $\{4,2,2,1\}$-blocked (just set $K := S$). If S consists of four points in nonconvex position, then we prove [2] that P is $\{4,2,2,2\}$-blocked. Otherwise $|S| \leq 3$, and we are done by Proposition 5. □

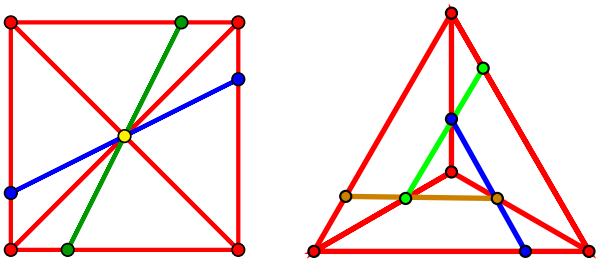


Figure 3: $\{4,2,2,1\}$-blocked and $\{4,2,2,2\}$-blocked point sets.

Corollary 8 *Every 4-blocked set has at most 12 points, and there is a 4-blocked set with 12 points.*

5 Midpoint-Blocked Point Sets

A k-blocked point set P is *k-midpoint-blocked* if for each monochromatic pair of distinct points $v, w \in P$ the midpoint of $\overline{vw}$ is in P. Of course, the midpoint of $\overline{vw}$ blocks v and w. A point set P is $\{n_1, \ldots, n_k\}$-*midpoint-blocked* if it is $\{n_1, \ldots, n_k\}$-blocked and k-midpoint-blocked. For example, the point set in Figure 1 is $\{3,3,3,3\}$-midpoint-blocked.

Another interesting example is the projection[1] of $[3]^d$. With $d = 1$ this point set is $\{2,1\}$-blocked, with $d = 2$ it is $\{4,2,2,1\}$-blocked, and with $d = 3$ it is $\{8,4,4,4,2,2,2,1\}$-blocked. In general, each set of

[1] If G is the visibility graph of some point set $P \subseteq \mathbb{R}^d$, then G is the visibility graph of some projection of P to $\mathbb{R}^2$.

points with exactly the same set of coordinates equal to 2 is a colour class, there are 2^{d-i} colour classes of points with exactly i coordinates equal to 2, and $[3]^d$ is $\{\binom{d}{i} \times 2^i : i \in [0,d]\}$-midpoint-blocked and 2^d-midpoint-blocked.

Hernández-Barrera et al. [5] defined $m(n)$ to be the minimum number of midpoints determined by some set of n points in general position in the plane, and proved that $m(n) \leq cn^{\log_2 3} = cn^{1.585\ldots}$. This upper bound was improved by Pach [8] (and later by Matousek [7]) to $m(n) \leq nc^{\sqrt{\log n}}$. Hernández-Barrera et al. [5] conjectured that $m(n)$ is super-linear, which was verified by Pach [8]; that is, $\frac{m(n)}{n} \to \infty$ as $n \to \infty$. Pór and Wood [10] proved the following more precise version: For some $c > 0$, for all $\epsilon > 0$ there is an integer $N(\epsilon)$ such that $m(n) \geq cn(\log n)^{1/(3+\epsilon)}$ for all $n \geq N(\epsilon)$.

Theorem 9 *For each k there is an integer n such that every k-midpoint-blocked set has at most n points. More precisely, there is an absolute constant c and for each $\epsilon > 0$ there is an an integer $N(\epsilon)$, such that for all k, every k-midpoint-blocked set has at most $k \max\{N(\epsilon), c^{(k-1)^{3+\epsilon}}\}$ points.*

Proof. Let P be k-midpoint-blocked set of n points. We may assume that $\frac{n}{k} > N(\epsilon)$. Let S be a set of exactly $s := \lceil \frac{n}{k} \rceil$ monochromatic points in P. Thus S is in general position by Lemma 2. And for every pair of distinct points $v, w \in S$ the midpoint of $\overline{vw}$ is in $P - S$. Thus $c\frac{n}{k}(\log \frac{n}{k})^{1/(3+\epsilon)} \leq m(s) \leq n - s \leq n(1 - \frac{1}{k})$. Hence $(\log \frac{n}{k})^{1/(3+\epsilon)} \leq (k-1)/c$, implying $n \leq k2^{((k-1)/c)^{3+\epsilon}}$. The result follows. □

We now construct k-midpoint-blocked point sets with a 'large' number of points. The method is based on the following product of point sets P and Q. Let (x_v, y_v) be the coordinates of each $v \in P \cup Q$. Let $P \times Q$ be the point set $\{(v,w) : v \in P, w \in Q\}$ where (v,w) is at (x_v, y_v, x_w, y_w) in 4-dimensional space. For brevity we do not distinguish between a point in $\mathbb{R}^4$ and its image in an occlusion-free projection of the visibility graph of $P \times Q$ into $\mathbb{R}^2$.

Lemma 10 *If P is a $\{n_1, \ldots, n_k\}$-midpoint-blocked point set and Q is a $\{m_1, \ldots, m_\ell\}$-midpoint-blocked point set, then $P \times Q$ is $\{n_i m_j : i \in [k], j \in [\ell]\}$-midpoint-blocked.*

Proof. Colour each point (v,w) in $P \times Q$ by the pair $(\mathrm{col}(v), \mathrm{col}(w))$. There are $n_i m_j$ points for the (i,j)-th colour class. It is straightforward to verify that two points in $P \times Q$ are blocked if and only if they have the same colour. Thus $P \times Q$ is blocked. Since every blocker is a midpoint, $P \times Q$ is midpoint-blocked. □

Say P is a k-midpoint blocked set of n points. By Lemma 10, the i-fold product $P^i := P \times \cdots \times P$ is a k^i-blocked set of $n^i = (k^i)^{\log_k n}$ points. Taking P to be the $\{3,3,3,3\}$-midpoint-blocked point set in Figure 1, we obtain the following result, which describes the largest known construction of k-blocked point sets.

Theorem 11 *For all k a power of 4, there is a k-blocked set of $k^{\log_4 12} = k^{1.79\ldots}$ points.*

References

[1] Zachary Abel, Brad Ballinger, Prosenjit Bose, Sébastien Collette, Vida Dujmović, Ferran Hurtado, Scott D. Kominers, Stefan Langerman, Attila Pór, and David R. Wood. Every large point set contains many collinear points or an empty pentagon. In *Proc. 21st Canadian Conference on Computational Geometry* (CCCG '09), pages 99–102, 2009. http://arxiv.org/abs/0904.0262.

[2] Greg Aloupis, Brad Ballinger, Sébastien Collette, Stefan Langerman, Attila Pór, and David R. Wood. Blocking coloured point sets, 2010. http://arxiv.org/abs/1002.0190

[3] Adrian Dumitrescu, János Pach, and Géza Tóth. A note on blocking visibility between points. *Geombinatorics*, 19(1):67–73, 2009. http://www.cs.uwm.edu/faculty/ad/blocking.pdf.

[4] Heiko Harborth. Konvexe Fünfecke in ebenen Punktmengen. *Elem. Math.*, 33(5):116–118, 1978.

[5] Antonio Hernández-Barrera, Ferran Hurtado, Jorge Urrutia, and Carlos Zamora. On the midpoints of a plane point set. *Manuscript*, 2001.

[6] Jan Kára, Attila Pór, and David R. Wood. On the chromatic number of the visibility graph of a set of points in the plane. *Discrete Comput. Geom.*, 34(3):497–506, 2005.

[7] Jiří Matoušek. Blocking visibility for points in general position. *Discrete Comput. Geom.*, 42(2):219–223, 2009.

[8] János Pach. Midpoints of segments induced by a point set. *Geombinatorics*, 13(2):98–105, 2003. http://www.math.nyu.edu/~pach/publications/midpoint.ps

[9] Rom Pinchasi. On some unrelated problems about planar arrangements of lines. In *Workshop II: Combinatorial Geometry. Combinatorics.* UCLA. 2009. http://11011110.livejournal.com/184816.html.

[10] Attila Pór and David R. Wood. On visibility and blockers. 2009. http://arxiv.org/abs/0912.1150.

Computing the depth of an arrangement of axis-aligned rectangles in parallel*

Helmut Alt† Ludmila Scharf*

Abstract

We consider the problem of computing the depth of the arrangement of n axis-aligned rectangles in the plane, which is the maximum number of rectangles containing a common point. For this problem we give a sequential $O(n \log n)$ time algorithm, and a parallel algorithm with running time $O(\log^2 n)$ in the classical PRAM model. We also describe how to adopt the parallelization to a shared memory machine model with a fixed number of processing units.

1 Introduction

In this paper we consider a basic geometric problem: how to compute the *depth* of the arrangement of n axis-aligned rectangles in $\mathbb{R}^2$. The depth of a point p is defined as the number of rectangles containing p, and the depth of the arrangement is the maximum depth over all points in $\mathbb{R}^2$, or, equivalently, it is the maximum number of rectangles that contain a common point.

We describe a parallel algorithm for this problem for a shared memory parallel machine model. The algorithm has a running time $O(\log^2 n)$ and total work $O(n \log^2 n)$ in the classical EREW-PRAM model. We also describe how to adopt the parallelization to a more realistic assumption of having a fixed number k of processing units in a shared memory machine, which fits better the modern multi-core processors.

The current trend in the microprocessor industry is to increase the performance in computing not by increasing the CPU clock rates but by multiple CPU cores working on shared memory and common cache. This trend in the hardware development makes the design of parallel algorithms once again an active topic in the algorithmic community.

In this paper we first describe a sequential $O(n \log n)$ time algorithm for computing the depth of the arrangement of axis-aligned rectangles in the plane. Namely, we construct a balanced search tree on the x-coordinates of the corners of rectangles and then perform a plane sweep along the y-axis, while updating the box coverage information in the tree.

*This research was performed in scope of the DFG-project "Parallel algorithms in computational geometry with focus on shape matching" under the contract number AL 253/7-1.

†Institute of Computer Science, Freie Universität Berlin, {alt,scharf}@mi.fu-berlin.de

To our knowledge no $O(n \log n)$ algorithm for the depth problem has been given before explicitly, although the result is somehow "folklore" knowledge in the computational geometry community. In fact, the scheme of algorithms given for solving Klee's measure problem (KMP), i.e., computing the volume of the union of n axis-parallel boxes, can be used for computing the depth of arrangements. For the two-dimensional KMP Bentley described but did not publish such an algorithm [2] the idea of which is given in [6], however. Our algorithm is similar to some extent. We develop and describe it in detail, mostly in order to derive from it in Section 3 the efficient parallel algorithm for the problem.

For the applications of the depth computation problem we mention two examples: One is a geometric pattern matching problem. For two m-point sets A and B in $\mathbb{R}^2$ finding a transformation minimizing the directed L_∞-Hausdorff distance from A to B can be reduced to finding the depth in an arrangement of $O(m^2)$ boxes. Another example is a clustering problem: For a given set of n points in $\mathbb{R}^2$ and a given radius r find a L_∞-disk of radius r containing the largest number of points, that is, the densest cluster of radius r. This clustering problem is dual to determining the deepest point in the arrangement of n boxes with side length $2r$.

For general shape (algebraic) regions, not just axis-aligned rectangles, there is no better algorithm known for computing the depth of their arrangement than to construct the complete arrangement and then to traverse it. For arrangements of disks the problem is known to be 3-SUM hard [1], so sub-quadratic algorithms are not likely to exist. For an arrangement of axis-aligned boxes in arbitrary dimension d Chan [3] describes a sequential algorithm with running time $O((n^{d/2}/\log^{d/2-1} n)\ \log^{d/2} \log n)$ for $d \geq 3$.

2 Sequential Algorithm

In this section we describe the sequential algorithm for computing the depth of the arrangement of axis-aligned rectangles.

The general idea is the following: For a given set of n axis-aligned rectangles we build a balanced binary search tree T on the x-coordinates of the vertical sides of the rectangles, so that all x-coordinates are in the leaves of the tree. Let $x_1, x_2, \ldots, x_{2n}$ be the x-coordinates of the vertical sides of the rectan-

gles in sorted order. With the leaf labeled with x_i, $i = 1, \ldots, 2n-1$, we associate the interval $[x_i, x_{i+1})$. The last leaf, labeled with x_{2n}, is associated with the interval $[x_{2n}, x_{2n}]$. With an internal node v we associate the union of the intervals of its children. Space requirement for the tree is linear in the number of rectangles.

Next we perform a top-down sweep along the y-axis. Each sweepline event, i.e., a y-coordinate of the top or bottom of a rectangle, has two x-coordinates a and b of the vertical sides of the corresponding rectangle and an event value d associated with it. The event value is $d = 1$ if it is the top of the rectangle (the rectangle is "opened") and $d = -1$ if it is the bottom (the rectangle is "closed").

To process a sweepline event we traverse the tree T from the root node to the leaves labeled a and b. In the nodes of the tree we want to count the number of rectangles covering the associated x-interval, and to update this information with each y-event. Of course, we cannot store this information directly and update it for all covered nodes for each rectangle, since that could make up to linear time per update.

Instead, we maintain in every internal node v for the current state of the sweepline in counters l and r the number of rectangles covering the interval of the left and right child of v that were opened minus the number of ones closed since the last traversal of that child. Additionally, counters l_m and r_m store the maximum value of the l and r counters, respectively, since the last traversal of the corresponding child node. Every leaf node contains counters c and c_m, which keep track of the current and maximum coverage of the associated interval during the sweep.

The information in the counters is exactly as described above when v is updated. For subsequent events that do not traverse v the information may get outdated. Thus, the counters of v store the updates that happened between the last traversal of the corresponding child node and the last traversal of v.

During each traversal of v by one of the searches the counter values are propagated from v to its child on the search path in temporary counters t and t_m, which are initially set to 0. I.e., once we updated v as described below and move to its left (right) child, t and t_m are set to the values of $v.l$ and $v.l_m$ ($v.r$ and $v.r_m$), and then $v.l$, $v.l_m$ ($v.r$, $v.r_m$) are reset to 0. Thus, when we enter the child node w the counter t is the additive change since the last update of w of the number of open rectangles that completely cover the interval of w; t_m is either the maximum value of that change between the last update of w and the current event, or 0 if the additive updates were all negative.

An update of an internal node v is performed slightly differently depending on whether both x-coordinates a and b associated with the event are contained in the subtree rooted at v (see Procedure 1: SearchBoth), or the search paths for a and b split earlier in the tree and the subtree of v contains only a (see Procedure 2: SearchLeft) or only b (procedure SearchRight).

If both a and b are contained in the subtree rooted at v we need to update the counters of v only with the values propagated from the parent node. For l and r we simply add the value of t. The max-counters (l_m and r_m) are set to the maximum of their old value, and the sum of the old counter (l or r, resp.) and t_m. If the paths to a and b split in the node v, we perform two separate searches in the left and right subtrees.

Procedure 1 : SearchBoth(v, a, b, t, t_m)

```
1  v.l_m = max(v.l_m, v.l + t_m)
2  v.r_m = max(v.r_m, v.r + t_m)
3  v.l = v.l + t
4  v.r = v.r + t
5  if a < b ≤ v.x then
6      SearchBoth (v.left, a, b, v.l, v.l_m)
7      v.l = v.l_m = 0
8  if v.x < a < b then
9      SearchBoth (v.right, a, b, v.r, v.r_m)
10     v.r = v.r_m = 0
11 if a ≤ v.x < b then
12     SearchLeft(v.left, a, v.l, v.l_m)
13     SearchRight(v.right, b, v.r, v.r_m)
14     v.l = v.l_m = v.r = v.r_m = 0
```

After the path split we know that the current rectangle spans all intervals of the right subtrees of the left search path, i.e., path to a, and all intervals of the left subtrees of the right search path, i.e., path to b. Therefore, for a node v on the left (right) search path we also add the event value d to the counter r (l). When we reach the leaves containing a and b we can update the current and maximum depth of the associated intervals.

Procedure 2 : SearchLeft(v, x_1, t, t_m)

```
1  if x_1 ≤ v.x and v is an internal node then
2      v.r_m = max(v.r_m, v.r + t_m, v.r + t + d)
3      v.r = v.r + t + d
4      SearchLeft(v.left, x_1, v.l + t,
5                 max(v.l_m, v.l + t_m))
6      v.l = v.l_m = 0
7  if x_1 > v.x then
8      v.l_m = max(v.l_m, v.l + t_m);      v.l = v.l + t
9      SearchLeft(v.right, x_1, v.r + t,
10                max(v.r_m, v.r + t_m))
11     v.r = v.r_m = 0
12 if x_1 = v.x and v is a leaf then
13     c_m = max (c_m, c + t_m, c + t + d) ; c = c + t + d
```

After all sweepline events have been processed, the depth of the arrangement is determined as the maximum of the c_m counters of the leaf nodes.

The correctness of the algorithm is based on the following observation: Once we reach the bottom of a rectangle, the counter c_m in the leaf node labeled with the x-coordinate of its left vertical boundary contains the maximum coverage of the associated x-interval between the highest y-coordinate and the y-coordinate of the bottom of the rectangle. Since, clearly, every maximally covered cell has a left vertical boundary that is a part of a left boundary of a rectangle, and thus covers at least one leaf interval, we capture at least one cell with maximum depth this way.

If we want not only to compute the depth, but also get a point with maximum depth, we can additionally store a y-coordinate for each max-counter. This y-coordinate has to be updated with the y-value of that event which results in the counter update.

The time needed to construct the tree and to sort the y-events is $O(n \log n)$. Each of the $2n$ events is processed in $O(\log n)$ time.

Theorem 1 summarizes the result of this section:

Theorem 1 *The depth of an arrangement of n axis-aligned rectangles in $\mathbb{R}^2$ can be computed in time $O(n \log n)$ with $O(n)$ additional memory.*

3 Parallel Algorithm

To enable a parallel execution of the algorithm we maintain so-called "history lists" in the nodes of the tree T. A history list of a node v contains an entry for each event of the sweepline that traverses the node v, i.e., for each y-coordinate associated with x-coordinates a and b, such that a or b is contained in the subtree rooted at v.

A history entry α of an internal node contains its "timestamp" – the y-coordinate, the corresponding x-values, the event value d and the counters l, r, l_m, r_m, as described in Section 2, and reset-flags ρ_l, ρ_r. The reset flags indicate whether the values of the left or right counters, respectively, "survive" until the next event: The value of ρ_l/ρ_r is 0 if the event of α causes the traversal of the left/right subtree, since in this case the counter values are propagated to the subtree and will be reset in the node v. Otherwise, the value is 1. Additionally, every history event has a pointer to the corresponding event, i.e., the event with the same y-coordinate, in the parent node. A history entry of a leaf node contains only its y-coordinate, the event value d, and two counters c and c_m.

Every y-event appears in at most two nodes of each level of the tree and requires constant space. Thus, the space for the tree is $O(n \log n)$.

All information of a history event, except for the counter values l, l_m, r, r_m, is known and can be set during the tree construction. Now we can "fill out" the counter values in the history lists starting with the root node down to the leaves. The left/right counters in all root node events are set to 0. For an internal node v let $\alpha^{(i)}$ be the i-th history entry of v. Let $t^{(i)}$ and $t_m^{(i)}$ denote the values of r and r_m of the corresponding history entry of the parent node of v if v is a right child of its parent node, and values of l and l_m otherwise. If $\alpha^{(i)}$ contains both x-coordinates a and b associated with $y^{(i)}$ then set

$$l^{(i)} = l^{(i-1)} \cdot \rho_l^{(i-1)} + t^{(i)} \tag{1}$$

$$l_m^{(i)} = \max\left\{ l_m^{(i-1)} \cdot \rho_l^{(i-1)},\ l^{(i-1)} \cdot \rho_l^{(i-1)} + t_m^{(i)} \right\} \tag{2}$$

$$r^{(i)} = r^{(i-1)} \cdot \rho_r^{(i-1)} + t^{(i)} \tag{3}$$

$$r_m^{(i)} = \max\left\{ r_m^{(i-1)} \rho_r^{(i-1)}, r^{(i-1)} \rho_r^{(i-1)} + t_m^{(i)} \right\}, \tag{4}$$

where the values with the high-index $(i-1)$ denote the values of the history event preceding $\alpha^{(i)}$.

Also if a history event $\alpha^{(i)}$ of a node v contains only a, and a is in the right subtree of v, or if $\alpha^{(i)}$ contains only b, and b is in the left subtree of v, the counters are set as above. That is, the values from the parent node v are propagated to the corresponding child node but the interval associated with the child is not completely covered by the rectangle causing the event, and thus, we do not need to consider the event value d.

In case $\alpha^{(i)}$ contains only a, and a is in the left subtree of v, then the complete right subtree is covered by the current rectangle. Therefore, the right counters are incremented by the event value $d^{(i)}$:

$$r^{(i)} = r^{(i-1)} \cdot \rho_r^{(i-1)} + t^{(i)} + d^{(i)} \tag{5}$$

$$r_m^{(i)} = \max\left\{ r_m^{(i-1)} \rho_r^{(i-1)}, r^{(i-1)} \rho_r^{(i-1)} + t_m^{(i)}, r^{(i)} \right\} \tag{6}$$

The left counters are updated as in equations (1), (2). In case $\alpha^{(i)}$ contains only b, and b is in the right subtree of v the left counters must be adjusted analogously and the right counters are updated as in equations (3), (4). The counters c and c_m of the leaf nodes are updated analogously.

Then, after all events have been processed, each leaf node stores the maximal coverage of its associated interval up to the position where the (last) rectangle with the corresponding vertical side was closed.

The depth of the arrangement is then the maximum over the c_m counters of the leaves.

So we could build the tree T and then traverse it level-by-level starting from the root node to the leaves, and node-by-node within one level, setting the counters in all history events. Thus, we would have a sequential algorithm with running time in $O(n \log n)$ as before but with $O(n \log n)$ memory usage.

Parallel implementation on a PRAM. For the parallel algorithm we assume that there are $O(n)$ processors on a EREW-PRAM available. Then sorting of the corner points of the rectangles once by y-coordinates and once by x-coordinates can be performed in $O(\log n)$ time, i.e., $O(n \log n)$ total work, using, for example, the sorting algorithm by Cole [4]. The tree T without the history lists can be build straightforwardly in time $O(\log n)$.

The unsorted history lists for each level of the tree can be constructed in constant time per level: We assign one processor to every history event. Every processor writes an entry for its event to the history lists of the nodes on the paths from the corresponding two leaves to the root. The entries of the history lists have correctly set timestamps (the y-coordinates), pointers to the parent entries, the event value d, and, for the internal nodes, the reset switches ρ_l, ρ_r. The counter values remain open.

The total size of the history lists in one level is at most $2n$. Therefore, the total time needed to sort the history lists of one level is $O(\log n)$. Then, for the complete tree, the construction time of the sorted history lists is $O(\log^2 n)$.

The computation of the left/right counters in the event entries is performed level-by-level starting with the root node down to the leaves. The computation of the counters $l^{(i)}$ and $r^{(i)}$ according to equations (1),(3),(5) corresponds to a prefix sum computation: Consider the left counter of the i-th history entry of a node v. Let j be the highest index $\leq i$ with $\rho_l^{(j)} = 0$. Then the value of $l^{(i)}$ is the sum $\sum_{k=j+1}^{i}(t^{(k)} + d^{(k)})$, where $d^{(k)}$ is set to 0 if the computation of $l^{(k+1)}$ follows equation (1). Thus, if we can subdivide the l-counters of a history list into subsequences corresponding to blocks of ones terminated by a zero of the ρ_l-switches, then we can in a first step set each $l^{(i)}$ to $t^{(i)}$ or $t^{(i)} + d^{(i)}$, respectively, and then perform parallel prefix sum computations on the subsequences. We omit the details about the subdivision into subsequences, which again can be performed using the parallel prefix sum computation.

The r-counter values are computed analogously. Prefix sum computation can be performed in $O(\log n)$ time [5].The total size of all prefix sum lists of one level is $O(n)$. So we need $O(\log^2 n)$ time in total.

For the computation of the max-counters according to equations (2), (4) or (6), e.g., for $r_m^{(i)}$, we need the values $r_m^{(i-1)}$, $r^{(i-1)}$, t_m, and possibly $l^{(i)}$. All of these values, except for $r_m^{(i-1)}$, are computed by now. Thus, we can compute the prefix maxima analogously to a prefix sum.

We summarize the preceding sketch of the parallel algorithm and its analysis:

Theorem 2 *The depth of an arrangement of n axis-aligned rectangles in $\mathbb{R}^2$ can be computed on a CREW-PRAM with $O(n)$ processing units in time $O(\log^2 n)$.*

Parallel implementation for a fixed number k of processors with shared memory: Sorting of the x- and y-coordinates of the vertical and horizontal sides of the rectangles can obviously be performed on a k-processor machine in time $O(\frac{n}{k}\log n)$.

Then we have to split the work performed by the algorithm between k processors. For this purpose we split the tree construction into k subtrees, each containing at most $\lceil 2n/k \rceil$ x-coordinates. Each of the subtrees is constructed by one processor sequentially. Afterwards, the k subtrees are combined into a single tree by adding a tree of height $\lceil \log k \rceil$ on top of the subtrees. The tree construction includes the history lists except for the values of the counters r, l, r_m, l_m in the internal nodes, and the counters c, c_m in the leaves.

For the computation of the counters in the history lists we apply the same idea: for the top tree we apply parallel prefix sum computation by processing the history lists in blocks of at most k elements. There are $O(n/k)$ such blocks in each level, and each block is processed in $O(\log k)$ time. Thus, the $\lceil \log k \rceil$ levels of the top tree can be processed in $O(\frac{n}{k}\log^2 k)$ time. For the k subtrees we apply the sequential algorithm to find the maximum depth in each subtree. The size of the subtrees is $O(n/k)$, thus, the time for the remaining levels is $O(\frac{n}{k}\log n)$. The total time is then $O(\frac{n}{k}(\log^2 k + \log n))$.

Summarizing, we have:

Corollary 3 *The depth of an arrangement of n axis-parallel rectangles can be computed in parallel by k processors with shared memory in time $O(n/k(\log^2 k + \log n))$.*

4 Future Work

Although the depth computation of a set of rectangles in $\mathbb{R}^2$ is an interesting problem on its own, we plan to develop parallel algorithms for higher dimensional depth computation. Further, we are interested in an implementation of the algorithm presented here, and possibly algorithms for higher dimensional problems, and in their experimental evaluation. The implementations should be performed for currently available parallel hardware platforms, such as multicore CPUs and general purpose GPUs.

References

[1] B. Aronov and S. Har-Peled. On approximating the depth and related problems. *SIAM J. Comput.*, 38(3):899–921, 2008.

[2] J. L. Bentley. Algorithms for Klee's rectangle problems. Unpublished notes, 1977.

[3] T. M. Chan. A (slightly) faster algorithm for Klee's measure problem. In *SoCG '08* , p. 94–040, 2008.

[4] R. Cole. Parallel merge sort. *SIAM J. Comput.*, 17(4):770–785, 1988.

[5] D. W. Hillis and G. L. Steele. Data parallel algorithms. *Communications of the ACM*, 1986.

[6] J. van Leeuwen and D. Wood. The measure problem for rectangular ranges in d-space. *J. Algorithms*, 2(3):282–300, 1981.

Even Triangulation of Planar Set of Points with Steiner Points

Victor Alvarez*

Abstract

Let $P \subset \mathbb{R}^2$ be a set of n points of which k are interior points. Let us call a triangulation T of P *even* if all its vertices have even degree, and *pseudo-even* if at least the k interior vertices have even degree. (Pseudo-) Even triangulations have one nice property; their vertices can be 3-colored, see [2, 3, 4]. Since one can easily check that for some sets of points, such triangulation do not exist, we show an algorithm that constructs a set S of at most $\lfloor (k+2)/3 \rfloor$ Steiner points (extra points) along with a pseudo-even triangulation T of $P \cup S = V(T)$.

1 Introduction

Let $P \subset \mathbb{R}^2$ be a set of n points. Let us for a moment suppose that along with P, we are given a parity, even or odd, for each of its n points. Given a triangulation T of P, we say that a vertex v of T is *happy* if and only if v has a degree of the parity that was originally set for v. If a vertex is not *happy* then we will say that it is *unhappy*. The problem of finding a triangulation of P that maximizes the number of *happy* vertices has recently got some attention. In [1], Aichholzer *et al.* showed that one can always find a triangulation that makes at least roughly $2n/3$ vertices happy, and they also showed a configuration of points and parities that will make at least $n/108$ vertices *unhappy*, regardless of the chosen triangulation.

In this paper we attack a problem with the same spirit, however, we use a different paradigm to solve it since the result of Aichholzer *et al.* does not ensure in general a solution. Let $P \subset \mathbb{R}^2$ be as before and assume that $0 \leq k \leq n-3$ points are inside the convex hull $Conv(P)$ of P, *i.e.* there are k interior points. In our setting, only those k interior points will have a parity assigned and it will be the same for each one of them, namely, even. Now, we look for a triangulation that makes *all* those k interior vertices *happy*. We will call such triangulations *pseudo-even*, or simply *even* in the case that also the vertices of $Conv(P)$ happen to have even degree. It is already known that a maximal planar graph is 3-colorable if and only if it is at least pseudo-even, see [4] for this characterization and [2, 3] for a general reference on 3-colorable planar graphs. So pseudo-even triangulations have at least one interesting property and we can also see this problem as that of embedding 3-colorable planar graphs on set of points. As one can easily check that for some sets of points, a pseudo-even triangulation do not exist, we will introduce extra points, also known as *Steiner points*, and then we will consider the questions: how many Steiner points are sufficient and how many are necessary to get a pseudo-even triangulation T such that $P \subseteq V(T)$? While we still have no answer for the latter question, we will present a non-trivial solution for the former, namely, we will show an algorithm with the following properties:

(i) Its output triangulation T is pseudo-even and $V(T) = P \cup S$.

(ii) $|S| \leq \lfloor (k+2)/3 \rfloor$.

(iii) At most two Steiner points of S fall on $Conv(P)$.

Note that, as T is a pseudo-even triangulation, the Steiner points of S that are interior must also get even degree.

This paper is divided as follows: in Section 2 we show our construction and in Section 3 we close with some interesting observations.

2 Points in general position

Let us quickly recall that given a polygon $\mathcal{P}$, a vertex of $\mathcal{P}$ is called *reflex* if the internal angle is larger than 180 degrees and we will call it convex otherwise.

The main result of this section is the following:

Theorem 1 *Let $P \subset \mathbb{R}^2$ be a set of n points such that k of those points are interior points. Then we can always obtain a pseudo-even triangulation adding at most $\lfloor (k+2)/3 \rfloor$ Steiner points to P, of which at most two fall on $Conv(P)$.*

Before showing the actual construction let us give the general idea. As it was pointed out in the introduction, we can talk about 3-colorable maximal planar graphs and pseudo-even triangulation unambiguously. Therefore, our idea to get a pseudo-even triangulation is to actually embed a 3-colorable maximal planar graph on P with the help of at most $\lfloor (k+2)/3 \rfloor$ Steiner points. So we will use a 3-coloration as a measure of the correctness of our algorithm. Having defined what we will actually aim at, let us start with our construction.

*Universität des Saarlandes, Saarbrücken, Germany. alvarez@cs.uni-sb.de. Supported by Graduiertenkolleg of the Deutsche Forschung Gemeinschaft (DFG) of Germany and partially supported by CONACYT-DAAD of México.

Proof. Let us fix a vertex $v \in Conv(P)$ such that v has the lowest y-coordinate among all points in P. Using v as a pivot, we will sort each interior point of P by its slope with respect to v. Let $p_1, \ldots, p_k$, be a labeling, from left to right with respect to this angular order, of the internal points of P. Let p_0, p_{k+1} be the left and right neighbors of v on $Conv(P)$ respectively.

We construct a simple polygon $\mathcal{P}$ from $P \setminus \{v\}$ as follows: we add each edge $p_i p_{i+1}$, for $0 \leq i \leq k$. We call this the lower part of $\mathcal{P}$ and we will denote it by $L(\mathcal{P})$. Also, we consider the edges of $Conv(P) \setminus \{p_0 v, p_{k+1} v\}$ and we call this the upper part of $\mathcal{P}$ and we will denote it by $U(\mathcal{P})$.

Next we will triangulate $\mathcal{P}$ as follows: we will scan $L(\mathcal{P})$ from left to right and we will consider the largest chains formed by convex vertices. Note that for each chain, the left and right endpoints must be reflex vertices of $\mathcal{P}$, see to the left in Figure 1. Now, for each chain, we will make adjacent its two endpoints and we will use its lowest convex vertex as a pivot to triangulate the resulting convex polygon in case that it has more than three vertices. These convex polygons can be thought as "ears" that can be cut from $\mathcal{P}$ on $L(\mathcal{P})$. The rest of $\mathcal{P}$, outside these "ears", can be triangulated in any way. See to the right in Figure 1. If there is no convex vertex of $\mathcal{P}$ in $L(\mathcal{P})$, then the triangulation of $\mathcal{P}$ is arbitrary.

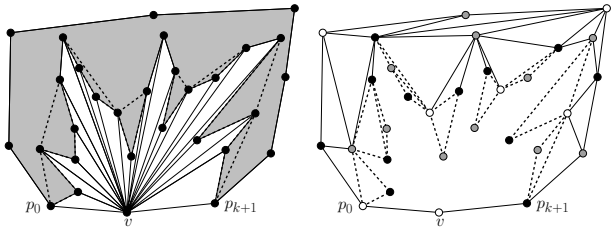


Figure 1: To the left we have the polygon $\mathcal{P}$ on $n-1$ vertices in light gray. The convex polygons formed by scanning $L(\mathcal{P})$ from left to right are shown in dashed. Note that each pair of consecutive convex polygons shares at most one vertex. To the right we see a triangulation $T(\mathcal{P})$ of $\mathcal{P}$. The dashed edges are the only ones that are not arbitrary.

Let $T(\mathcal{P})$ be the aforementioned triangulation of $\mathcal{P}$. We know that we can 3-color it, see for example [5], and note that the only point yet to be colored is v. We will show how to color v while keeping a 3-coloration of $T(\mathcal{P})$ by using Steiner points.

From this point on, our construction is done by case analysis. Note that as $T(\mathcal{P})$ is already 3-colored, if all the interior vertices of P are colored by only two colors, say $i+1, i+2$, $1 \leq i \leq 3$[1], we could use color i for v without violating the 3-coloration of $T(\mathcal{P})$, and hence, using the straight-line segments that connect v with each vertex of $L(\mathcal{P})$, we obtain a pseudo-even triangulation $T(P)$.

However, in general it is not going to happen that the interior vertices can be colored using only two colors, hence we need to do something else in such cases. We will proceed in a line-sweep fashion from p_0 to p_{k+1} with respect to the angular order given by v.

Let us fix the color of v as the color of the smallest chromatic class in the lower part $L(\mathcal{P})$ of $\mathcal{P}$ using the 3-coloration of $T(\mathcal{P})$, say that color is i without loss of generality, $1 \leq i \leq 3$. Note that the points in $L(\mathcal{P})$ with color i are the ones causing trouble to complete the desired triangulation, hence we will handle those points depending on their kind in $\mathcal{P}$, namely if they are reflex or convex vertices of $\mathcal{P}$. We will keep the invariant that, by the time we are processing an interior point p_j, *all* interior points to the left have already even degree. Also note that by this time, if the degree of p_j is odd it is because p_{j+1} has color i, otherwise we could join v and p_{j+1} and hence the conflict is somewhere to the left or even in $\mathcal{P}$, which would contradict our invariant or the valid 3-coloration of $\mathcal{P}$.

Let us start now with our case analysis, we will assume that we are currently processing the interior point p_j, $1 \leq j < k$.

(1) Point p_j of color, say $i+1$, and p_{j+1} of color i is a reflex vertex. Note that if p_{j+2} has color $i+2$, then we could introduce the edge $p_j p_{j+2}$, as p_{j+1} has already even degree in $T(\mathcal{P})$. Hence we will assume that p_j and p_{j+2} have the same color.

As p_{j+2} is of color $i+1$, then we need to complete the degree of both p_j and p_{j+1} as both degrees are odd. Here we will introduce one Steiner point s of color $i+2$ that will be adjacent, without introducing any crossing, to p_j, p_{j+1}, p_{j+2}, v, hence of even degree and we will add the straight-line segment between p_{j+2} and v, move to p_{j+2} and continue. See to the left in Figure 2

(2) Point p_j again of color $i+1$ and p_{j+1} of color i is a convex vertex. Again see that if p_{j+2} is of color $i+1$, as before, we introduce one Steiner point s of color $i+2$, and exatly the same set of adjacencies as in the case when p_{j+1} is reflex. See in the middle in Figure 2.

So let us assume that p_{j+2} has color $i+2$. Note that, as p_{j+1} is a convex vertex of $\mathcal{P}$, it must be part of one of the convex polygons we first got when $\mathcal{P}$ got triangulated, after all remember that the triangulation $T(\mathcal{P})$ is in general not arbitrary. We have now the following sub-cases:

(2.1) The vertex p_{j+1} was used as a pivot in the triangulation of $\mathcal{P}$. Consider the convex chain C formed by the points $p_j, p_{j+1}, p_{j+2}, \ldots, p_l$, where p_l is a reflex vertex. See to the right in Figure 2. Note as well that since p_{j+1} was used as a pivot, all the edges

[1] Arithmetic taken modulo three

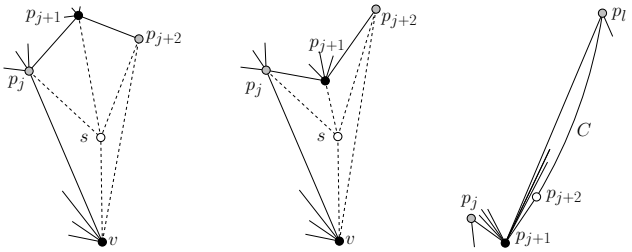


Figure 2: The point p_j is currently being processed. Point p_{j+1} is of the same color i of v. If p_j and p_{j+2} have the same color, then one Steiner point suffices to be able to move to p_{j+2}. To the right, p_j and p_{j+2} have different colors and p_{j+1} is a convex vertex that was used as a pivot to triangulate the convex polygon it is part of in $\mathcal{P}$.

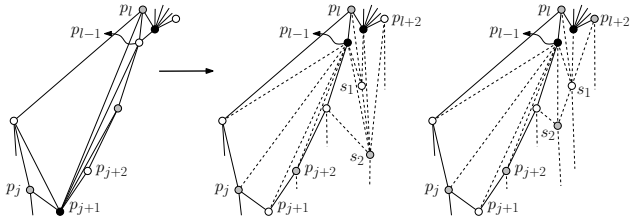


Figure 3: If p_{j+1} was used as a pivot to triangulate a convex polygon that can be cut from $\mathcal{P}$, then we can use p_{l-1} as the new pivot without changing the color of p_j or anything to its left. Note that p_l must be necessarily a reflex vertex of $\mathcal{P}$. In the middle we see the final configuration in the case that p_{l+1} is of color i and p_{l+2} is of color $i+2$. To the right we see the final configuration when p_{l+1} is of color i and p_{l+2} is of color $i+1$.

$p_{j+1}p_{j+2}, \ldots, p_{j+1}p_l$ are present.

Now we distinguish between the following cases:

(2.1.1) Point p_l is of color $i+1$, p_{l+1} is of color i and p_{l+2} is of color $i+2$.

We know that the union of all the triangles that share p_{j+1} as a vertex forms a convex polygon $\mathcal{C}$. We will change all the adjacencies inside $\mathcal{C}$ as follows:

Instead of taking p_{j+1} as the pivot that is adjacent to all vertices in $\mathcal{C}$ we will take p_{l-1}. Now we recolor p_{l-1} with color i and we will change the color of $p_{j+1}, p_{j+2}, \ldots, p_{l-1}$ to $i+2, i+1, \ldots, i+2$ respectively. Note that no other color needs to be changed.

Finally we will introduce two Steiner points s_1, s_2 of color $i+2, i+1$ respectively and we will make the following adjacencies:

(i) s_1 gets adjacent to p_{l-1}, p_l, p_{l+1} and s_2.

(ii) s_2 gets adjacent to $p_{l-2}, p_{l-1}, s_1, p_{l+1}, p_{l+2}, v$.

Additionally we introduce the edges $p_{j+1}v, \ldots, p_{l-2}v$ and $p_{l+2}v$. See to the left and in the middle of Figure 3.

Look that the previous construction can always be done without introducing any crossing. Moreover, note that with two Steiner points we complete the even degree of each point in the region $p_j, \ldots, p_{l+2}$ in which there were originally two points of color i. Thus we can move to p_{l+2} and continue.

(2.1.2) Point p_l and p_{l+1} as before and p_{l+2} is of color $i+1$.

We will proceed as before except that this time, the adjacencies of s_1, s_2 are as follows:

(i) s_1 gets adjacent to $s_2, p_{l-1}, p_l, p_{l+1}, p_{l+2}$ and v.

(ii) s_2 gets adjacent to p_{l-2}, p_{l-1}, s_1 and v.

As before, we also introduce the adjacencies $p_{j+1}v, \ldots, p_{l-2}v$ and $p_{l+2}v$. Again, every even degree is now completed and we can move to p_{l+2}. See to the right in Figure 3 for the final configuration.

(2.1.3) Point p_l as before and p_{l+1} is of color $i+2$.

Note that in this case, from p_j to p_{l+1} we are in presence of only one vertex of color i, namely p_{j+1}, thus we will introduce only one Steiner point s_1.

We will proceed as before with $\mathcal{C}$ and note that this time p_{l-1} and p_{l+1} have different colors, namely i and $i+2$ respectively. Hence the degree of p_l is already even and since p_l is a reflex vertex of $\mathcal{P}$, we can introduce the adjacency $p_{l-1}p_{l+1}$. Now we make s_1 adjacent to $p_{l-2}, p_{l-1}, p_{l+1}$ and v and finally we introduce the adjacencies $p_{j+1}v, \ldots, p_{l-2}v$ and $p_{l+1}v$.

Note that again each even degree in $p_j, \ldots, p_{l+1}$ is completed and hence we can move to p_{l+1} and continue. See to the left in Figure 4 for the final configuration.

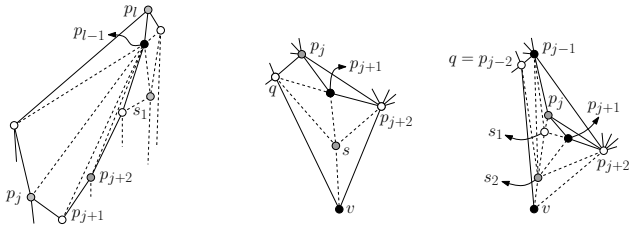


Figure 4: To the left we see the final configuration in the case that p_{j+1} was a pivot of color i and p_{l+1} is of color $i+2$. In the middle and to the right we have that, if p_{j+1} of color i was not a pivot and its neighbors have different color from each other, then one of them must necessarily be a pivot, in this case p_{j+2}. So we have to go back and remove some adjacencies that will allow us to introduce the Steiner points appropriately.

Note that the following three cases are also possible:

(2.1.4) Point p_l is of color $i+2$, p_{l+1} is of color i and p_{l+2} is of color $i+1$.

(2.1.5) Point p_l and p_{l+1} as before and p_{l+2} is of color $i+2$.

(2.1.6) Point p_l as before and p_{l+1} is of color $i+1$.

However, those cases are essentially the same as

the ones explained, so we would proceed in exactly the same way but we will exchange the color of the Steiner points we are introducing. The details are left for the reader.

(2.2) In this case p_{j+1} of color i was not used as a pivot and it just takes part in a convex polygon where the pivot p_{j+2} is of color $i+2$. This restriction in colors arises from the fact that we are assuming that the triangle $p_j, p_{j+1}, p_{j+2} \in T(\mathcal{P})$ is well-colored, as explained in the beginning of case **(2)**.

Since the edge $p_j v$ is currently in the triangulation being built, there is one triangle t using it. Let $q \notin \{p_j, v\}$ be the third vertex of such a triangle t. Note that q lies to the left of the edge $p_j v$ and hence it already has even degree, moreover, the color of q is $i+2$. Now we have the following two cases:

(2.2.1) The vertex q is a Steiner point or the quadrilateral $Q = q, p_j, p_{j+1}, v$ is convex. Let us consider only the case that Q is convex, if it is not the case then q is a Steiner point and it can be moved as pleased to make Q convex without affecting anything. Thus we will flip the edge $p_j v$ for the edge $q p_{j+1}$ and introduce one Steiner point s of color $i+1$ inside Q with its incidences to the vertices of Q, see in the middle of Figure 4.

(2.2.2) If q is not a Steiner point and Q is non-convex, then it is not hard to see that the only possible case is $q = p_{j-2}$, and p_{j-1} is a reflex vertex of $\mathcal{P}$ of color i. Note then that the edge $e = p_{j-2}p_j$ must be present in the triangulation and that p_{j-1} is adjacent to no Steiner point. Hence we will remove e and we will introduce one Steiner point s_1 of color $i+2$ that is adjacent to $p_{j-1}, p_j, p_{j+1}, s_2$, where s_2 is another new Steiner point of color $i+1$ that is adjacent to $p_{j-2}, p_{j-1}, s_1, p_{j+1}, p_{j+2}, v$. We can now move to p_{j+2} and continue. See to the right in Figure 4.

Note that the color i of v was chosen as the color of the smallest chromatic class in $L(\mathcal{P})$ and note that its cardinality can be at most $\lfloor (k+2)/3 \rfloor$. Also note that in our analysis, we assumed that the current point p_j that we are processing is neither p_0 nor p_{k+1} of $Conv(P)$. So in the case that those two extreme vertices are of color i we will introduce two Steiner points of color different that i that will subdivide the edges of $Conv(P)$ that connect p_0 and p_{k+1} with v, and hence removing any possible conflict at that stage. As we introduce one Steiner point per element of the smallest chromatic class in $L(\mathcal{P})$ the total number of Steiner points is $\lfloor (k+2)/3 \rfloor$ and the result follows.

□

3 Conclusions and Discussion

We have presented an algorithm that produces a pseudo-even triangulation adding at most $\lfloor (k+2)/3 \rfloor$ Steiner points to a given point set $P \subset \mathbb{R}^2$. It is important to note that at most two Steiner points lie on $Conv(P)$ and hence our construction keeps many extent measures of P, *i.e.* diameter, width, etc. If we do not care about modifying $Conv(P)$ or the position of the Steiner points, then *only two* Steiner points far away from $Conv(P)$ would do the job, say one at ∞ and the other at $-\infty$. Albeit being this construction possible, we do not know why it would be interesting to use it, since the output set of points does not look anything like the one that was given as the input.

For the sake of completeness it is also interesting to discuss what happens when P is in convex position and we look this time for an *even* triangulation. In [4] it was proven that if T is an even triangulation, then $|Conv(V(T))| \equiv 0 \mod 3$. The other direction can be easily proven by induction, so we do not see the necessity of writing down the details. Hence, given P in convex position, we can obtain an even triangulation T adding at most two Steiner points such that $V(T)$ remains in convex position.

We are aware that our technique could be push further to obtain a smaller number of Steiner points, probably $\lfloor (k+2)/6 \rfloor$ might be doable and we already started working out the details. Nevertheless, what it has been rather frustrating is the fact that we have not been able to come up with a lower bound on the number of Steiner points and actually, everything points to the fact that really few Steiner points might suffice, this number might even be *constant*! Finding a *simple* algorithm that uses fewer Steiner points, and finding a lower bound for this number seem interesting and challenging.

Acknowledgement

We thank Marco Heredia and Jorge Urrutia for having suggested the even triangulation problem in first place and for valuable discussions. We also thank Raimund Seidel for valuable comments.

References

[1] O. Aichholzer, T. Hackl, M. Hoffmann, A. Pilz, G. Rote, B. Speckmann and B. Vogtenhuber. *Plane Graphs with Parity Constraints*. WADS '09: Proceedings of the 11th International Symposium on Algorithms and Data Structures, (2009), 13–24.

[2] P. J. Heawood, *On the Four-color Map Theorem*. Quart. J. Pure Math. **29**, (1898), 270–285.

[3] R. Steinberg, *The State of the Three-color Problem*. Quo Vadis, Graph Theory?, Annals of Discrete Mathematics, **55** (1993), 211–248.

[4] K,. Diks L. Kowalik M. Kurowski, *A New 3-color Criterion for Planar Graphs*. LNCS **2573** (2002), 138–149.

[5] S. Fisk, *A Short Proof of Chvátal's Watchman Theorem*. J. Combinatorial Theory, Series B. **18**, (1978), 374.

Separability of Point Sets by k-Level Linear Classification Trees

Esther M. Arkin* Delia Garijo† Alberto Márquez† Joseph S. B. Mitchell* Carlos Seara‡

Abstract

Let R and B be sets of red and blue points in the plane in general position. We study the problem of computing a k-level binary space partition (BSP) tree to classify/separate R and B, such that the tree defines a linear decision at each internal node and each leaf of the tree corresponds to a (convex) cell of the partition that contains only red or only blue points.

1 Introduction

Consider a set of n points in the plane in general position. Each point is either "red" or "blue". We let R denote the set of red points and let B denote the set of blue points. We study the separability of R and B by a k-level binary space partition tree. We say that R and B are *separated* by a k-level binary space partition tree, T, if each region in the partition of the plane induced by T is monochromatic (contains only points of R or only points of B). The separating k-level tree T is a recursive partition of the plane into monochromatic and disjoint convex regions using (up to) $2^k - 1$ separating straight cuts (lines, rays or segments). Such a tree T of height k (i.e., with k levels) can be used as a classification tree for red/blue points; we can classify, in time $O(k)$, a new point as "red" or "blue" based on the color associated with the cell (corresponding to a leaf in the tree) in which it is located. See Figure 1.

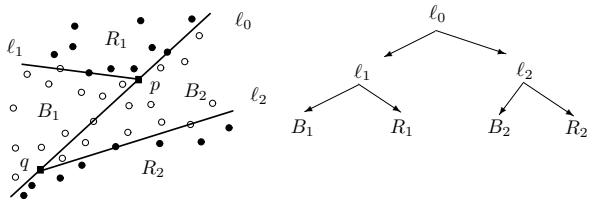


Figure 1: A separating 2-level tree.

*Applied Mathematics and Statistics, State University of New York, Stony Brook, NY, 11794-3600, USA, {estie,jsbm}@ams.sunysb.edu

†Departamento de Matemática Aplicada I, Universidad de Sevilla, Avda. Reina Mercedes s/n, 41012 Sevilla, Spain, {dgarijo,almar}@us.es

‡Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, Jordi Girona 1, 08034 Barcelona, Spain, carlos.seara@upc.edu

Related work. The separating k-level tree generalizes simple separability criteria that have been previously studied. The most basic separability criteria for R and B is that of linear separability, which corresponds to a separating 1-level tree: there exists a line separating R and B. Linear separability can be decided in linear time [8]. For sets R and B that are not linearly separable, generalizations include the following separability criteria: a strip (two parallel lines, partitioning the plane into three regions), a wedge (two rays with common origin, partitioning the plane into two regions), a double wedge (two intersecting lines), or three parallel lines. All of these criteria can be decided, and corresponding partitions computed, in optimal $\Theta(n \log n)$ time [1, 2, 6, 7]. (Note that if R and B are strip separable, then they are also wedge separable.) Strip, wedge, double-wedge, or three parallel lines separability criteria are special cases of separability by a 2-level tree.

Separability by multiple parallel lines is a special case of separability by a k-level tree; in particular, $m = 2^k - 1$ parallel lines can be a associated with a (height-balanced) k-level tree. The minimum number of parallel lines needed to separate R and B can be computed in $O(n^2 \log n)$ time [2]. If R and B are the vertices of a regular n-gon, $\lfloor n/2 \rfloor$ is a tight upper bound for the number of parallel lines, and, given the minimum number of separating lines, their common orientation can be computed in $O(n \log n)$ time [3].

Other separability criteria have also been studied. Given any disjoint point sets, R and B, there always exists a separating polygonal chain, which can be computed in $O(n \log n)$ time. Computing a minimum-link separating polygonal chain that turns alternatively left and right by a constant angle $\alpha \geq \pi/2$ can be done in $O(n \log n)$ time [6]. Separability by m parallel lines is a special case of separability by a monotone m-link polygonal chain. The problem of determining a minimum-link separating polygonal chain of R and B is NP-complete [5]. Edelsbrunner and Preparata [4] solved, in time $O(n \log n)$, the special case of computing a minimum-edge convex polygon separating R and B (if a convex separator exists); their time bound was shown to be optimal in [1].

Outline of the paper. We initiate the study of separability by k-level trees by considering first the special case of $k = 2$, separability by a 2-level tree. Section 2 is devoted to a special case of 2-level separability, that of separability by a *zigzag*, which corresponds

to 2-level tree partitioining such that monochromatic cells of the same color are adjacent (Figure 2). In Section 3 we study the general version of 2-level tree separability, including the generalizations to three or four distinct colors of point sets (instead of just two, red and blue). In Section 4 we consider k-level tree separability and possible configurations of points with $O(\log n)$-level trees. Section 5 is devoted to separability by k-level trees whose partitioning cuts are axis-parallel.

2 Zigzag Separability

In this section we consider the zigzag separability problem: Determine whether the sets R and B are separable by a *zigzag* $Z = (\ell_1, s, \ell_2)$, which is a simple, nonconvex 3-link polygonal chain formed by two rays ℓ_1, ℓ_2 and a segment s joining the origins of the rays (Figure 2). Let ℓ_s be the line containing the segment s, and let ℓ_1' (ℓ_2') be the line containing the ray ℓ_1 (ℓ_2). We can assume that the simpler known special cases of separability have already been tested; specifically, we assume that R and B are not separable by a line, strip, wedge, or convex polygonal chain, each which can be decided in $O(n \log n)$ time. Thus, under this condition, the following lemma is straightforward, where $CH(X)$ denotes the convex hull of a point set X.

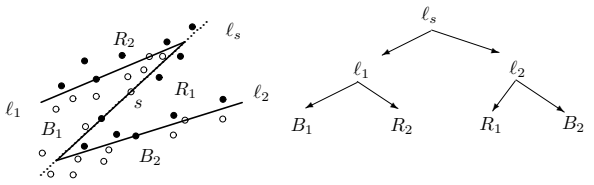


Figure 2: A separating zigzag.

Lemma 1 *Let R and B be zigzag separable but not separable by a line, strip, wedge, or convex polygon. Then, $CH(R)$ contains at least one blue point, and $CH(B)$ contains at least one red point.*

A separating zigzag $Z = (\ell_1, s, \ell_2)$ defines four wedges that partition $R \cup B$ into four subsets, denoted by R_1, R_2, B_1, and B_2, where $R_2 = R - R_1$ and $B_2 = B - B_1$; all four subsets are non-empty, since R and B are not wedge separable. Let α (β) be the angle defined by ℓ_s and ℓ_1 (ℓ_2). Two optimal separating zigzags are considered: either a zigzag maximizing $\min\{\alpha, \beta\}$, called *the most convex separating zigzag* (approximating linear separability), or a zigzag that minimizes $\max\{\alpha, \beta\}$ (approximating separability by three parallel lines).

Lemma 2 *Let $Z = (\ell_1, s, \ell_2)$ be the most convex separating zigzag for R and B. Then each of the two rays, and the segment of Z pass through two points of different colors. Moreover, either ℓ_1' is an interior supporting line of $CH(R_2)$ and $CH(B)$, or ℓ_2' is an interior supporting line of $CH(B_2)$ and $CH(R)$.*

Lemma 3 *Let R and B be zigzag separable and let $I_{B,R}$ be the number of intersections between pairs of edges of $CH(B)$ and $CH(R)$. Then $I_{B,R} \in \{0, 2, 4, 6\}$.*

Let R_I (B_I) be the subset of red (blue) interior points of $CH(B)$ ($CH(R)$). By Lemma 1, $|R_I| \geq 1$ and $|B_I| \geq 1$. If $I_{B,R} = 6$, let R_1', R_2', and R_3' (B_1', B_2', and B_3') be the three disjoint subsets of red (blue) exterior points of $CH(B)$ ($CH(R)$). These eight subsets and their respective convex hulls can be computed in $O(n \log n)$ time (Figure 3).

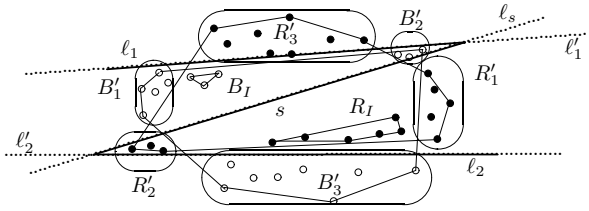


Figure 3: Subsets of red and blue points for $I_{B,R} = 6$.

Lemma 4 *Let $Z = (\ell_1, s, \ell_2)$ be the most convex separating zigzag of R and B. Then ℓ_s is a supporting line of some of the following eight convex polygons: $CH(R_I)$, $CH(B_I)$, $CH(R_1')$, $CH(R_2')$, $CH(R_3')$, $CH(B_1')$, $CH(B_2')$, and $CH(B_3')$.*

Lemma 4 provides the key tool to design the following $O(n \log n)$ time algorithm for computing a separating zigzag $Z = (\ell_1, s, \ell_2)$ for R and B (if it exists).

ZIGZAG-ALGORITHM

Input: R and B

Output: a separating zigzag $Z = (\ell_1, s, \ell_2)$, or report that none exists

1. Compute $CH(R)$, $CH(B)$, R_I, B_I, $CH(R_I)$, $CH(B_I)$, and $I_{B,R}$. Check whether $I_{B,R} \in \{0, 2, 4, 6\}$, and compute the intersecting edges of $CH(R)$ and $CH(B)$. Check that $CH(R_I)$ or $CH(B_I)$ are monochromatic. For $R_I = \{r\}$ and $B_I = \{b\}$, do as follows: If $r \in CH(R)$ and $b \in CH(B)$, then R and B are zigzag separable and it is easy to see how to compute the separating zigzag. Analogously if $r \in CH(R)$ and b is interior to $CH(B)$ or vice versa. From now on assume that $|R_I| \geq 2$ or $|B_I| \geq 2$.

2. Let P be any of the polygons: $CH(R_I)$, $CH(B_I)$, $CH(R_1')$, $CH(R_2')$, $CH(R_3')$, $CH(B_1')$, $CH(B_2')$,

or $CH(B_3')$, with their interior points. Do the following:

(a) Sort the points in $(R \cup B) - P$ by a counterclockwise rotational sweep over P with an oriented supporting line ℓ_s according to Lemma 4.

(b) Do a second rotational sweep over P. Each time ℓ_s hits a red or blue point of $(R \cup B) - P$, maintain and update the convex hulls $CH(R_2)$, $CH(B_1)$ ($CH(R_1)$, $CH(B_2)$) of the red and blue points on the left (right) side of ℓ_s in $O(\log n)$ time [9]. In $O(\log n)$ time, check the linear separability between $CH(R_2)$ and $CH(B_1)$, and between $CH(R_1)$ and $CH(B_2)$, and compute their respective supporting lines (Figure 4). In the affirmative case, a separating zigzag is found.

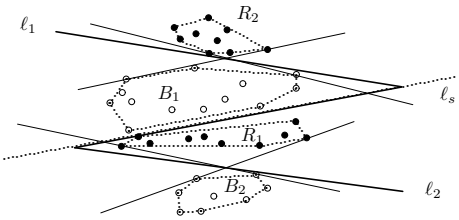


Figure 4: Supporting lines between monochromatic convex hulls.

To prove the $\Omega(n \log n)$ time lower bound for deciding the zigzag separability, we reduce the strip separability problem [1] to the zigzag separability problem.

Theorem 5 *Computing a separating zigzag for R and B requires $\Theta(n \log n)$ time.*

3 Separability by a 2-Level Tree

We turn now to the problem of computing a separating 2-level tree $T = (\ell_1, \ell_0, \ell_2)$ for R and B, where ℓ_0, ℓ_1, and ℓ_2 are the oriented line, the ray on the left side of ℓ_0, and the ray on the right side of ℓ_0, respectively (recall Figure 1). Let ℓ_1' (ℓ_2') be the line containing ℓ_1 (ℓ_2). Denote by $m(\ell)$ the slope of ℓ. Let p (q) be the intersection point of ℓ_0 and ℓ_1 (ℓ_2). T splits the plane into four convex regions.

Criteria. The following criteria provide a systematic classification of the separating 2-level trees: (1) $m(\ell_0) > 0$, $m(\ell_0) < 0$, or ℓ_0 is horizontal or vertical. (2) Relative position of p and q along ℓ_0: $p \preceq q$ or $q \preceq p$. (3) Slopes of ℓ_1 and ℓ_2 with respect to ℓ_0. (4) Different color assignments to the convex regions.

Classification. We reduce to the following cases: (1) Slope of ℓ_0: we only consider the $m(\ell_0) \geq 0$ case. The configuration of points where $m(\ell_0) < 0$ can be analyzed by rotating this configuration by 90 degrees and applying the $m(\ell_0) > 0$ case (Figure 5); the case ℓ_0 vertical is symmetric to the ℓ_0 horizontal case, by a 90-degree rotation. (2) Relative position of p and q: we only study the case $q \preceq p$. By applying symmetry with respect to a vertical line, followed by a 90-degree rotation, we get the case $p \preceq q$ (Figure 5). (3) If two consecutive regions have the same color, it corresponds to some of the following criteria: linear, zigzag ($p \neq q$), or wedge separability ($p = q$) which can be solved in $\Theta(n \log n)$ time [1, 6, 7]. Thus, we assume that the colors alternate, ℓ_0 has positive slope or is horizontal, and $q \preceq p$.

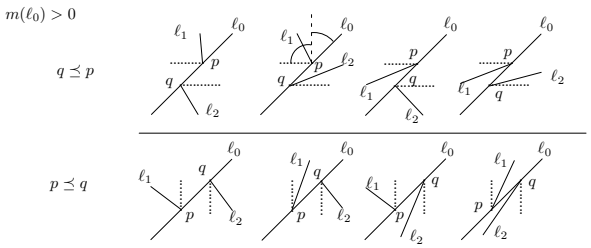


Figure 5: $m(\ell_0) > 0$.

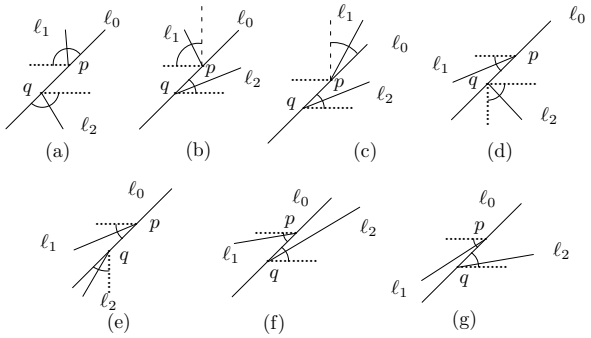


Figure 6: Configurations for $m(\ell_0) > 0$ and $q \preceq p$.

For an easier analysis of the point configurations we refine the four cases in Figure 5 for $q \preceq p$ into the seven cases in Figure 6 which can be reduced as follows: case (d) is obtained from case (b) by a 180-degree rotation; case (e) is obtained from case (c) by a 180-degree rotation; and case (g), where ℓ_2' intersect ℓ_1, is obtained from case (f), where ℓ_1' intersect ℓ_2, by a 180-degree rotation. Thus, we only consider the four types (1), (2), (3), and (4) of 2-level trees in Figure 7 with a concrete assignment of colors. For types (2), (3), and (4), the line ℓ_1' always intersects ℓ_2.

We design algorithms for the types of 2-level trees $T = (\ell_1, \ell_0, \ell_2)$ illustrated in Figure 7. From now on, we assume that R and B are not separable by a line, wedge, strip, zigzag, or convex polygonal chain. The following lemma is straightforward.

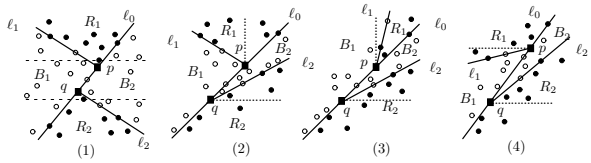


Figure 7: The 4 types of 2-level trees up to symmetry.

Lemma 6 *If R and B are separable by a 2-level tree, then $I_{B,R} \in \{0, 2, 4, 6\}$, where $I_{B,R}$ is the number of intersections between pairs of edges of $CH(B)$ and $CH(R)$.*

In the full paper we prove:

Lemma 7 *If R and B are separable by a 2-level tree $T = (\ell_0, \ell_1, \ell_2)$, then it holds that ℓ_0 is a supporting line of $CH(R_1)$ or $CH(R_2)$, and ℓ'_1 (ℓ'_2) is a common supporting line of $CH(R_1)$ and $CH(B_1)$ ($CH(R_2)$ and $CH(B_2)$).*

An overview of algorithm is: Compute a line that classifies/separates one of the point sets (say R) into subsets R_1 and R_2, and use this classification to look for a classification of B into subsets B_1 and B_2 according to a 2-level tree. We present an $O(n \log n)$ time algorithm for 2-level trees of type (1), and we show an $O(n^2)$ time algorithm for 2-level trees of types (2), (3), and (4).

Theorem 8 *Computing all the separating 2-level trees for R and B can be done in $O(n^2)$ time and space.*

In the full paper we consider also 3 or 4 colors:

Theorem 9 *A separating 2-level tree for three colored sets of n points can be computed in $\Theta(n \log n)$ time. For four colored sets of n points it can be computed in $O(n)$ time.*

4 k-Level Trees

We consider separating ($k \geq 3$)-level trees for R and B. A separating $O(\log n)$-level tree for R and B can be computed using the ham-sandwich cut theorem as follows: Compute a line yielding an equitable bipartition $B_1 \cup R_1$, $B_2 \cup R_2$ of $B \cup R$, then proceed recursively on each part until we get monochromatic subsets. At the end we get a k-level tree for $n = 2^k$. Since $n \leq 2^{O(\log n)}$, k is $O(\log n)$.

A k-level tree produces a subdivision of the plane into monochromatic convex cells, bounded by at most k lines. From the $\binom{n}{2}$ lines we get $\binom{\binom{n}{2}}{k} = O(n^{O(k)})$ cells. In $O(n^{O(k)})$ time we can compute all the cells and check which are monochromatic. Then, we use a dynamic programming algorithm to compute a minimum k-level tree for R and B in $O(n^{O(k)})$ time. This yields a quasi-polynomial time algorithm (since $k = O(\log n)$).

Theorem 10 *A separating k-level tree for R and B can be computed in $O(n^{O(k)})$ time.*

On the other hand, there exist point configurations such that the depth of the minimum k-level tree is $k = \Omega(\log n)$. See the full paper.

5 Separability With Axis-Parallel Partitions

In the full paper we prove:

Theorem 11 *A separating axis-parallel lines 2-level tree for R and B can be computed in $\Theta(n)$ time.*

Acknowledgments

D. Garijo and A. Márquez are partially supported by projects MTM2008-05866-C03-01, P06-FQM-01649 and 2008/FQM-164. E. Arkin and J. Mitchell are partially supported by the National Science Foundation (CCF-0729019). C. Seara is partially supported by projects MTM2009-07242 and Gen. Cat. DGR2009GR1040.

References

[1] E. M. Arkin, F. Hurtado, J. S. B. Mitchell, C. Seara, and S. S. Skiena. Some lower bounds on geometric separability problems. *Internat. J. Comput. Geometry and App.* 16(1):1–26, 2006.

[2] E. M. Arkin, F. Hurtado, J. S. B. Mitchell, C. Seara, and S. S. Skiena. Some separability problems in the plane. *EuroCG*, 2000, pp. 51–54.

[3] T. Asano, J. Hershberger, J. Pach, E. Sontag, D. Souvaine, and S. Suri. Separating bi-chromatic points by parallel lines. *Proc. 2nd Canadian Conf. on Computational Geometry*, 1990, pp. 46–49.

[4] H. Edelsbrunner and F. P. Preparata. Minimum polygonal separation. *Information and Computation* 77:218–232, 1988.

[5] S. Fekete. On the complexity of min-link red-blue separation problem. *Manuscript*, 1992.

[6] F. Hurtado, M. Mora, P. A. Ramos, and C. Seara. Separability by two lines and by nearly-straight polygonal chains. *Discrete Applied Math.* 144:110–122, 2004.

[7] F. Hurtado, M. Noy, P. A. Ramos, and C. Seara. Separating objects in the plane by wedges and strips. *Discrete Applied Math.* 109:109–138, 2000.

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

[9] F. P. Preparata and M. I. Shamos, *Computacional Geometry, An Introduction*, Springer-Verlag, 1988.

Order types of segments in floorplan partitions

Andrei Asinowski* Gill Barequet† Toufik Mansour‡ Ron Y. Pinter§

Abstract

Floorplan partitions, whose generation is a critical stage in e.g. integrated circuit layout and in architectural design, give rise to several interesting orders. Ackerman, Barequet and Pinter studied the orders induced by the neighborhood relations between *rectangles* forming a partition, and obtained a natural bijection between them and Baxter permutations (that can be described as $(2\text{-}41\text{-}3, 3\text{-}14\text{-}2)$-avoiding permutations). In the present paper, we study orders induced by the neighborhood relations between *segments* forming such partitions, and show a natural bijection between these order types and another family of permutations, namely of those avoiding $(2\text{-}14\text{-}3, 3\text{-}41\text{-}2)$, and investigate related questions.

1 Introduction: Neighborhood relations between rectangles; R-permutations

In [1], Ackerman, Barequet and Pinter studied floorplan partitions and a representation of neighborhood relations between rectangles in such partitions in terms of permutation patterns.

A *floorplan partition* is a partition of a rectangle into smaller interior-disjoint rectangles. It is required that segments forming the partition do not cross, and a meeting of segments can have one of the following forms: $\dashv$, $\perp$, $\vdash$, $\top$ (but not $+$). In particular, this implies that if the number of segments in a floorplan partition P is n, then the number of rectangles in P is $n+1$.

Neighborhood relations of rectangles forming the partition are defined as follows. A rectangle A *is a left-neighbor of* B if there is a vertical segment in the partition that contains the right side of A and the left side of B. A relation A *is to the left of* B, denoted by $A \leftarrow B$, is defined to be the transitive closure of the relation A *is a left neighbor of* B. We denote: $A \twoheadleftarrow B$ if $A = B$ or $A \leftarrow B$.

*Department of Mathematics, Technion - Israel Institute of Technology. E-mail andrei@tx.technion.ac.il.

†Department of Computer Science, Technion - Israel Institute of Technology, *and* Department of Computer Science, Tufts University, Medford, MA 02155. E-mail barequet@cs.technion.ac.il.

‡Department of Mathematics, University of Haifa, Israel. E-mail toufik@math.haifa.ac.il.

§Department of Computer Science, Technion - Israel Institute of Technology. E-mail pinter@cs.technion.ac.il.

The terms A *is a below-neighbor of* B, and A *is below* B (denoted by $A \downarrow B$) are defined similarly. We also denote: $A \downarrow\!\!\!\downarrow B$ if $A = B$ or A is below B.

Two partitions are considered identical if they can be obtained from each other by a continuous transformation that does not change neighborhood relations of each rectangle. See Fig. 1 for an example (observe in this figure $A \downarrow D$, $B \leftarrow C$).

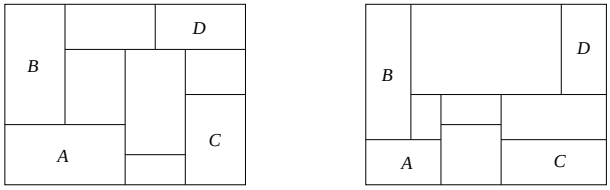


Figure 1: Two representations of the same floorplan partition.

The following results are obtained in [1]. Let P be a floorplan partition. Any two different rectangles A and B in P are in exactly one neighborhood relation: either $A \leftarrow B$, or $B \leftarrow A$, or $A \downarrow B$, or $B \downarrow A$. It follows that the relations $\swarrow$ and $\nwarrow$ between rectangles of P defined by

$$A \swarrow B \quad \text{if} \quad A = B, \text{ or } A \leftarrow B, \text{ or } A \downarrow B,$$
$$A \nwarrow B \quad \text{if} \quad A = B, \text{ or } A \leftarrow B, \text{ or } B \downarrow A$$

are linear orders. Each of them can be used for labeling the rectangles of P by $1, 2, \ldots, n+1$. For example, in the $\swarrow$ order, the rectangle in the lower left corner will then be labeled 1, and the rectangle in the upper right corner $n+1$. Let $R(P)$ be the sequence $a_1, a_2, \ldots, a_{n+1}$, where a_i is the label in the $\nwarrow$ order of the rectangle which is labeled i in $\swarrow$ order, for all $1 \le i \le n+1$. Then $R(P)$ is a permutation of $[n+1] = \{1, 2, \ldots, n+1\}$, we shall call it *the R-permutation corresponding to* P. Loosely speaking, $R(P)$ is determined by labeling the rectangles according to the $\swarrow$ order, and then reading these labels passing the rectangles according to the $\nwarrow$ order. Fig. 2 shows a floorplan partition and the corresponding R-permutation. The main result of [1] is a theorem which states that for any floorplan partition P, the permutation $R(P)$ is a Baxter permutation (that is, a $(2\text{-}41\text{-}3, 3\text{-}14\text{-}2)$-avoiding permutation[1]); furthermore, the correspondence $P \mapsto R(P)$ is a natural bijection: all the information about the neighborhood

[1]This kind of notation is discussed in Section 3.1.

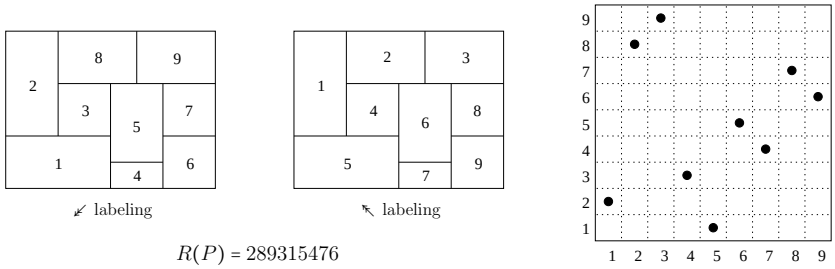


Figure 2: Constructing the R-permutation corresponding to a floorplan partition.

relations between rectangles in P can be read from $R(P)$.

2 Neighborhood relations between segments; S-permutations

2.1 Four orders of segments: $\twoheadleftarrow$, $\twoheaddownarrow$, $\swarrow$ and $\nwarrow$

Similarly to the described above, in our work we define and study neighborhood relations between **segments** which form a floorplan partition P.

A segment A is a *left neighbor* of a segment B if one of the following holds:

- A and B are vertical, and there is a rectangle of P such that its left side is included in A and the right side is included in B; or
- A is vertical, B is horizontal, and the left endpoint of B lies in A; or
- A is horizontal, B is vertical, and the right endpoint of A lies in B.

Several examples are shown in Fig. 3.

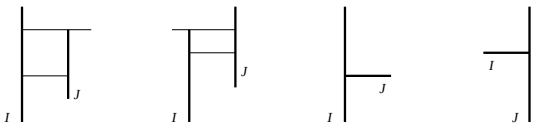


Figure 3: The segment A is a left neighbor of the segment B (several cases).

A relation A *is to the left of* B, denoted by $A \leftarrow B$ is the transitive closure of the relation "A is a left neighbor of the segment B". We denote: $A \twoheadleftarrow B$ if $A = B$ or $A \leftarrow B$.

The terms: A *is a below-neighbor of* B and; A *is below* B (and then B *is above* A) (denoted by $A \downarrow B$), are defined similarly. Denote: $A \twoheaddownarrow B$ if $A = B$ or $A \downarrow B$.

Similarly to the facts for the corresponding orders between rectangles, we prove that the relations $\twoheadleftarrow$ and $\twoheaddownarrow$ are partial order relations, and that every two different segments, A and B, in a floorplan partition P, are in precisely one of these relations: either $A \leftarrow B$, or $B \leftarrow A$, or $A \downarrow B$, or $B \downarrow A$.

Finally, we define two relations $\swarrow$ and $\nwarrow$ between segments of P as follows:

$$A \swarrow B \quad \text{if} \quad A = B, \text{ or } A \leftarrow B, \text{ or } A \downarrow B,$$
$$A \nwarrow B \quad \text{if} \quad A = B, \text{ or } A \leftarrow B, \text{ or } B \downarrow A.$$

These relations $\swarrow$ and $\nwarrow$ are linear orders.

2.2 S-permutations

Let P be a floorplan partition of a rectangle with n segments. Let $S(P)$ be the sequence $b_1, b_2, \ldots, b_n$ where b_i is the label in the $\nwarrow$ order of the segment which is labeled i in the $\swarrow$ order, for all $1 \le i \le n$. It is clear that $S(P)$ is a permutation of $[n] = \{1, 2, \ldots, n\}$; we shall call it the *S-permutation* of P and denote it by $S(P)$.

Thus, we assign a permutation to a partition in a way similar to that from [1], but this time we use not rectangles but segments. Note that $S(P)$ is a permutation of $[n]$, while $R(P)$ is a permutation of $[n+1]$.

If a segment of a floorplan partition P is labeled j in the $\swarrow$ order and labeled i in the $\nwarrow$ order, then $S(P)(i) = j$. In other words, the graph[2] of $S(P)$ has the point (i, j) which will be denoted by N_i.

Fig. 4 shows a partition P with segments labeled in the form (i, j) where j is the label of a segment according to the $\swarrow$ order, and i is its label according to the $\nwarrow$ order, and the graph of $S(P)$.

3 Our Results

3.1 (2-14-3, 3-41-2)-avoiding permutations

First we introduce a family of permutations that will appear in the main result.

We say that a permutation $\pi = a_1 a_2 \ldots a_n$ is (2-14-3, 3-41-2)-avoiding if there are no $1 \le i_1 < i_2 < i_3 < i_4 \le n$ such that $i_3 = i_2 + 1$ and either $\pi(i_2) < \pi(i_1) < \pi(i_4) < \pi(i_3)$ or $\pi(i_3) < \pi(i_4) < \pi(i_1) < \pi(i_2)$.

[2] For π, a permutation of $[n]$, the graph of π is the point set $\{(i, \pi(i)) : i \in [n]\}$.

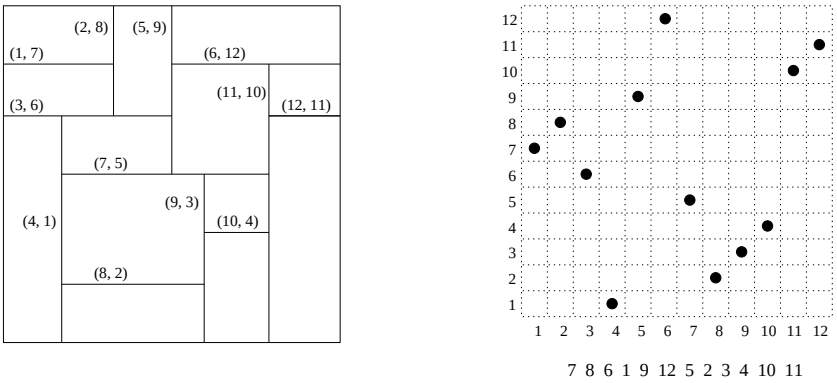


Figure 4: A partition P and the corresponding S-permutation $S(P)$.

In other words, π does not contain a subpermutation isomorphic to 2143 or to 3412 so that the labels corresponding to 1 and 4 are adjacent.

This is a special case of so-called dashed notation in permutation patterns. See [3] for a general definition and many results concerning generalized permutation patters.

3.2 A bijection between order types of segments in floorplan partitions and (2-14-3, 3-41-2)-avoiding permutations

Our main result is the following two theorems:

Theorem 1 *Let P be a floorplan partition. Then $S(P)$ is a* (2-14-3, 3-41-2)*-avoiding permutation.*

Theorem 2 *For each π, a* (2-14-3, 3-41-2)*-avoiding permutation of $[n]$, there exists a floorplan partition P with n segments such that $S(P) = \pi$.*

In other words, there is a bijection between order types of **segments** in floorplan partitions and (2-14-3, 3-41-2)-avoiding permutations. Compare this with the result from [1] which says that there is a bijection between order types of **rectangles** in floorplan partitions and Baxter permutations (which are described in the dashed notation as (2-41-3, 3-14-2)-avoiding permutations).

3.3 Relations between R-permutations to S-permutations

Let P be a floorplan partition. We show how $S(P)$ is related to $R(P)$. The following property of Baxter permutations will be used.

Proposition 3 *Let ρ be a Baxter permutation of $[n+1]$. For each i, $1 \le i \le n$ there exists a unique j_i, $1 \le j_i \le n$, such that:*

- *For i with $\rho(i) < \rho(i+1)$ we have $\rho(i) \le j_i < \rho(i+1)$ and $\rho^{-1}(j_i) \le i < \rho^{-1}(j_i+1)$;*
- *for $\rho(i) > \rho(i+1)$ we have $\rho(i+1) \le j_i < \rho(i)$ and $\rho^{-1}(j_i+1) \le i < \rho^{-1}(j_i)$.*

The relation of $S(P)$ to $R(P)$ is the following:

Theorem 4 *Let P be a floorplan partition, and let $\rho = R(P)$. For each i, let j_i be as in Proposition 3, with respect to ρ. Then $S(P) = j_1, j_2, \ldots, j_n$.*

It is convenient to draw the graphs of $R(P)$ and $S(P)$ on the same diagram where the points of the graph of $R(P)$ are in the centers of the grid squares, and the points of the graph of $S(P)$ are in the grid nodes, see Fig. 5 (the points of the graph of $R(P)$ are black, the points of the graph of $S(P)$ are white).

3.4 Which partitions have the same S-permutation

Let P_1 and P_2 be two floorplan partitions. We shall see when we have $S(P_1) = S(P_2)$.

We first characterize the floorplan partitions whose S-permutation is $123\ldots n$. Such a partition will be called *a (separate) ascending F-block of the size $n+1$*. In such a partition, all vertical segments extend from the lower to the upper side of the border, and between a pair of adjacent vertical segments there is at most one horizontal segment. Therefore an ascending F-block consists of several rectangles that extend from the lower to the upper side of the boundary and several pairs of rectangles whose union is a rectangle that extends from the lower to the upper side of the border. Fig. 6 shows several F-blocks of the size 5. *Descending F-blocks*, which are partitions whose S-permutation is $n\ldots321$, are similarly described. Two F-blocks are *equivalent* if have the same size and if they are both ascending or both descending.

Let now P be any floorplan partition. We define an *F-block in P* as a set of rectangles in a partition, whose union is an F-block, as defined above.

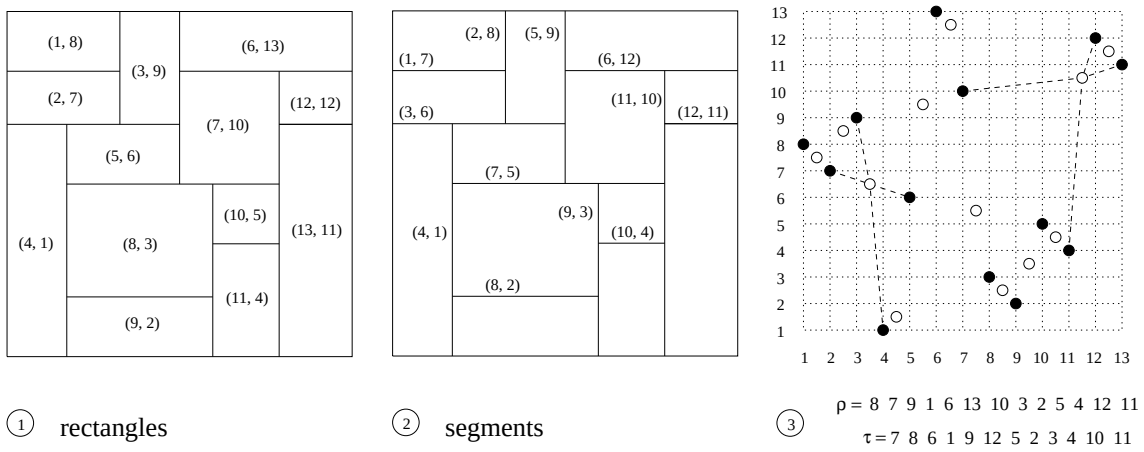


Figure 5: A partition P with labeling of rectangles (1) and with labeling of segments (2); $R(P)$ (black points) together with $S(P)$ (white points).

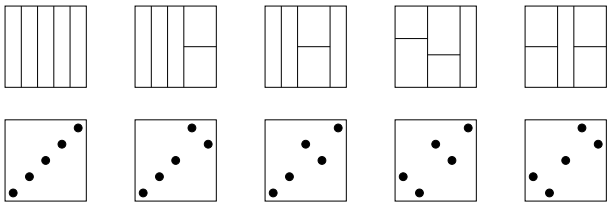


Figure 6: Five (out of eight) ascending F-blocks for $n = 4$, and their R-permutations.

Theorem 5 *Let P_1 and P_2 be two floorplan partitions with n segments. Then $S(P_1) = S(P_2)$ if and only if P_1 and P_2 may be obtained from each other by replacing several F-blocks by equivalent F-blocks.*

3.5 Other results

Other results in this work include:

- Relations between rectangles and segments in a floorplan partition. For example, we show how to see from $R(P)$ and $S(P)$ what segments contain the sides of a specified rectangle.

- Enumerating issues: a generating tree for the enumerating sequence for (2-14-3, 3-41-2)-avoiding permutations of $[n]$ and several functional equations related to its generating function. Yet the question of finding an explicit formula for the number of (2-14-3, 3-41-2)-avoiding permutations of $[n]$ remains open.

- The case of guillotine partitions. The family of permutations which correspond to the order types induced by segments in guillotine partitions, is that of (2-14-3, 3-41-2, 2-4-1-3, 3-1-4-2)-avoiding permutations. The enumeration is easy in this case.

- A multidimensional generalization of the previous item: guillotine partitions of a d-dimensional box. Here we have $d-1$-dimensional cuts instead of segments, and neighborhood relations between them are naturally defined. We prove that the number of order types induced by cuts in guillotine partitions of a d-dimensional box with n cuts is

$$\sum_{k=0}^{n}\sum_{j=0}^{n-k}(-1)^{n-k-j}c_k\binom{2k+j}{j}\binom{k+1+j}{n-k-j}(d-1)^{n-j}.$$

  (A formula for the number of order types of *rectangles* in guillotine partitions of a d-dimensional box with n cuts was found by Ackerman *et al.* in [2].)

References

[1] E. Ackerman, G. Barequet, and R. Y. Pinter. A bijection between permutations and floorplans, and its applications. *Discrete Applied Mathematics*, **154**, (2006), 1674 – 1684.

[2] E. Ackerman, G. Barequet, R. Y. Pinter, and D. Romik. The number of guillotine partitions in d dimensions. *Information Processing Letters*, **98** (2006), 162 – 167.

[3] E. Steingrímsson. Generalized permutation patterns – a short survey. To appear in: Permutation Patterns, St. Andrews 2007, S. A. Linton, N. Ruskuc, V. Valter (eds.), LMS Lecture Note Series, Cambridge University Press. `http://www.math.ru.is/download/St08__Generalized_permutation.pdf`

The geodesic diameter of polygonal domains

Sang Won Bae* Matias Korman† Yoshio Okamoto‡

Abstract

This paper studies the geodesic diameter of polygonal domains having h holes and n corners. For simple polygons (i.e., $h = 0$), it is known that the geodesic diameter is determined by a pair of corners of a given polygon and can be computed in linear time. For general polygonal domains with $h \geq 1$, however, no algorithm for computing the geodesic diameter was known prior to this paper. We present first algorithms that compute the geodesic diameter of a given polygonal domain in worst-case time $O(n^{7.73})$ or $O(n^7(\log n + h))$. The algorithms are based on new geometric observations, part of which states as follows: the geodesic diameter of a polygonal domain can be determined by two points in its interior, and in that case there are at least five shortest paths between the two points.

1 Introduction

In this paper, we address the geodesic diameter problem in polygonal domains. The geodesic distance $\mathrm{d}(p, q)$ between any two points p, q in a polygonal domain $\mathcal{P}$ is defined as the (Euclidean) length of a shortest obstacle-avoiding path between p and q. The *geodesic diameter* $\mathrm{diam}(\mathcal{P})$ of a polygonal domain $\mathcal{P}$ is defined as $\mathrm{diam}(\mathcal{P}) := \max_{s,t\in\mathcal{P}} \mathrm{d}(s, t)$. A pair (s, t) of points in $\mathcal{P}$ that realizes the geodesic diameter $\mathrm{diam}(\mathcal{P})$ is called a *diametral pair*. The geodesic diameter problem is to find the value of $\mathrm{diam}(\mathcal{P})$ and a diametral pair.

For simple polygons (i.e., $h = 0$), the geodesic diameter has been extensively studied and fully understood. Chazelle [2] provided the first $O(n^2)$-time algorithm computing the geodesic diameter of a simple polygon, and Suri [9] presented an $O(n \log n)$-time algorithm that solves the all-geodesic-farthest neighbors problem, computing the farthest neighbor of every corner and thus finding the geodesic diameter. At last, Hershberger and Suri [5] showed that the diameter can be computed in linear time using their fast matrix search technique. On the other hand, to the best of our knowledge, no algorithm for computing $\mathrm{diam}(\mathcal{P})$ has yet been discovered when $\mathcal{P}$ is a polygonal domain having one or more holes ($h \geq 1$).

This fairly wide gap between simple polygons and polygonal domains is seemingly due to the uniqueness of the shortest path between any two points; it is well known that there is a unique shortest path between any two points in a simple polygon [4]. Using this uniqueness, one can show that the diameter is indeed realized by a pair of corners in V; that is, $\mathrm{diam}(\mathcal{P}) = \max_{u,v\in V} \mathrm{d}(u, v)$ if $h = 0$ [5, 9]. For general polygonal domains with $h \geq 1$, however, this is not the case. In this paper, we exhibit several examples where the diameter is realized by non-corner points on $\partial\mathcal{P}$ or even by interior points of $\mathcal{P}$ (see Figure 1). This observation also shows an immediate difficulty in devising any exhaustive algorithm since the search space like $\partial\mathcal{P}$ or the whole domain $\mathcal{P}$ is not discrete.

In this paper, we present the first algorithms that compute the geodesic diameter of a given polygonal domain in $O(n^{7.73})$ or $O(n^7(\log n + h))$ time in the worst case. We also show that for small constant h the diameter can be computed much faster.

2 Preliminaries

We are given as input a polygonal domain $\mathcal{P}$ with h holes and n corners. More precisely, $\mathcal{P}$ consists of an outer simple polygon in the plane $\mathbb{R}^2$ and a set of h (≥ 0) disjoint simple polygons inside the outer polygon. As a subset of $\mathbb{R}^2$, $\mathcal{P}$ is the region contained in its outer polygon *excluding* the interior of the holes; thus $\mathcal{P}$ is a bounded, closed subset of $\mathbb{R}^2$. The boundary $\partial\mathcal{P}$ of $\mathcal{P}$ is regarded as a series of *obstacles* so that any feasible path inside $\mathcal{P}$ is not allowed to cross $\partial\mathcal{P}$. Note that some portion or the whole of a feasible path may go along the boundary $\partial\mathcal{P}$. The *length* of a path is the sum of the Euclidean lengths of its segments. It is well known from earlier work that there always exists a *shortest (feasible) path* between any two points $p, q \in \mathcal{P}$ [7]. The *geodesic distance*, denoted by $\mathrm{d}(p, q)$, is then defined to be the length of a shortest path between $p \in \mathcal{P}$ and $q \in \mathcal{P}$.

*Department of Computer Science and Engineering, POSTECH, swbae@postech.ac.kr. Work supported by the Brain Korea 21 Project.

†Computer Science Department, Université Libre de Bruxelles, mkormanc@ulb.ac.be. Work supported by the Communauté française de Belgique - Actions de Recherche Concertées (ARC).

‡Graduate School of Information Science and Engineering, Tokyo Institute of Technology, okamoto@is.titech.ac.jp. Work supported by Global COE Program "Computationism as a Foundation for the Sciences" and Grant-in-Aid for Scientific Research from Ministry of Education, Science and Culture, Japan, and Japan Society for the Promotion of Science

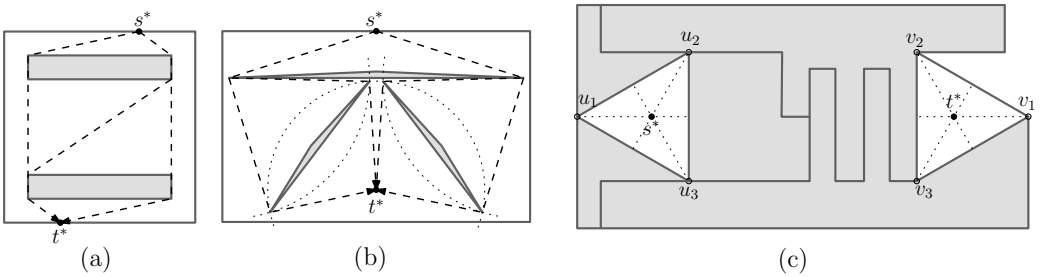


Figure 1: Three polygonal domains where the geodesic diameter is determined by a pair (s^*, t^*) of non-corner points; Gray-shaded regions depict the interior of the holes and dark gray segments depict the boundary $\partial\mathcal{P}$. Recall that $\mathcal{P}$, as a set, contains its boundary $\partial\mathcal{P}$. (a) Both s^* and t^* lie on $\partial\mathcal{P}$. There are three shortest paths between s^* and t^*. In this polygonal domain, there are two (symmetric) diametral pairs. (b) $s^* \in \partial\mathcal{P} \setminus V$ and $t^* \in \mathrm{int}\mathcal{P}$. Three triangular holes are placed in a symmetric way. There are four shortest paths between s^* and t^*. (c) Both s^* and t^* lie in the interior $\mathrm{int}\mathcal{P}$. Here, the five holes are packed like jigsaw puzzle pieces, forming narrow corridors (dark gray paths) and two empty, regular triangles. Observe that $\mathrm{d}(u_1, v_1) = \mathrm{d}(u_1, v_2) = \mathrm{d}(u_2, v_2) = \mathrm{d}(u_2, v_3) = \mathrm{d}(u_3, v_3) = \mathrm{d}(u_3, v_1)$. s^* and t^* lie at the centers of the triangles formed by the u_i and the v_i, respectively. There are six shortest paths between s^* and t^*. More details on this example can be found in the extended version of this paper [1].

Shortest path map. Let V be the set of all corners of $\mathcal{P}$ and $\pi(s,t)$ be a shortest path between $s \in \mathcal{P}$ and $t \in \mathcal{P}$. Then, it is represented as a sequence $\pi(s,t) = (s, v_1, \ldots, v_k, t)$ for some $v_1, \ldots, v_k \in V$; that is, a polygonal chain through a sequence of corners [7]. Note that possibly we may have $k = 0$ when $\mathrm{d}(s,t) = \|s-t\|$. If two paths (with possibly different endpoints) induce the same sequence of corners, then they are said to have the same *combinatorial structure*.

The *shortest path map* $\mathsf{SPM}(s)$ for a fixed $s \in \mathcal{P}$ is a decomposition of $\mathcal{P}$ into cells such that every point in a common cell can be reached from s by shortest paths of the same combinatorial structure. Each cell $\sigma_s(v)$ of $\mathsf{SPM}(s)$ is associated with a corner $v \in V$ or s itself, which is the last corner of $\pi(s,t)$ for any t in the cell $\sigma_s(v)$. In particular, the cell $\sigma_s(s)$ is the set of points t such that $\pi(s,t)$ passes through no corner in V and thus $\mathrm{d}(s,t) = \|s-t\|$. Each edge of $\mathsf{SPM}(s)$ is an arc on the boundary of two incident cells $\sigma_s(v_1)$ and $\sigma_s(v_2)$ and thus determined by two corners $v_1, v_2 \in V \cup \{s\}$. Similarly, each vertex of $\mathsf{SPM}(s)$ is determined by at least three corners $v_1, v_2, v_3 \in V \cup \{s\}$. Note that for fixed $s \in \mathcal{P}$ a point t that locally maximizes $\mathrm{d}_s(t) := \mathrm{d}(s,t)$ lies at either (1) a vertex of $\mathsf{SPM}(s)$, (2) an intersection between the boundary $\partial\mathcal{P}$ and an edge of $\mathsf{SPM}(s)$, or (3) a corner in V.

The shortest path map $\mathsf{SPM}(s)$ has $O(n)$ complexity can be computed in $O(n \log n)$ time using $O(n \log n)$ working space [6]. For more details on shortest path maps, see [7, 6, 8].

Path-length function. If $\pi(s,t) \neq \overline{st}$, then there are two corners $u, v \in V$ such that $\pi(s,t)$ is formed as the union of a shortest path from u to v and two segments $\overline{su}$ and $\overline{vt}$. Note that u and v are not necessarily distinct. In order to realize such a path, we assert that s is visible from u and t is visible from v; thus, $s \in \mathsf{VP}(u)$ and $t \in \mathsf{VP}(v)$, where $\mathsf{VP}(p)$ for any $p \in \mathcal{P}$ is defined to be the set of all points $q \in \mathcal{P}$ such that $\overline{pq} \subset \mathcal{P}$. The set $\mathsf{VP}(p)$ is also called the *visibility profile* of $p \in \mathcal{P}$ [3].

We now define the *path-length function* $\mathrm{len}_{u,v} \colon \mathsf{VP}(u) \times \mathsf{VP}(v) \to \mathbb{R}$ for any fixed pair of corners $u, v \in V$ to be

$$\mathrm{len}_{u,v}(s,t) := \|s-u\| + \mathrm{d}(u,v) + \|v-t\|.$$

Then, $\mathrm{len}_{u,v}(s,t)$ represents the length of the path from s to t that has the fixed combinatorial structure, entering u from s and exiting v to t. Also, unless $\mathrm{d}(s,t) = \|s-t\|$ (equivalently, $s \in \mathsf{VP}(t)$), the geodesic distance $\mathrm{d}(s,t)$ can be expressed as the pointwise minimum of some path-length functions:

$$\mathrm{d}(s,t) = \min_{u \in \mathsf{VP}(s),\ v \in \mathsf{VP}(t)} \mathrm{len}_{u,v}(s,t).$$

Consequently, we have two possibilities for a diametral pair (s^*, t^*); either we have $\mathrm{d}(s^*, t^*) = \|s^* - t^*\|$ or the pair (s^*, t^*) is a local maximum of the lower envelope of several path-length functions.

3 Properties of Geodesic-Maximal Pairs

We call a pair $(s^*, t^*) \in \mathcal{P} \times \mathcal{P}$ *maximal* if (s^*, t^*) is a local maximum of the geodesic distance function d. That is, (s^*, t^*) is maximal if and only if there are two neighborhoods $U_s, U_t \subset \mathbb{R}^2$ of s^* and of t^*, respectively, such that for any $s \in U_s \cap \mathcal{P}$ and any $t \in U_t \cap \mathcal{P}$ we have $\mathrm{d}(s^*, t^*) \geq \mathrm{d}(s,t)$. For any pair (s,t), let $\Pi(s,t) = \{\pi_1, \ldots, \pi_m\}$ be the set of all distinct shortest paths from s to t, where m denotes the

(VV)	$s^* \in V$,	$t^* \in V$	implies	$\lvert\Pi(s^*,t^*)\rvert \geq 1, \lvert V_{s^*}\rvert \geq 1, \lvert V_{t^*}\rvert \geq 1$;
(VB)	$s^* \in V$,	$t^* \in \mathcal{B}$	implies	$\lvert\Pi(s^*,t^*)\rvert \geq 2, \lvert V_{s^*}\rvert \geq 1, \lvert V_{t^*}\rvert \geq 2$;
(VI)	$s^* \in V$,	$t^* \in \mathrm{int}\mathcal{P}$	implies	$\lvert\Pi(s^*,t^*)\rvert \geq 3, \lvert V_{s^*}\rvert \geq 1, \lvert V_{t^*}\rvert \geq 3$;
(BB)	$s^* \in \mathcal{B}$,	$t^* \in \mathcal{B}$	implies	$\lvert\Pi(s^*,t^*)\rvert \geq 3, \lvert V_{s^*}\rvert \geq 2, \lvert V_{t^*}\rvert \geq 2$;
(BI)	$s^* \in \mathcal{B}$,	$t^* \in \mathrm{int}\mathcal{P}$	implies	$\lvert\Pi(s^*,t^*)\rvert \geq 4, \lvert V_{s^*}\rvert \geq 2, \lvert V_{t^*}\rvert \geq 3$;
(II)	$s^* \in \mathrm{int}\mathcal{P}$,	$t^* \in \mathrm{int}\mathcal{P}$	implies	$\lvert\Pi(s^*,t^*)\rvert \geq 5, \lvert V_{s^*}\rvert \geq 3, \lvert V_{t^*}\rvert \geq 3$.

Figure 2: Necessary conditions for a pair of points to be maximal.

.

number of shortest paths. Let u_i and v_i be the first and the last corners in V along π_i from s to t, and let $V_s := \{u_1, \ldots, u_m\}$ and $V_t := \{v_1, \ldots, v_m\}$.

Let E be the set of all sides of $\mathcal{P}$ without their endpoints and $\mathcal{B}$ be their union. Note that $\mathcal{B} = \partial\mathcal{P}\backslash V$, the boundary of $\mathcal{P}$ except the corners V.

Theorem 1 *Suppose that (s^*, t^*) is a maximal pair in $\mathcal{P}$ and $\Pi(s^*, t^*)$, V_{s^*}, and V_{t^*} be defined as above. The implications of Figure 2 hold. Moreover, each of the above bounds is best possible by examples.*

Due to space constraints proofs of this theorem is omitted (and can be found in the extended version [1]).

4 Computing the Geodesic Diameter

Since a diametral pair is in fact maximal, it falls into one of the cases shown in Theorem 1. In order to find a diametral pair we examine all possible scenarios accordingly.

Cases **(V–)**, where at least one point is a corner in V, can be handled in $O(n^2 \log n)$ time by computing $\mathsf{SPM}(v)$ for every $v \in V$ and traversing it to find the farthest point from v, as discussed in Section 2. We thus focus on Cases **(BB)**, **(BI)**, and **(II)**, where a diametral pair consists of two non-corner points.

From the computational point of view, the most difficult case corresponds to Case **(II)** of Theorem 1; in particular, the case in which $|\Pi(s^*, t^*)| = |V_{s^*}| = |V_{t^*}| = 5$. For such a case we do the following: we choose any five corners $u_1, \ldots, u_5 \in V$ (as a candidate for the set V_{s^*}) and overlay their shortest path maps $\mathsf{SPM}(u_i)$. Since each $\mathsf{SPM}(u_i)$ has $O(n)$ complexity, the overlay consists of $O(n^2)$ cells. Then, any cell of the overlay is the intersection of five cells associated with $v_1, \ldots, v_5 \in V$ in $\mathsf{SPM}(u_1), \ldots, \mathsf{SPM}(u_5)$, respectively. Choosing a cell of the overlay, we get five (possibly, not distinct) $v_1, \ldots, v_5$ and thus a constant number of candidate pairs by solving the system $\mathrm{len}_{u_1,v_1}(s,t) = \cdots = \mathrm{len}_{u_5,v_5}(s,t)$. We iterate this process for all possible tuples of five corners $u_1, \ldots, u_5$, obtaining a total of $O(n^7)$ candidate pairs in $O(n^7 \log n)$ time. Note that this method also covers the case of $|\Pi(s^*, t^*)| > 5$. Recall that each path-length function $\mathrm{len}_{u,v}$ is an algebraic function of degree at most 4. Thus, given five distinct pairs (u_i, v_i) of corners, we can compute all candidate pairs (s, t) in $O(1)$ time by solving the system[1]. Indeed when five distinct pairs $(u_1, v_1), \ldots, (u_5, v_5)$ of corners in V such that $\mathrm{len}_{u_i,v_i}(s^*, t^*) = \mathrm{d}(s^*, t^*)$ for any $i \in \{1, \ldots, 5\}$ are known, their system of equations $\mathrm{len}_{u_1,v_1}(s,t) = \cdots = \mathrm{len}_{u_5,v_5}(s,t)$ determines a 0-dimensional zero set corresponding to a constant number of candidate pairs in $\mathrm{int}\mathcal{P}\times\mathrm{int}\mathcal{P}$. The **(II)** case (in which $|V_{s^*}| \leq 4$) can be handled similarly, resulting in $O(n^6)$ candidate pairs.

In order to test the validity of each candidate pair (s, t), we check the geodesic distance $\mathrm{d}(s, t)$ using a two-point query structure of Chiang and Mitchell [3]: for a fixed parameter $0 < \delta \leq 1$ and any fixed $\epsilon > 0$, we can construct, in $O(n^{5+10\delta+\epsilon})$ time, a data structure that supports $O(n^{1-\delta} \log n)$-time two-point shortest path queries. Then, the total running time is $O(n^7 \log n) + O(n^{5+10\delta+\epsilon}) + O(n^7) \times O(n^{1-\delta} \log n)$. We set $\delta = \frac{3}{11}$ to optimize the running time to $O(n^{7+\frac{8}{11}+\epsilon})$.

Also, we can use an alternative two-point query data structure whose performance is sensitive to the number h of holes [3]: after $O(n^5)$ preprocessing time using $O(n^5)$ storage, two-point queries can be answered in $O(\log n + h)$ time.Using this alternative structure, the total running time of our algorithm becomes $O(n^7(\log n + h))$. Note that this method outperforms the previous one when $h = O(n^{\frac{8}{11}})$.

For Case **(BI)**, we handle only the case of $|\Pi(s^*, t^*)| = 4$ with $|V_{t^*}| = 3$ or 4. For the subcase with $|V_{t^*}| = 4$, we choose any four corners from V as $v_1, \ldots, v_4$ as a candidate for V_{t^*} and overlay their shortest path maps $\mathsf{SPM}(v_i)$. The overlay, together with V, decomposes $\partial\mathcal{P}$ into $O(n)$ intervals. Then, each such interval determines $u_1, \ldots, u_4$ as above, and the side $e_s \in E$ on which s^* should lie. Now, we

[1] Here, we assume that fundamental operations on a constant number of polynomials of constant degree with a constant number of variables can be performed in constant time.

have a system of four equations on four variables: three from the corresponding path-length functions len_{u_i,v_i} which should be equalized at (s^*,t^*) and the fourth from the supporting line of e_s. Solving the system, we get a constant number of candidate maximal pairs, again by Theorem 1 and its proof. In total, we obtain $O(n^5)$ candidate pairs. The other subcase with $|V_{t^*}|=3$ can be handled similarly, resulting in $O(n^4)$ candidate pairs. As above, we can exploit two different structures for two-point queries. Consequently, we can handle Case **(BI)** in $O(n^{5+\frac{10}{11}+\epsilon})$ or $O(n^5(\log n+h))$ time.

In Case **(BB)** when $s^*,t^*\in\mathcal{B}$, we handle the case of $|\Pi(s^*,t^*)|=3$ with $|V_{s^*}|=2$ or 3. For the subcase with $|V_{s^*}|=3$, we choose three corners as a candidate of V_{s^*} and take the overlay of their shortest path maps $\mathsf{SPM}(u_i)$. It decomposes $\partial\mathcal{P}$ into $O(n)$ intervals. Then, each such interval determines three corners v_1,v_2,v_3 forming V_{t^*} and a side $e_t\in E$ on which t^* should lie. Note that we have only three equations so far; two from the three path-length functions and the third from the line supporting to e_t. Since s^* also should lie on a side $e_s\in E$ with $e_s\neq e_t$, we need to fix such a side e_s that $\bigcap_{1\le i\le 3}\mathsf{VP}(u_i)$ intersects e_s. In the worst case, the number of such sides e_s is $\Theta(n)$. Thus, we have $O(n^5)$ candidate pairs for Case **(BB)**; again, the other subcase with $|V_{s^*}|=2$ contributes to a smaller number $O(n^4)$ of candidate pairs. Testing each candidate pair can be performed as above, resulting in $O(n^{5+\frac{10}{11}+\epsilon})$ or $O(n^5(\log n+h))$ total running time.

As Case **(II)** being a bottleneck, we conclude the following.

Theorem 2 *Given a polygonal domain having n corners and h holes, the geodesic diameter and a diametral pair can be computed in $O(n^{7+\frac{8}{11}+\epsilon})$ or $O(n^7(\log n+h))$ time in the worst case, where ϵ is any fixed positive number.*

We can avoid some difficult cases when h is a small constant based on a simple observation: if there are two distinct shortest paths between s and t in $\mathcal{P}$, then we know that there is at least one hole in the region closed by the two paths. In general, if $h<k-1$, there cannot exist two points that have k or more distinct shortest paths between them.

Theorem 3 *Given a polygonal domain having n corners and h holes, the geodesic diameter and a diametral pair can be computed in the following worst-case time bound, depending on h.*

- *$O(n)$ time, if $h=0$ (by Hershberger and Suri [5]),*
- *$O(n^2\log n)$ time, if $h=1$,*
- *$O(n^5\log n)$ time, if $h=2$ or 3,*
- *$O(n^7(\log n+h))$ time, if $4\le h=O(n^{\frac{8}{11}})$,*
- *$O(n^{7+\frac{8}{11}+\epsilon})$ time, otherwise.*

5 Concluding Remark

It is worth noting that with analysis in Section 4 the number of geodesic-maximal pairs is shown to be at most $O(n^7)$. On the other hand, one can easily construct a simple polygon in which the number of maximal pairs is $\Omega(n^2)$. An interesting question is how many maximal pairs are there in a polygonal domain in the worst case.

Though we, in this paper, have focused on exact diameters only, an efficient algorithm for finding an approximate diameter would be interesting. Notice that any point $s\in\mathcal{P}$ and its farthest point $t\in\mathcal{P}$ yield a $\frac{1}{2}$-approximate diameter; that is, $\mathrm{d}(s,t)\ge\frac{1}{2}\mathrm{diam}(\mathcal{P})$. Also, based on a standard technique using a rectangular grid with a specified parameter $0<\epsilon<1/2$, one can easily obtain a $(1-\epsilon)$-approximate diameter in $O((\frac{n}{\epsilon^2}+\frac{n^2}{\epsilon})\log n)$ time. However, breaking the quadratic bound (in n) for the $(1-\epsilon)$-approximate diameter seems a challenge at this stage.

Acknowledgments

The authors thank Hee-Kap Ahn, Jiongxin Jin, Christian Knauer, and Joe Mitchell for fruitful discussion.

References

[1] S.W. Bae, M. Korman and Y. Okamoto The geodesic diameter of polygonal domains. *CoRR*, abs/1001.0695, 2010.

[2] B. Chazelle. A theorem on polygon cutting with applications. In *Proc. Annu. Sympos. Found. Comput. Sci. (FOCS)*, pages 339–349, 1982.

[3] Y.-J. Chiang and J. S. B. Mitchell. Two-point Euclidean shortest path queries in the plane. In *Proc. 10th ACM-SIAM Sympos. Discrete Algorithms (SODA)*, pages 215–224, 1999.

[4] L. J. Guibas and J. Hershberger. Optimal shortest path queries in a simple polygon. *J. Comput. Syst. Sci.*, 39(2):126–152, 1989.

[5] J. Hershberger and S. Suri. Matrix searching with the shortest path metric. *SIAM J. Comput.*, 26(6):1612–1634, 1997.

[6] J. Hershberger and S. Suri. An optimal algorithm for Euclidean shortest paths in the plane. *SIAM J. Comput.*, 28(6):2215–2256, 1999.

[7] J. S. B. Mitchell. Shortest paths among obstacles in the plane. *Internat. J. Comput. Geom. Appl.*, 6(3):309–331, 1996.

[8] J. S. B. Mitchell. Shortest paths and networks. In *Handbook of Discrete and Computational Geometry*, chapter 27, pages 607–641. CRC Press, Inc., 2nd edition, 2004.

[9] S. Suri. The all-geodesic-furthest neighbors problem for simple polygons. In *Proc. 3rd Annu. Sympos. Comput. Geom. (SoCG)*, page 64, 1987.

On the complexity of the edge guarding problem ☆

Vicente H. F. Batista† Fernando L. B. Ribeiro† Fábio Protti‡

Abstract

We revisit the complexity of the edge guarding problem on polyhedral terrains. We prove that it is NP-hard to decide whether there exists an edge set of size k that covers all of the faces of an n-vertex triangulated terrain. To such end, we introduce the notion of (F, H)-*transversals*. Also, we present a family of maximal planar graphs on n vertices that require at least $(n-2)/3$ edge guards to be covered. This reduces the gap between the previously known lower and upper bounds on the minimum edge guard set for such graphs.

1 Introduction

Since its formulation by Victor Klee in the 1970s, the *art gallery* problem has stimulated an increasing number of researchers, notably from the computational geometry community. The original question asks for the minimum number of guards that can patrol the interior of a gallery. In its standard version, a gallery is represented by a simple polygon and guards are placed at fixed points belonging to this polygon. Chvátal [4] was the first to prove that $\lfloor n/3 \rfloor$ guards are always sufficient and sometimes necessary. By using the fact that the triangulation of a simple polygon is 3-colorable, Fisk [7] designed a simpler and elegant demonstration for the same problem.

Several generalizations have also been studied, such as considering polygons with holes and orthogonal polygons, or allowing guards to patrol along areas of different shapes. We are concerned with the problem of guarding polyhedral terrains. A polyhedral terrain T can be viewed as the graph of a polyhedral function $z = F(x, y)$, defined over the xy-plane [5]. Given two points u and v on T, we say that u is *visible* from v if the line segment $\overline{uv}$ does not intersect any point strictly below T. The *visibility region* of a point u is defined by the set of points on T visible from u. If guards are supposed to have fixed positions, i.e., if they cannot move during the surveillance, they are called *point guards*. If we further restrict their positions to the terrain vertices only, we call them *vertex guards*. In another interesting variety, called *edge guards*, they are allowed to patrol along a straight line, usually the terrain edges.

Early results on the hardness of guarding polyhedral terrains were presented by Cole and Sharir [5], who showed that it is NP-complete to determine the minimum number of vertex guards that collectively see the whole terrain. Based on such proof, Zhu [13] showed that it is also NP-complete to compute the smallest edge guard set. Both works, however, require the construction of elaborate terrains where a reduction from 3-SAT is carried out. Employing recent complexity achievements for the cycle transversal problem on planar graphs, we have succeeded in developing an improved polynomial reduction that constructs a simpler terrain for demonstrating the NP-hardness of the edge guarding problem.

Regarding lower and upper bounds on the number of edge guards, Everett and Rivera-Campo [6] and Bose et al. [2] have simultaneously provided the best known bounds so far. While any terrain can be guarded using at most $\lfloor n/3 \rfloor$ edge guards [2, 3, 6], the corresponding lower bound is only $\lfloor (4n-4)/13 \rfloor$ [2]. More recently, Kaučič et al. [9] have claimed to find an inconsistency in Bose et al. [2]'s lower bound demonstration, which would weaken it to the value of $\lfloor (2n-4)/7 \rfloor$. In response to [9], Bose [1] has presented a more detailed proof ensuring that his previous result [2] was in fact correct. Here, we also consider reducing the gap between the best known lower and upper bounds so far. In this direction, we show a family of maximal planar graphs on n vertices whose minimum edge guard set is of size $(n-2)/3$.

2 Preliminaries

Let $G = (V, E)$ be an arbitrary graph, and let G_1, G_2 be two subgraphs of G. If G_1 and G_2 are disjoint and there is no edge connecting both, then we say that G_1 and G_2 are *independent*. A collection of subgraphs of G is independent if their members are pairwise independent.

Planar 3-SAT$_{2+1}$. Given a 3-SAT formula φ in conjunctive normal form, its *incidence graph* is a bipartite graph $G[\varphi] := (V, E)$, with partitions (V_c, V_v), where the sets V_c and V_v correspond to clauses and

☆This work has been partially supported by CNPq.

†Departamento de Engenharia Civil, Universidade Federal do Rio de Janeiro, COPPE, Caixa Postal 68506, Ilha do Fundão, Rio de Janeiro, RJ, 21945-970, Brazil. Email: {helano, fernando}@coc.ufrj.br.

‡Instituto de Computação, Universidade Federal Fluminense, São Domingos, Niterói, RJ, 24210-240, Brazil. Email: fabio@ic.uff.br.

variables in φ, respectively. The edges in $G[\varphi]$ denote inclusion between clauses and variables. If the graph $G[\varphi]$ is planar, we say φ is a *planar* 3-SAT instance. The problem of deciding whether such a formula is satisfiable is known to be NP-complete [10].

The 3-SAT$_{2+1}$ problem is a 3-SAT variation characterized by clauses with 2 or 3 literals, whose variables occur exactly 3 times, twice positively and once negatively. The standard decision problem defined over these formulas is still NP-complete [11]. Based on arguments from [10] and [11], we can state that Planar 3-SAT$_{2+1}$ remains hard to solve:

Lemma 1 *Planar 3-SAT$_{2+1}$ is NP-complete.*

Triangle transversal. Let G be an arbitrary graph with no loops and multiple edges, and let H be any given family of graphs. An *H-subgraph* of G is an induced subgraph of G isomorphic to an element of H. Let F be another fixed family of graphs. Then, a collection X of F-subgraphs of G is an *(F, H)-transversal* of G if every H-subgraph in G is intersected by members of X. In case F is composed by vertices only, the set X is simply called an *H-transversal*.

Given a graph G and an integer $k > 0$, the problem of deciding whether G has an H-transversal of size at most k was proved to be NP-complete [12]. Dichotomy results about C_k-transversals were presented in [8] for bounded degree graphs. In this paper, we are interested on the case where H consists only of triangles, and F is composed by edges, i.e., we deal with (C_2, C_3)-transversals. For obvious reasons, these are termed *edge-triangle-transversals*.

3 Main results

We use the fact that if the polyhedral surface is convex then the visibility region of any vertex is limited to its incident faces. Thus, the computation of an edge guard set can be reduced to an edge-triangle-transversal query on planar triangulations. First, we show that the decision version of the edge-triangle-transversal problem restricted to planar graphs is as hard to solve as Planar 3-SAT$_{2+1}$, which is guaranteed to be NP-complete by Lemma 1.

Theorem 2 *The edge-triangle-transversal problem for planar graphs is NP-complete.*

Proof. Clearly, it is in NP. To prove its NP-hardness, let F be an instance of planar 3-SAT$_{2+1}$ with variables $x_1, x_2, \ldots, x_n$ and clauses $C_1, C_2, \ldots, C_m$.

Variables. For each variable x_i, we associate a subgraph G_i in G, as illustrated in Fig. 1a. Notice that any transversal of G_i has at least 3 edges because it has a maximum of three independent triangles. Observe that if either b_i or b'_i belongs to a transversal

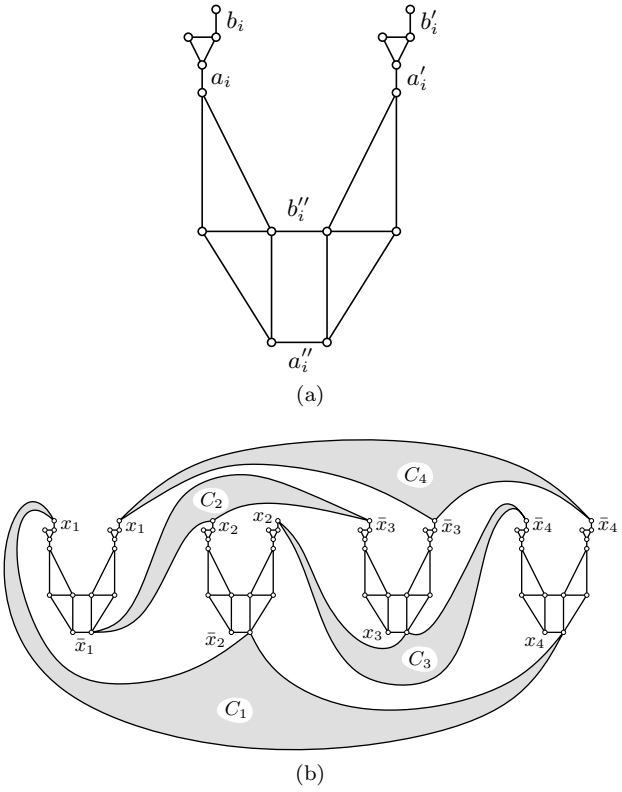


Figure 1: (a) Subgraph G_i associated with variable x_i. (b) A schematic view of the resulting graph associated to the Planar 3-SAT$_{2+1}$ formula $(x_1 \vee \bar{x}_2 \vee x_4) \wedge (x_1 \vee \bar{x}_3 \vee \bar{x}_4) \wedge (x_2 \vee x_3 \vee \bar{x}_4) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3)$.

together with a''_i, then it will be necessary to use an additional edge, e.g., b'_i, so as to cover the remaining untouched triangles. Furthermore, if a_i is in the same transversal as b''_i, then we can replace a_i with b_i without changing the transversal size. The same holds for a'_i and b'_i. Thus it is always possible to select either sets $\{a_i, a'_i, a''_i\}$ or $\{b_i, b'_i, b''_i\}$.

The upper vertices of edges b_i and b'_i, and any vertex of edge a''_i are used as connectors between G_i and the F clauses. Furthermore, if x_i (resp. $\bar{x}_i$) occurs twice in F then edges b_i and b'_i correspond to these occurrences and edge a''_i to the occurrence of its negation $\bar{x}_i$ (resp. x_i).

Clauses. For each clause C_j, $j = 1, 2, \ldots, m$, construct a triangle whose vertices represent its literals. When the clause has only two literals, we simply add an artificial vertex, since it has no influence on any assignment of *true* values. To finish the reduction, let $k = 3n$. Figure 1b illustrates the whole reduction. Next, it is shown that F is satisfiable if and only if G has an edge-triangle-transversal of size exactly $3n$.

First, suppose F is satisfiable. Let X be an initially empty transversal. If the variable x_i occurs twice with value *true* in F, we insert $\{b_i, b'_i, b''_i\}$ into X. Otherwise, we pick $\{a_i, a'_i, a''_i\}$. It is easy to see that every triangle in G is covered by X and that $|X| = 3n$.

Now, suppose X is an edge-triangle-transversal of G of size $3n$. Since each G_i has at most 3 independent triangles and G has n independent copies of G_i, the set X is irreducible. The smallest transversal of G_i can be either $\{a_i, a_i', a_i''\}$ or $\{b_i, b_i', b_i''\}$. If the first subset is selected, we assign *true* to the literals x_i occurring only once. Otherwise, we assign *true* to the literals x_i with double occurrence. □

Actually, Theorem 2 implies a stronger result that says the edge-triangle-transversal problem remains NP-complete even for planar graphs of maximum degree five. It is equally interesting that it remains a hard computational task when restricted to triangulations.

Theorem 3 *The edge-triangle-transversal problem is NP-complete for maximal planar graphs.*

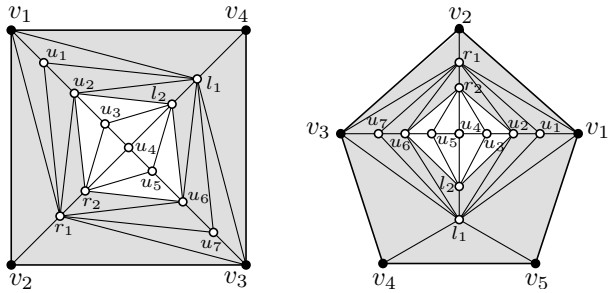
Proof. Let G be an arbitrary planar graph. The proof consists in transforming G into a maximal planar graph G' whose edge-triangle-transversal is trivially determined from any transversal for G.

The triangles in G are kept untouched, while all the other faces are triangulated as follows. Let $f = (v_1, v_2, \ldots, v_s)$ be a face in G of size $s > 3$. We begin by splitting f into two cycles towards a path joining the vertices v_1 and $v_{\lfloor s/2 \rfloor + 1}$. This path is composed by 8 edges, namely, (v_1, u_1), (u_1, u_2), $\ldots$, $(u_7, v_{\lfloor s/2 \rfloor + 1})$. This gives rise to two cycles C_r and C_l containing $\lfloor n/2 \rfloor + 8$ and $n - \lfloor n/2 \rfloor + 8$ vertices each. In the interior of cycle C_r (resp. C_l), we insert vertices r_1 and r_2 (resp. l_1 and l_2), which are then connected to the vertices $\{v_1, v_2, \ldots, v_{\lfloor s/2 \rfloor + 1}\} \cup \{u_1, u_2, u_6, u_7\}$ and $\{v_{\lfloor s/2 \rfloor + 1}, v_{\lfloor s/2 \rfloor + 2}, \ldots, v_1\} \cup \{u_2, u_3, \ldots, u_6\}$, respectively. Figure 2 illustrates this construction for 4- and 5-faces. All vertices, edges, and faces created during this step are called *false*. Otherwise, we call such entities *true*.

Let ℓ denote the number of non-triangular faces in G, and let G' be a maximal planar graph resulting from the construction process just described. Given any positive integer k, we claim that G has an edge-triangle-transversal of size at most k if and only if G' has an edge-triangle-transversal of size not exceeding $k + 2\ell$.

Suppose X is an edge-triangle-transversal for G of size k. Let $X' = X \cup E^*$, where E^* is the set of all edges of type (r_1, r_2) and (l_1, l_2) (see Fig. 2). We claim that X' is an edge-triangle-transversal of G'. First, observe that every false face has a vertex in E^*, while the true ones are touched by elements of X. Moreover, the following holds: $|X'| \leq |X| + |E^*| \leq k + 2\ell$.

We shall now assume that there exists an edge-triangle-transversal X' for G' of size at most $k + 2\ell$. Clearly, even if all true edges are selected, there will be uncovered faces (see unshaded areas in Fig. 2). The key observation is that, in any circumstance, we can always choose the edges belonging to E^*. Thus, true faces must be covered only by true edges. Hence $X = X' \setminus E^*$ is a transversal of G. Since E^* has exactly 2ℓ edges, the inequality $|X| \leq k$ holds. □



(a) 4-face. (b) 5-face.

Figure 2: Examples of constructions of triangulations for (a) 4-faces and (b) 5-faces. The shaded regions indicate the sole triangles covered by edges with end points at the outer cycle.

Now, the computational complexity of the edge guarding problem over polyhedral terrains turns out to be easily characterizable:

Theorem 4 *The edge guard problem for triangulated polyhedral terrains is NP-hard.*

Proof. Given a maximal planar graph G, we construct a terrain T as follows. For each face f in G, insert a point p at its centroid and connect it to the boundary vertices of f. Then, slightly translate p by h along the z direction, with $h < 0$, forming a small pit. For convenience, the new edges are labeled *false*. Otherwise they are called *true*.

Let X be an edge-triangle-transversal of G with size $k > 0$. We claim that the collection X is an edge-triangle-transversal of G if and only if it is also an edge guard set for T.

Since every pit in T can be entirely seen from its rim, the edges in X are always sufficient to cover the whole terrain T constructed as above. Suppose now that X' is an arbitrary guard set for T returned by some algorithm, for example, a standard greedy one. Mark the edges in X' as true or false. Note that any true edge in X' covers at least the same number of faces as a false one. Thus, it is likely that an optimal solution for T consists only of true edges. Otherwise, observe that we can always replace a false edge in X' by any true edge incident to it. Hence $X = X'$ is an edge-triangle-transversal of G. □

Lower bound. In Refs. [2] and [3], it was argued whether it would be possible to reduce the gap between sufficiency and necessity for edge guarding triangulated terrains. Alternatively, whether there

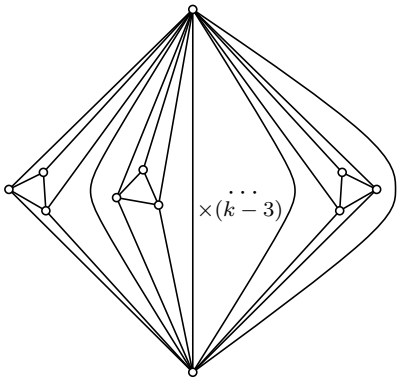


Figure 3: Planar triangulation with $n = 3k+2$ vertices that requires $(n-2)/3$ edge guards, where k is the number of independent triangles.

would be a planar triangulation on 9 vertices requiring exactly 3 edge guards, because the gap between $\lfloor(4n-4)/13\rfloor$ and $\lfloor n/3\rfloor$ is only stressed for graphs with more than 8 vertices. A brute-force solution would be to enumerate all planar triangulations on n vertices, to compute all possible edge guard sets for each one of them, and check if their sizes are all above $\lfloor(4n-4)/13\rfloor$. We have observed, however, that the two-connected planar graph presented in [2] for proving the best known lower bound so far could be triangulated without the addition of new vertices, and thus extending this result to planar triangulations:

Theorem 5 *There exists a maximal planar graph on n vertices, with $n \equiv 2 \mod 3$, that requires $\lfloor n/3\rfloor$ edge guards.*

Proof. We proceed by modifying the graph presented in [2, Fig. 6]. It is composed by k disjoint triangles arranged side-by-side, and two vertices, one above and the other below the base line where these triangles are placed. Additionally, we insert $k-1$ edges linking the upper and the lower vertices, passing through the regions between pairs of consecutive triangles. The resulting maximal planar graph G is composed by $3k+2$ vertices, as shown in Fig. 3. Since G has a maximum of k independent triangles, the size of any edge guard set is at least $(n-2)/3$. □

4 Conclusion

Recent results in graph theory [8] have motivated us to provide a purely combinatorial proof for the edge guarding problem. In fact, our results also extend to the vertex guard version, after contracting some edges in the gadgets we have designed.

In the proof of Theorem 2, we have produced a planar graph whose maximum degree does not exceed 5. An interesting open question is whether there exists an equivalent bound for maximal planar graphs.

Acknowledgments

We would like to thank the anonymous referees for their careful work and helpful comments.

References

[1] P. Bose. A note on the lower bound of edge guards of polyhedral terrains. *International Journal of Computer Mathematics*, 86(4):577–583, 2009.

[2] P. Bose, T. Shermer, G. Toussaint, and B. Zhu. Guarding polyhedral terrains. *Computational Geometry*, 7(3):173–185, 1997.

[3] P. Bose, D. Kirkpatrick, and Z. Li. Worst-case-optimal algorithms for guarding planar graphs and polyhedral surfaces. *Computational Geometry*, 26(3): 209–219, 2003.

[4] V. Chvátal. A combinatorial theorem in plane geometry. *Journal of Combinatorial Theory, Series B*, 18: 39–41, 1975.

[5] R. Cole and M. Sharir. Visibility problems for polyhedral terrains. *Journal of Symbolic Computation*, 7 (1):11 – 30, 1989.

[6] H. Everett and E. Rivera-Campo. Edge guarding polyhedral terrains. *Computational Geometry*, 7(3): 201–203, 1997.

[7] S. Fisk. A short proof of Chvátal's watchman theorem. *Journal of Combinatorial Theory, Series B*, 24: 374, 1978.

[8] M. Groshaus, P. Hell, S. Klein, L. T. Nogueira, and F. Protti. Cycle transversals in bounded degree graphs. *Electronic Notes in Discrete Mathematics*, 35:189 – 195, 2009.

[9] B. Kaučič, B. Žalik, and F. Novak. On the lower bound of edge guards of polyhedral terrains. *International Journal of Computer Mathematics*, 80(7):811–814, 2003.

[10] D. Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982.

[11] C. Tovey. A simplified NP-complete satisfiability problem. *Discrete Applied Mathematics*, 8(1):85–89, 1984.

[12] M. Yannakakis. Node-and edge-deletion NP-complete problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing*, pages 253–264, 1978.

[13] B. Zhu. *Computational geometry in two and a half dimensions*. PhD thesis, McGill University, Montreal, Canada, 1994.

From invariants to predicates: example of line transversals to lines

Guillaume Batog*

Abstract

This work explores a method that reduces the design of evaluation strategies for geometric predicates to the computation of polynomial invariants of a group action. We apply it to the classical problem of counting line transversals to lines in $\mathbb{P}^3$ and capture polynomials previously obtained by more pedestrian approaches.

1 Introduction

In computational geometry, algorithms are often designed over the reals but are implemented in floating-point arithmetic, which may lead to inconsistent decisions. To ensure correctness, refinement strategies or exact computations can be used but they may become very time–consuming, depending on the *evaluation strategy* of the decision problem or *predicate*. Consider for example the problem of deciding if four points in the plane are cocyclic. One approach is to compute the circumscribed circle of three points and test if the fourth point lies on it. Another approach consists in testing the vanishing of the determinant

$$\begin{vmatrix} 1 & x_1 & y_1 & x_1^2+y_1^2 \\ 1 & x_2 & y_2 & x_2^2+y_2^2 \\ 1 & x_3 & y_3 & x_3^2+y_3^2 \\ 1 & x_4 & y_4 & x_4^2+y_4^2 \end{vmatrix}$$

where (x_i, y_i) are the coordinates of the points. A major question is to find *efficient* and *robust* evaluation strategies for a given predicate.

We are here interested in strategies involving only polynomial evaluations from the inupts of a predicate. Robustness issues are guaranted through exact computation paradigm [10] and efficiency can be improved by using simplest possible polynomials. An immediate approach to find such polynomials consists in translating the problem into equations and extracting polynomial constraints that characterize the solutions of the resulting system. This has to be carried out carefully in order to avoid polynomials of huge degrees. Consider for example the problem of counting line transversals to four lines of $\mathbb{R}^3$ given as pairs of points. There may be 0, 1, 2 or infinitely many ones. Indeed, consider the ruled quadric generated by three input lines: the fourth line intersects it in at most two points or is contained in it. While the naive approach gives polynomial of degree 24 [3], the predicate can be decided with polynomials of degree at most 12 [2]. This gap can become more substantial: for ordering planes through a line ℓ, each containing a line transversal to three lines and ℓ, degree 144 in [3] collapses to degree 36 in [2]. These two *ad-hoc* approaches provide polynomials whose "complexity" strongly depends on the analytical formulation of the problem.

A general approach mainly based on the geometry of the predicate would be more satisfying. From this perspective, Petitjean [8] proposed an invariant–based method he applied to the problem of deciding the real intersection type of two projective planar conics (four simple points, two double points, a quadruple point,*etc* there are altogether 12 different types). What are the symmetries of the problem? Given two conics, observe that their intersection type is left unchanged under any simultaneous projective transformation of the two conics. The same is true when exchanging both conics or, more generally, replacing their equations by linear combinations of them. All of these symmetries are structured in a *group* that *acts* on the set of pairs of conics: any element of the group maps any pair of conics to another pair with the same intersection type. All pairs of conics obtained in this way from a fixed pair form an *orbit* of the group action. Invariant theory provides polynomial *invariants* that discriminate these orbits, and therefore distinguish intersection types.

In this work, we unfold the invariant–based method of [8] on the problem of counting line transversals to four linearly independent lines in $\mathbb{P}^3$. It provides the same polynomial of degree 12 in [2] in a more geometric manner and yields a better understand of the geometry of the problem. In this article, we focus on the construction of an appropriate group action and leave apart the computation of its polynomial invariants. For the latter problem, the interested reader will find an introduction in [6] and two different techniques in [4] and [5] among various existing strategies.

2 Preliminaries

Notations. The general linear group GL_n over $\mathbb{R}$ is the set of real invertible matrices of size n. (By extension, $GL(W)$ is the set of invertible linear trans-

*LORIA–Nancy 2 Univ., VEGAS Project, batog@loria.fr

formations of the vector space W.) We denote by $\mathbb{P}^n = \mathbb{P}^n(\mathbb{R})$ the real projective space of dimension n whose points are represented by homogeneous coordinates $[x_0 : \ldots : x_n]$. (By extension, $\mathbb{P}W$ represents the quotient of the vector space W by nonzero scalings.) A *collineation* (or *projective transformation*) of $\mathbb{P}^n$ is defined by a matrix M of GL_{n+1}: it maps $[x_0 : \ldots : x_n]$ to $[Mx_0 : \ldots : Mx_n]$. The set of hyperplanes of $\mathbb{P}^n$ is denoted by $\mathbb{P}^{n\star}$. The *duality operator* $\star : \mathbb{P}^n \to \mathbb{P}^{n\star}$ maps a point $[x_0 : \ldots : x_n]$ to the hyperplane defined by the equation $\sum_{i=0}^{n} x_i y_i = 0$. A *correlation* of $\mathbb{P}^n$ is the composition of a collineation with the duality operator.

2.1 Invariants of group actions

A *transformation group* is a subset of GL_n containing the identity matrix and closed by multiplication. In what follows, G will denote an abstract group but it is sufficient to restrict to transformation groups for the sake of understanding.

Group action. The *action* ρ of a group G on a set X is denoted by $\rho : G \circlearrowright X$ and is defined as follows: all $\rho(g)$ with $g \in G$ are bijections of X such that $\rho(1)$ is the identity map on X and $\rho(gg') = \rho(g) \circ \rho(g')$ for any $g, g' \in G$. For example, the group of isometries preserving a cube acts on the set of diagonals of that cube: applying two successive isometries on the cube induces a composition of two permutations of its diagonals. A *linear group action* of G on a vector space W is a group action ρ of G on W where the bijections $\rho(g)$ on W are linear[1] (*i.e.* elements of $GL(W)$). We denote it by $\rho : G \to GL(W)$.

Consider a fixed element x in X and form the set of all $y \in Y$ that can be obtained from x by a map $\rho(g)$ (with $g \in G$): this defines an *orbit* of ρ. These orbits form a partition of X. In the previous example, there is just one orbit because any diagonal of the cube can be mapped to any other one by an isometry preserving the cube. Let us give two another examples.

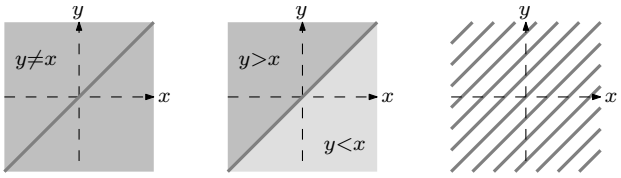
Example 1. Let G be the group of affine motions[2] of the real line $\mathbb{R}$ and $\rho : G \circlearrowright \mathbb{R}^2$ its action on pairs of points defined by $\rho(g)(x, y) = (g(x), g(y))$. Figure 1 represents the orbits of ρ where we restrict to different subgroup of G. We observe that the smaller (for inclusion) the group, the larger the number of orbits.

Example 2. Consider the action S_2^2 of GL_2 on the space $S^2(\mathbb{R}^2)$ of binary quadratic forms, defined by:

$$S_2^2 \left(\begin{smallmatrix} \alpha & \beta \\ \gamma & \delta \end{smallmatrix}\right)^{-1} (ax^2 + 2bxy + cy^2) = \bar{a}x^2 + 2\bar{b}xy + \bar{c}y^2$$

[1] W is a *representation* of the group G in other words.

[2] An affine motion g of $\mathbb{R}^n$ is represented by a matrix of GL_{n+1} in the form $\left(\begin{smallmatrix} 1 & 0 \\ \vec{t} & M \end{smallmatrix}\right)$ where $\vec{t}$ is the translation vector of g and $M \in GL_n$ its vector part. We define $\det g = \det M$.



(a) affine motions (b) non-reversing affine motions (c) rigid motions

Figure 1: Orbits of ρ from Example 1.

$$\text{where} \quad \begin{cases} \bar{a} = \alpha^2 a + 2\alpha\gamma b + \gamma^2 c \\ \bar{b} = \alpha\beta a + (\alpha\delta + \beta\gamma)b + \gamma\delta c \\ \bar{c} = \beta^2 a + 2\beta\delta b + \delta^2 c \end{cases} .$$

It simply consists of a change of coordinates induced by $g \in GL_2$ on the quadratic form. This action has three orbits, depending on the number of distinct factors in which a quadratic form can be factored.

Invariant. Let $\rho : G \to GL(W)$ be a linear group action of a transformation group $G \subset GL_n$. A homogeneous polynomial P on W is a *(relative) invariant* for ρ if there exists $\lambda \in \mathbb{Z}$ such that

$$\forall\, (g, w) \in G \times W \quad P(\rho(g)(w)) = (\det g)^\lambda\, P(w). \quad (1)$$

Some properties of an invariant remains unchanged on each orbit, as the previous two examples illustrate.

In Example 1, the polynomial $P(x, y) = x - y$ is invariant for ρ (with $\lambda = 1$) since $g(x) - g(y) = (\det g)(x - y)$. If we restrict G to rigid motions (for which $\det g = 1$), the invariant P is constant on each orbit and its value discriminates the orbits.

In Example 2, a straightforward computation shows that the discriminant is a polynomial invariant on $S^2(\mathbb{R}^2)$ (with $\lambda = 2$):

$$\bar{b}^2 - \bar{a}\bar{c} = (\alpha\delta - \beta\gamma)^2(b^2 - ac).$$

We observe that the sign $(+, -, 0)$ of this invariant is constant on each orbit and it entirely characterizes the orbits.

Covariant. A *covariant* for $\rho : G \to GL(W)$ is a polynomial invariant C for some action $\rho' : G \to GL(W \times (\mathbb{R}^n)^m)$ defined by $\rho'(g)(w, x_1, \ldots, x_m) = (\rho(g)(w), g(x_1), \ldots, g(x_m))$. We write $C \sim_w 0$ for $w \in W$ if $C(w, x_1, \ldots, x_m) = 0$ for all $(x_1, \ldots, x_m) \in (\mathbb{R}^n)^m$. By definition, either $C \sim 0$ or $C \nsim 0$ on a whole orbit.

2.2 Line geometry

Plücker quadric. A line ℓ of $\mathbb{P}^3$ can be represented by its (homogeneous) Plücker coordinates $\xi = [\xi_0 : \cdots : \xi_5]$ that fullfill the quadratic Plücker relation

$$q(\xi) = \xi_0\xi_3 + \xi_1\xi_4 + \xi_2\xi_5 = 0. \quad (2)$$

It is the equation of a quadric $\mathbb{G}$ of $\mathbb{P}^5$ called the *Plücker quadric*. We denote by $\gamma(\ell)$ the Plücker coordinates of a line ℓ. For a line ℓ in $\mathbb{R}^3$, $\vec{v} = (\xi_0, \xi_1, \xi_2)$ is a direction of ℓ and (ξ_3, ξ_4, ξ_5) the moment of $\vec{v}$ with respect to the origin of $\mathbb{R}^3$. A complete presentation can be found in [9].

Span of lines. For a subset $H \subset \mathbb{P}^5$, $\operatorname{span} H$ is the minimal (for inclusion) projective subspace of $\mathbb{P}^5$ containing H. We define the *span* of a family of lines as the span of their Plücker coordinates. A family of k lines is said *linearly independent* if its span has dimension $k-1$. Any set $\mathcal{L}$ of lines contains a family $\mathcal{L}'$ with at most six linearly independent lines and any line of $\mathcal{L}$ is linearly dependent of those of $\mathcal{L}'$.

Conjugation. The quadric q defines a bilinear form $\odot$ called *side-operator*. We can observe that two lines ℓ and ℓ' meet if and only if $\gamma(\ell) \odot \gamma(\ell') = 0$ ([9]). Given a set H of $\mathbb{P}^5$, we define its *conjugate* as

$$H^\circ = \{x \in \mathbb{P}^5 \mid \forall h \in H \quad x \odot h = 0\}.$$

Geometry of quadratic forms [1, 13.3] shows that H° is a subspace of $\mathbb{P}^5$ of codimension $\dim(\operatorname{span} H)$ and it satisfies $(H^\circ)^\circ = \operatorname{span} H$. In terms of transversality, we immediately have

Observation 1 *$\gamma^{-1}(H^\circ \cap \mathbb{G})$ is the set of line transversals to all of the lines $\gamma^{-1}(H \cap \mathbb{G})$.*

Transformations preserving $\mathbb{G}$. We here consider $\mathbb{G}$ as a homogeneous subset of $\mathbb{R}^6$. A transformation $M \in GL_6$ globally preserves $\mathbb{G}$ if and only if there is $\mu \in \mathbb{R}^*$ such that $q(Mx) = \mu q(x)$ for any $x \in \mathbb{R}^6$ ([7, V.7]). Such transformations form a group $GO_6(q)$ called the *similarity group of q*. The subgroup of similarities M such that $\mu = 1$ and $\det M = 1$ is called the *rotation group of q* and is denoted by $SO_6(q)$.

Since a projective transformation g maps lines to lines and preserves incidences, it naturally induces a bijection of $\mathbb{G}$. The same is true for correlations. In fact ([9, Theorem 2.2.1]), such a bijection extends to a projective transformation $\wedge_4^2 g$ of $\mathbb{P}^5$ where $\wedge_n^k M$ is the k^{th} *compound matrix* of the matrix M of size n whose entries are the minors of size k of M.

Lemma 1 [9, Theorem 2.1.10] *$\mathbb{P}GO_6(q)$ is exactly the set of transformations of $\mathbb{P}^5$ induced through $\wedge_4^2$ by collineations and correlations of $\mathbb{P}^3$.*

3 Invariant–based method step by step

In this section, we unfold the invariant–based method for the following predicate: given the Plücker coordinates of four linearly independent lines, how many lines intersect all of them? We denote by X the set of inputs of a predicate.

Step 1: *Find all symmetries of any kind on the inputs X that leave invariant the outputs of the predicate and model them by a group G acting on X by $\psi : G \circlearrowright X$.*

Here, the inputs of the predicate are quadruplets $(\xi_1, \ldots, \xi_4)$ of linearly independent lines (X is an open subset of $\mathbb{G}^4$). Observe first that the order in which the input lines are considered does not matter, hence we can consider the action ψ_1 of the permutation group $\mathfrak{S}_4$ on X defined by

$$\psi_1(\sigma)(\xi_1, \ldots, \xi_4) = (\xi_{\sigma(1)}, \ldots, \xi_{\sigma(4)}).$$

Since a projective transformation preserves incidences between lines, the action of $\mathbb{P}GL_4$ on X defined by $\wedge_4^2$ leaves the output of the predicate invariant on an orbit. In other words, any change of coordinates does not change the number of line transversals to the input lines. By this process, lines are considered as *intrinsic* geometric objects. In the same way, we can consider the action of correlations that also preserves incidences between lines. According to Lemma 1, the action of collineations and correlations writes as $\psi_2 : \mathbb{P}GO_6(q) \circlearrowright X$ defined by

$$\psi_2(g)(\xi_1, \ldots, \xi_4) = (g(\xi_1), \ldots, g(\xi_4)).$$

Altogether, we construct $G = \mathfrak{S}_4 \times \mathbb{P}GO_6(q)$ and $\psi : G \circlearrowright X$ defined by $\psi(\sigma, g) = \psi_1(\sigma) \circ \psi_2(g)$. In the point of view of Erlangen's program, ψ encodes the geometry of "*sets* of four lines", that is, we identify two *ordered* families of line coordinates if they represent the same set of lines. At this step, our method differs from other approaches based on manipulations of coordinates, here only geometry matters.

Step 2: *Construct an encoding $\pi : X \to Y$ and a group action $\rho : G \circlearrowright Y$ with finitely many orbits in $\pi(X)$ and "simulating" ψ on Y, i.e.*

$$\forall\, (g, x) \in G \times X \quad \rho(g)(\pi(x)) = \pi(\psi(g)(x)).$$

Hence the predicate has the same output on x and x' if $\pi(x)$ and $\pi(x')$ are in the same orbit of ρ.

According to Observation 1, the line transversals to an input line family $x \in X$ are exactly those of the span H of x, that is, $H^\circ \cap \mathbb{G}$. Since the four lines of x are linearly independent, H has dimension 3, thus H° has dimension one: it is a line of $\mathbb{P}^5$. As $\mathbb{G}$ is a quadric in $\mathbb{P}^5$, either H° is contained in $\mathbb{G}$ or H° intersects $\mathbb{G}$ in at most two points. The corresponding quadrics $H \cap \mathbb{G}$ are listed in Table 1. We observe that the type of $H \cap \mathbb{G}$ entirely characterizes the number of line transversals to the family x. So we consider the encoding $\pi : x \mapsto H$ that maps a line family x to its span and Y the set $\mathbb{G}_{4,6}$ of 3-dimensional subspaces of $\mathbb{P}^5$. We can show that $\pi(X) = Y$.

$H^\circ \cap \mathbb{G}$	$H \cap \mathbb{G}$	$q_{\|H}$
2 points	hyperboloid	$(2,2)$
0 point	ellipsoid	$(3,1)$ or $(1,3)$
1 point	cone	$(2,1)$ or $(1,2)$
a line	two planes sharing a line	$(1,1)$

Table 1: Types of spans of four linearly independent lines. The third column denotes the inertia of the restriction to H of the quadratic form q given in (2).

Let us "simulate" ψ on Y. Since $\pi(\psi_1(\sigma)(x)) = \pi(x)$ for any $x \in X$, the action of $\mathfrak{S}_4$ has no effect on Y thus we can remove this group from G. We construct $\rho : \mathbb{P}GO_6(q) \circlearrowleft \mathbb{G}_{4,6}$ defined by $\rho(g) = \wedge_6^4 g$. By Witt's Theorem [1, 13.7.1 and 13.7.9], the orbits of ρ restricted to the group $\mathbb{P}SO_6(q)$ (Figure 2b) are characterized by the inertia of the quadric defined by $H \cap \mathbb{G}$ (see Table 1). Since a similarity with negative multiplicator μ change the sign of q (Equation (2)), the orbits of ρ (Figure 2a) are obtained by merging the previous orbits with the same *unsigned* inertia.

Step 3: *Use appropriate techniques to compute some polynomial invariants of ρ.*

Here, we consider $\rho' : SO_6(q) \to GL(\mathbb{R}^{15})$ (Y is an homogenous subset of $\mathbb{R}^{15}$) defined by $\rho'(g) = \wedge_6^4 g$. Using the symbolic method of [4], we obtain[3] a polynomial invariant of degree 2:

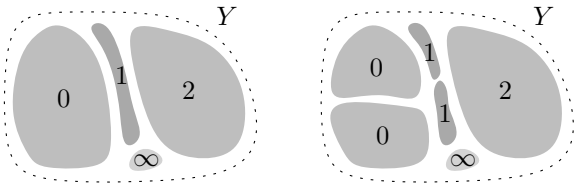
$$\Delta = y_4^2 + y_8^2 + y_{13}^2 - 2y_1y_{10} - 2y_{14}y_2 - 2y_3y_{15} + 2y_7y_5 + 2y_{11}y_6 + 2y_{12}y_9$$

and a covariant $Cov(y, x, x')$ defined on $Y \times (\mathbb{R}^6)^2$ with 21 distinct coefficients in x, x' of degree 2. Since Δ is a homogenous polynomial of degree 2, its sign remains unchanged up to nonzero scalings, thus is invariant on each orbit of $\rho : \mathbb{P}SO_6(q) \circlearrowleft Y$. Since Cov is homogeneous, Cov is a covariant of ρ.

Step 4: *Evaluate the previous polynomials on some representative of each orbit and observe if geometric situations are discriminated.*

Finally, we obtain the following algorithm for counting line transversals to a family x of four linearly independent lines. We compute $y = \pi(x)$. If $\Delta(y) > 0$, there are 2 line transversals. If $\Delta(y) < 0$, there is no transversal. Otherwise, if $Cov \sim_y 0$, then there are infinitely many transversals, else there is only one.

[3] In a symbolic form, Δ is written as the bracket polynomial $[\alpha^{(4)}ab][\beta^{(4)}ab]$ and Cov as $[\alpha^{(4)}au][\beta^{(4)}av]$ where α, β are letters representing $\mathbb{R}^{15} = \Lambda^4\mathbb{R}^6$, a, b representing $S^2\mathbb{R}^6$ (it simulates $SO_6(q) \subset GL_6$) and u, v representing $\mathbb{R}^6$.



(a) $G = \mathbb{P}GO_6(q)$ (b) $G = \mathbb{P}SO_6(q)$

Figure 2: Orbits of ρ.

4 Conclusion

For counting line transversals to four linearly independent lines, our invariant–based method provides the same polynomial Δ as [2] but polynomials of higher degrees than in [2] to discriminate the degenerate cases. The same technique applies for five lines and gives rise to the same polynomial as [2]. Finally, some polynomials involved in predicates appear as invariants of group actions, that is they originate from the geometry of the problem. They might be essential in any evaluation strategy for a predicate, based on polynomials. This point of view seems to be a promising approach to tackle optimality questions on predicates.

References

[1] M. Berger. *Geometry II.* Springer, 1996.

[2] O. Devillers, M. Glisse, and S. Lazard. Predicates for line transversals to lines and line segments in three-dimensional space. In *Proc. 24th ACM Symp. Computat. Geom.*, pages 174–181, 2008.

[3] H. Everett, S. Lazard, B. Lenhart, and L. Zhang. On the degree of standard geometric predicates for line transversals in 3D. *Computational Geometry: Theory and Applications*, 42(5):484–494, 2009.

[4] F. D. Grosshans, G.-C. Rota, and J. A. Stein. *Invariant Theory and Superalgebras*, volume 69 of *CBMS Region. Conf. Ser. Math.* AMS, 1987.

[5] E. Hubert and I. A. Kogan. Rational invariants of a group action. Construction and rewriting. *Journal of Symbolic Computation*, 42:203–217, 2007.

[6] P. J. Olver. *Classical Invariant Theory*, volume 44 of *London Mathematical Society Student Text.* Cambridge University Press, 1999.

[7] D. Perrin. *Cours d'algèbre.* Ellipses, 1996.

[8] S. Petitjean. Invariant-based characterization of the relative position of two projectives conics. In I. Z. Emiris, F. Sottile, and T. Theobald, editors, *Nonlinear Computational Geometry*, volume 151 of *The IMA Volumes in Mathematics and its Applications*, pages 189–220. Springer, 2009.

[9] H. Pottmann and J. Wallner. *Computational Line Geometry.* Springer-Verlag, 2001.

[10] C. K. Yap. Robust geometric computation. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 41, pages 927–952. Chapmen & Hall/CRC, 2nd edition, 2004.

On the Diameter of a Geometric Johnson Type Graph.*

C. Bautista-Santiago† J. Cano† R. Fabila-Monroy‡ D. Flores-Peñaloza† H. González-Aguilar†
D. Lara† E. Sarmiento‡ J. Urrutia†

Abstract

An island of a set S of points on the plane is a subset I of S with the property that $\text{Conv}(I) \cap S = I$. In this paper we introduce the (k,l)-*island intersection graph* of S, $J(S,k,l)$, as the graph whose vertex set is the set of all islands of S of cardinality k, where two of them are adjacent if their intersection consists of exactly l elements. For sets of points in general position, we show that if n is large enough with respect to k and l, then $J(S,k,l)$ is connected; we also give upper and lower bounds on the diameter of this graph.

1 Introduction

Let S be a set of n points on the plane. A subset $I \subseteq S$ is called an *island* if $\text{Conv}(I) \cap S = I$. We say that an island is a k-*island* if it has k elements.

Given two integers k and l we define the (k,l)-*island intersection graph of* S, $J(S,k,l)$, to be the graph whose vertex set is the set of all k-islands of S; where two islands are adjacent if their intersection has exactly l elements.

Various problems in Combinatorial Geometry can be restated as the problem of determining some graph-theoretic property of $J(S,k,l)$. For example, the number of empty triangles in S [2, 3, 4, 7, 9] is the number of vertices in $J(S,3,l)$. In [1] the following question is posed: what is the maximum number of empty triangles that can share an edge? This translates to the problem of determining the clique number of $J(S,3,2)$.

A related graph $J(n,k,l)$ (see [5]) has long been studied. This graph has as vertices all the subsets of k elements of a given set of cardinality n, two of which are adjacent if their intersection has l elements. The particular cases of $J(n,k,0)$ and $J(n,k,k-1)$ are the well known Kneser and Johnson graphs.

When S is in convex position, every subset of S with k elements is a k-island. Thus, in this case, $J(S,k,l)$ is isomorphic to $J(n,k,l)$. For sets of points not in convex position, $J(S,k,l)$ is an induced subgraph of $J(n,k,l)$. We may regard $J(S,k,l)$ as a *geometric* version of $J(n,k,l)$.

The paper is organized as follows: In Section 2 we give a sufficient condition for $J(S,k,l)$ to be connected. To do so, we first consider the case when all the points are on a line, and then the case when all but one point are on a line. In the first case $J(S,k,l)$ is disconnected for all but trivial cases; somewhat surprisingly, in the second case $J(S,k,l)$ is connected provided that n is large enough with respect to k and l.

In Section 3, we show that when S is in general position, $J(S,k,l)$ contains as a subgraph the case of all but one point on a line. We then show that every k-island is connected by a path to this subgraph, and thus prove that $J(S,k,l)$ is connected when n is large enough with respect to k and l.

The upper bound on the diameter of $J(S,k,l)$ implied by the connectivity results, is improved in Section 4 for the particular case when $l \leq k/2$. In the same section we conclude the paper by giving lower bounds for the diameter of $J(S,k,l)$.

We omit several proofs for lack of space. We indicate that the proof of one result is omitted, with the addition of a box at the end of its statement.

2 Collinear and almost collinear points

In this section we consider two classes of point sets: when all the points are collinear, and when all the points but one are on a line.

2.1 Collinear points

Let L be a set of n points on a straight line. Assume an orientation of this line and let $x_1, \ldots, x_n$ be the elements of L in the order induced by the line. In this case, a k-island of L is a set of k consecutive elements of L.

We say that a point $x_i \in L$ is to the *left* of an island $I \subseteq L$ if its index i is smaller than the index of each point in I.

We now show that $J(L,k,l)$ is the disjoint union of paths. To see this, let P_r be the subgraph of $J(L,k,l)$

*Part of the work was done in the 2nd Workshop on Discrete Geometry and its applications. Oaxaca, México, September 2009.

†Instituto de Matemáticas, Universidad Nacional Autónoma de México. {crevel,dlara,j_cano}@uxmcc2.iimas.unam.mx, {hernan,dflores,urrutia}@matem.unam.mx, partially supported by CONACyT (Mexico) grant CB-2007/80268.

‡Departamento de Matemáticas, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, México. {ruyfabila,esarmiento}@math.cinvestav.edu.mx.

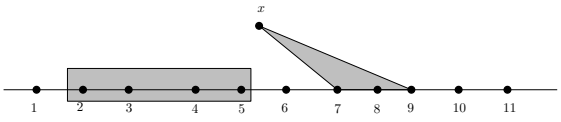


Figure 1: Two 4-islands of $L \cup \{x\}$.

induced by the set of all k-islands such that the number of points of L to its left congruent to r modulo $(k-l)$. We label the islands in P_r by $A_{r,i}$, where $A_{r,i}$ is the island with $i(k-l)+r$ points in L to its left, $0 \leq r < k-l$. Clearly each k island of L is an $A_{r,i}$ island, for some i and r. Observe now that each k-island of L has at most two neighbours in $J(L,k,l)$: one to its left and one to its right. It is easy to see that $J(L,k,l)$ is a union of disjoint paths: the paths P_r, for $0 \leq r < k-l$. We thus have:

Theorem 1 *The graph $J(L,k,l)$ is the disjoint union of all P_r $(0 \leq r < k-l)$.*

Note that $J(L,k,l)$ is connected when $n = k$ or $l = k-1$, and disconnected otherwise. Remarkably, the addition of one extra point makes the graph connected, as we show next.

2.2 Almost collinear points

Let L be a set of $n-1$ points on a straight line, and let x be a point outside the line containing L, and let $L' := L \cup \{x\}$. A k-island of L' is either a k-island of L, or a $(k-1)$-island of L together with the point x, see Figure 1. Note that $J(L,k,l)$ is an induced subgraph of $J(L',k,l)$.

Let P_r be the subgraph of $J(L',k,l)$ induced by the set of all k-islands that *do not* contain x and have a number of points of L to its left congruent to r modulo $(k-l)$. We label the islands in P_r as before: The island $A_{r,i}$ in P_r is the one having exactly $i(k-l)+r$ points to its left. Similarly, let P'_r, be the subgraph of $J(L',k,l)$ induced by the set of all k-islands that *contain* x and leave a number of points of L to its left congruent to r modulo $(k-l)$. Denote as $A'_{r,i}$ the island in P'_r having exactly $i(k-l)+r$ points to its left. The following lemma, given without proof, characterizes the edges of $J(L',k,l)$, as can be seen in Figure 2.

Lemma 2 *In $J(L',k,l)$:*

1. *$A_{r,i}$ is adjacent to $A_{r,i+1}$ and $A'_{r,i+1}$.*
2. *$A'_{r,i}$ is adjacent to $A'_{r,i+1}$, and to $A_{r-1,i+1}$, if $r \geq 1$, or to $A_{k-l-1,i}$, if $r = 0$.*

Furthermore, each edge of $J(L',k,l)$ falls in one of these types of adjacencies.

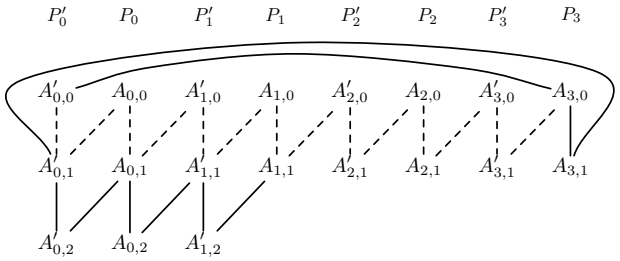


Figure 2: The graph $J(L',6,2)$ with $|L'| = 15$. Dashed edges illustrate the path used in Theorem 3.

With the previous lemma, it is easy to prove the connectivity of $J(L',k,l)$ for any sufficiently large n.

Theorem 3 *For $n \geq 3k-2l-1$, the graph $J(L',k,l)$ is connected.*

Proof. Note that by Lemma 2, the subgraphs P_r and P'_r, $0 \leq r \leq k-l-1$, are paths, and thus are connected.

Observe that if $n \geq k + (2k-2l-1)$, then the following islands exist:

- the $k-l$ islands $A_{r,0}$, with $0 \leq r < k-l$,
- the $k-l-1$ islands $A_{r,1}$, with $0 \leq r < k-l-1$,
- the $k-l$ islands $A'_{r,0}$, with $0 \leq r < k-l$, and
- the $k-l$ islands $A'_{r,1}$, with $0 \leq r < k-l$.

Let $\pi = A'_{0,0}A'_{0,1}A_{0,0}A_{0,1} \cdots A'_{k-l-1,0}A'_{k-l-1,1}A_{k-l-1,0}$. By Lemma 2, path π is in $J(L',k,l)$, see Figure 2. Since π contains at least one vertex of each P_r and P'_r, $0 \leq r < k-l$, it follows that $J(L',k,l)$ is connected. □

It can be proved that the graph $J(L',k,l)$ is disconnected whenever $n < 3k-2l-1$. Furthermore, the next bound on the diameter of $J(L',k,l)$ follows directly from Lemma 2.

Proposition 4 *The diameter of $J(L',k,l)$ is at most $\frac{n-k}{k-l} + 2(k-l) + 1$.* □

3 Points in General Position

Let S be a set of n points in general position in the plane and let x be its topmost point. Sort the remaining points by angle around x, and denote them as $x_1, \ldots, x_{n-1}$. Note that a set of consecutive points in this order is an island, as is a set of consecutive elements together with x. We call both of these types of islands *projectable*.

Lemma 5 *The subgraph of $J(S,k,l)$ induced by the projectable islands contains $J(L',k,l)$ as a subgraph.*

Note that if n is large enough with respect to k and l, then the subgraph of $J(S,k,l)$ induced by the projectable islands is connected (Theorem 3).

In order to show that $J(S,k,l)$ is connected, we exhibit a path in $J(S,k,l)$ from any island to a projectable island.

Let i be the weight of x_i, and define the *weight* of an island with at least two elements as the difference between the largest and the smallest indexes of its elements (excluding x). An island of weight $k-1$ is always projectable.

Lemma 6 (Shrinking Lemma) *If $n > (k-l)(k-l+1)+k$, then every non projectable island A_i has a neighbour in $J(S,k,l)$, which is either a projectable island, or an island whose weight is at most the weight of A_i minus $k-l$.*

Proof. Let A_i be a k-island with its elements distinct from x being $x_{i_1},\ldots,x_{i_m}$ (ordered also by angle around x). Thus, m is equal to k or to $k-1$ depending on whether A_i contains x or not. Consider the following intervals of $S\setminus\{x\}$:

$$\begin{aligned}[x_1, x_{i_{l+1}}) &:= \{x_j \in S\setminus\{x\} | 1 \le j < i_{l+1}\},\\ (x_{i_{m-l}}, x_{n-1}] &:= \{x_j \in S\setminus\{x\} | i_{m-l} < j \le n-1\},\end{aligned}$$

and, for every h s.t. $1 \le h \le m-l-1$,

$$(x_{i_h}, x_{i_{h+l+1}}) := \{x_j \in S\setminus\{x\} | i_h < j < i_{h+l+1}\}.$$

We refer to the first two intervals as *end intervals*.

Note that there are at most $k-l+1$ such intervals, each containing exactly l elements of A_i, and that every element of $S\setminus\{x\}$ is in at least one interval.

Thus, since $n > (k-l)(k-l+1)+k$, one of these intervals, I, must contain at least $(k-l)$ points of $S\setminus A_i$.

If I is of the form $(x_{i_h}, x_{i_{h+l+1}})$, set $J := \{x_{i_{h+1}}, x_{i_{h+2}}, \ldots, x_{i_{h+l}}\}$ (if $l=0$, set $J:=\emptyset$), and set B to be the set of the $k-l$ points of $I\setminus A_i$ closest to $\mathrm{Conv}(J)$ (if $l=0$, set B to be any k-island inside $I\setminus A_i$). The k-island $J\cup B$ is a neighbour of A_i in $J(S,k,l)$, and its weight is smaller than the weight of A_i by at least $k-l$.

If I is an end interval, then let x_i and $x_{i'}$ be the first and last elements in $A_i\cap I$. If $[x_i, x_{i'}]$ contains at least $k-l$ elements of $S\setminus A_i$, then proceed as in the previous case. Otherwise, there are $r < k-l$ points of $S\setminus A_i$ in I. On one hand, if I is the first interval, then we take B to be the $k-l-r$ previous points to x_i in $S\cap A_i$. On the other hand, if I is the last interval, then we take as B the $k-l-r$ points after x'_i. Note that in either case, $[x_i, x_{i'}]\cup B$ is a projectable island adjacent to A_i. □

As a consequence of Lemma 5 and Proposition 4 we have the following result:

Theorem 7 *If $n > (k-l)(k-l+1)+k$, then $J(S,k,l)$ is connected and has diameter at most $3\frac{n}{k-l} - \frac{k}{k-l} + 2(k-l)+3$.*

Proof. Let A and B be k-islands. We apply Lemma 6 successively to find a sequence of consecutive adjacent islands $A := A_0, A_1, \ldots, A_m$ and $B := B_0, B_1, \ldots, B_{m'}$, in which each element has weight smaller than the previous by at least $k-l$, and the last element is a projectable island.

Since the weight of the initial terms is at most n, these sequences have length at most $n/(k-l)+1$.

As noted in Lemma 5 the subgraph of projectable islands contains $J(L',k,l)$ as a subgraph. Simple arithmetic shows that if $n > (k-l)(k-l+1)+k$, then $n > 3k-2l-2$. Thus, this subgraph is connected and has diameter at most $\frac{n-k}{k-l} + 2(k-l)+1$ (Theorem 3 and Proposition 4).

Hence, the diameter of $J(S,k,l)$ is at most $2(n/(k-l)+1) + \frac{n-k}{k-l} + 2(k-l)+1 = 3\frac{n}{k-l} - \frac{k}{k-l} + 2(k-l)+3$, as claimed. □

4 Diameter

4.1 Upper Bound

Theorem 7 yields an upper bound for the diameter of $J(S,k,l)$. For the case when $l \le k/2$ this bound can be greatly improved. At the moment, in the complementary case of $l > k/2$, we are unable to do better than Theorem 7.

Our general approach for finding short paths between two vertices A and B of $J(S,k,l)$, is to use a *divide and conquer* strategy; we find neighbours of A and B and at each step of the process, we discard half of the points.

We cannot do this indefinitely since $J(S,k,l)$ may be disconnected if S has few points. Just before arriving at such a situation, we use Theorem 7.

The following lemma provides the divide and conquer part of the argument. Its proof is technical and uses many of the same arguments as the proof of Lemma 6.

Lemma 8 *Let A and B be two k-islands of S. If $n \ge 2((k-l)(k-l+1)+k)$, and $l \le k/2$, then there exists a semiplane H containing at most $n/2$ and at least $(k-l)(k-l+1)+k$ elements of S, with the property that A and B, considered as vertices of $J(S,k,l)$, each has a neighbour contained entirely in H.* □.

Theorem 9 *If $n \ge 2((k-l)(k-l+1)+k)$ and $l \le k/2$, then the diameter of $J(S,k,l)$ is at most $2\log_2(n) - 2\log_2(((k-l)(k-l+1)+k)) + 5\frac{k}{k-l} + 8(k-l) + 7$.*

Proof. Consider the following algorithm.

Let A and B be two k-islands of S. We start by setting $A_0 := A$, $B_0 := B$, $S_0 := S$, $n_0 := n$, and $i = 0$.

While $n_i \geq 2((k-l)(k-l+1)+k)$, we apply Lemma 8 to S_i, A_i, and B_i. At each step we obtain the semiplane H_i containing at most $n_i/2$ and at least $(k-l)(k-l+1)+k$ elements of S_i, with the additional property that both A_i and B_i have neighbours A_{i+1} and B_{i+1} in $J(S_i,k,l)$ contained entirely in H_i. We set $S_{i+1} := H_i \cap S_i$, $n_{i+1} := |S_{i+1}|$, and $i := i+1$, and continue the iteration.

We can do this procedure at most $\log_2(n) - \log_2(((k-l)(k-l+1)+k)) - 1$ times.

In the last iteration, we have a point set S_j with less than $2((k-l)(k-l+1)+k)$ elements, and at least $(k-l)(k-l+1)+k$ elements. The islands A_j and B_j are both contained in S_j, and both are joined by paths of length at most $\log_2(n)-\log_2(((k-l)(k-l+1)+k))-1$ to A and B. We apply Theorem 7 to obtain a path of length at most $6(k-l+1)+5\frac{k}{k-l}+2(k-l)+3$ from A_i to B_i.

Concatenating the three paths we obtain a path of length at most $2\log_2(n) - 2\log_2(((k-l)(k-l+1)+k))+5\frac{k}{k-l}+8(k-l)+7$ from A to B in $J(S,k,l)$. □

4.2 Lower Bound

For the lower bound we use Horton sets [6]. We follow the notation used in [8].

Given two point sets X and Y in the plane, we say that X is *high above* Y if Y is below every line containing two elements of X. Conversely we say that Y is *deep below* X if X is above any line containing two points of Y.

Now let X be a set of n points in the plane such that not two of them define a vertical line, and let $x_1,\ldots,x_n$ be the elements of X sorted by their x-coordinate.

Define X_0 to be the subset of X containing the elements with *even* index, and X_1 the subset of elements of X with *odd* index.

Definition 1 *A finite point set H with no two of its elements on a vertical line, is a Horton set if $|H| \leq 1$, or the following conditions are met:*

- *Both H_0 and H_1 are Horton sets.*
- *H_0 is deep below H_1.*
- *H_1 is high above H_0.*

Horton sets of any size were shown to exists in [6]. Thus, let H be a Horton set of n points. Since by definition H_0 and H_1 are Horton sets, we may speak of H_{00}, H_{01}, H_{10}, and H_{11}. We will do so and in general speak of H_b, where b is a word of 0's and 1's.

To every k-island A of H, we associate the only set H_b with the property that H_b contains A but H_{b0} and H_{b1} do not.

The following lemma follows from the definition of H.

Lemma 10 *Let A and B be two neighbouring vertices in $J(H,k,l)$, and let H_a and H_b be, respectively, their associated sets. Then, a differs from b in at most $k-l$ letters.* □

Theorem 11 *The diameter of $J(H,k,l)$ is at least $\log_2(n)+\log_2(k)$.*

Proof. For the sake of clarity, assume that n is a power of 2. If b_0 and b_1 are the words of length $\log(n)+\log(k)$ with only 0's (resp. only 1's), then H_{b_0} and H_{b_1} are k-islands of H; by Lemma 10, they are at distance at least $\log_2(n)+\log_2(k)$ in $J(H,k,l)$. □

References

[1] I. Bárány, G. Károlyi, Problems and results around the Erdős-Szekeres convex polygon theorem. In *Discrete and computational geometry.* pp. 91–105. 2000.

[2] I. Bárány, P. Valtr, Planar point sets with a small number of empty convex polygons. *Studia Sci. Math. Hungar.* 41, pp. 243–266, 2004.

[3] I. Bárány, Z. Füredi, Empty simplices in Euclidean space. *Canad. Math. Bull.* 30, pp. 436–445, 1987.

[4] A. Dumitrescu, Planar sets with few empty convex polygons. *Studia Sci. Math. Hungar.* 36, pp. 93–109, 2000.

[5] C. Godsil, G. Royle, Algebraic Graph Theory. New York. *Springer Verlag*, 2001

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

[7] M. Katchalski, A. Meir, On empty triangles determined by points in the plane. *Acta Mathematica Hungarica.* 51, pp.323–328, 1988.

[8] J. Matoušek, Lectures on discrete geometry. New York. *Springer Verlag*, 2002.

[9] P. Valtr , On the minimum number of empty polygons in planar point sets. *Studia Sci. Math. Hungar.* 30, pp. 155–163, 1995.

Polygonal Reconstruction from Approximate Offsets*

Eric Berberich† Dan Halperin‡ Michael Kerber§ Roza Pogalnikova‡

Abstract

Given a polygonal shape Q with n vertices, can it be expressed, up to a tolerance in Hausdorff distance, as the Minkowski sum of another polygonal shape with a disk of fixed radius? If it does, we also seek a preferably simple solution shape P; P's offset constitutes an accurate, vertex-reduced, and smoothened approximation of Q. We give a decision algorithm for fixed radius in $O(n \log n)$ time that handles any polygonal shape. For convex shapes, the complexity drops to $O(n)$, which is also the time required to compute a solution shape P with at most one more vertex than a vertex-minimal one.

1 Introduction

Computing the *offset* of a polygon, namely points at most some fixed distance r away from the polygon, is a fundamental geometric operation recurring in a variety of applications. A standard way to obtain it is via the *Minkowski sum* of the polygon and a disk of radius r, which results in a shape bounded by straight-line segments and circular arcs. Modeling the disk in the Minkowski sum with a (tight) polygon yields an approximate piecewise-linear offset. Often, such an approximation is the legacy data which a program has to deal with – the original shape before offsetting is unknown.

While offset computation and smoothening of shapes have been extensively studied, we address the *(offset-)reconstruction problem*, that seems not to have been addressed in the literature: Given a polygonal shape Q, is it the approximate offset of another polygonal shape? And if so, is there a good such P (say, one with a small number of vertices)? As offsetting blurs small features, a definite reconstruction of the original shape from Q (or even of its topology) is impossible in general. However, a good choice of P could lead to a more compact and smooth representation of the shape given by Q.

*Work by E.B. and D.H. has been supported in part by the German-Israeli Foundation (grant no. 969/07). Work by D.H. has also been supported in part by the Israel Science Foundation (grant no. 236/06) and the Hermann Minkowski–Minerva Center for Geometry at Tel Aviv University.

†Max-Planck-Institut für Informatik, Saarbrücken, Germany, eric@mpi-inf.mpg.de

‡Tel Aviv University, Tel Aviv, Israel, {danha|rozapoga}@post.tau.ac.il

§IST Austria, Klosterneuburg, Austria, mkerber@ist.ac.at

In Section 2, we present an algorithm that decides for any given polygonal shape Q with n vertices (possibly unbounded), and two real parameters $r, \ > 0$, whether Q is within Hausdorff-distance to the r-offset of some other (yet unknown) polygonal shape P; if the answer is yes, we also return one such P. It gives the exact answer after $O(n \log n)$ operations in the real-RAM model by constructing offsets with increasing radii three times, exploiting this increase in a particular fashion. For convex Q we reduce the running time to optimal $O(n)$ in Section 3 and also compute a P as above which even minimizes (up to one extra vertex) the number of vertices among all valid choices. Furthermore, P's r-offset constitutes a tangent-continuous arc spline approximation of Q where all circular arcs have the same radius. This abstract summarizes [2] in which we give more details and full proofs.

Related work LEDA and CGAL contain code to compute Minkowski sums of polygons. The latter implementation also computes the exact or approximate offset of a polygon [5].

Smoothening polygonal shapes is desirable for NC machining. Such aims at tangent-continuous *arc-splines* consisting of segments and circular arcs which enable a uniform and fast processing and often alleviate the problem of overheating of the machine or the material. For purely polygonal input one can distinguish results using single arcs or biarcs (besides segments). Drysdale et al. [3] compute a vertex-minimal solution not adding new vertices, while Held et al. [4] compute approximations with arbitrary vertex placements and their tolerance band might even be asymmetric. Our reconstruction approach constrains the solution by allowing a single radius only. It disables tangent-continuity in general. But this can also be seen as a relaxation: We consider our reconstruction approach as an interesting alternative to existing approaches because on success, it yields an approximation that reflects the construction history of Q.

We also seek a vertex-minimal P whose offset is close to Q. P is actually constrained by a set of shapes. A related problem is to find a *minimal-link polygon* that is nested between two others; see [1] from which our approach adapts some ideas.

2 The Decision for Polygonally Bounded Sets

For a set $X \subset \mathbb{R}^2$ we denote its boundary by ∂X and its complement by $X^C := \mathbb{R}^2 \setminus X$. For a point p and

a closed X, letting $d(\cdot,\cdot)$ be the Euclidean distance function, we write $d(p,X) := \min\{d(p,x) \mid x \in X\}$. A *polygonal region* $X \subset \mathbb{R}^2$ has a piecewise-linear (finite number of lines) boundary. The points where these straight-line segments intersect are the *vertices* of the polygonal region. If X is bounded, ∂X is a set of (weakly) simple polygons. For two sets X and Y, we denote their Minkowski sum by $X \oplus Y := \{x+y \mid x \in X, y \in Y\}$. For any $c \in \mathbb{R}^2, v \in \mathbb{R}$, we write $D_v(c) := \{p \in \mathbb{R}^2 \mid d(c,p) \leq v\}$ for the (closed) v-disk around c, and $D_v := D_v(O)$ for the disk centered at the origin. The r-offset of a set X, offset(X,r), is the Minkowski sum $X \oplus D_r$.

The (symmetric) Hausdorff distance of two closed point sets X and Y is $H(X,Y) := \max\{\max\{d(x,Y) \mid x \in X\}, \max\{d(y,X) \mid y \in Y\}\}$. *We say that X is -close to Y (and Y to X) if $H(X,Y) \leq$, which can also be expressed alternatively:*

Proposition 1 *For X,Y closed, X is -close to Y if and only if $Y \subseteq$ offset$(X,$ $)$ and $X \subseteq$ offset$(Y,$ $)$.*

Decision algorithm From now, we fix $r > 0$, $>$ 0, and a polygonal region Q, and consider the following question: Can we find a polygonal region P such that Q and the r-offset of P have Hausdorff-distance at most ? First of all, we can assume that $r >$; otherwise, we can choose $P := Q$, because offset(Q,r) and Q have Hausdorff-distance at most .

Definition 1 *For $r > 0$, and $X \subset \mathbb{R}^2$, the r-inset of X is the set* inset$(X,r) :=$ offset$(X^C,r)^C = \left\{x \in \mathbb{R}^2 \mid D_r(x) \subseteq X\right\}$.

Algorithm 1 Is there any closed polygonal region P such that a given Q is -close to offset(P,r)?

(1) $Q \leftarrow$ offset$(Q,$ $)$
(2) $\Pi \leftarrow$ inset$(Q$ $,r)$
(3) $\tilde{Q} \leftarrow$ offset$(\Pi, r+$ $)$
(4) return $Q \subseteq \tilde{Q}$

We next prove that Algorithm 1 correctly decides whether Q is -close to some r-offset of a polygonal region. A first observation is that for any polygonal region P, offset$(P,r) \subseteq Q$ if and only if $P \subseteq \Pi$. This is an immediate consequence of the definition of insets. This shows that for any offset(P,r) that is -close to Q, P must be inside Π. Moreover, it shows that any choice of $P \subseteq \Pi$ already satisfies one of Proposition 1's inclusions. It is only left to check whether $Q \subseteq$ offset(offset$(P,r),$ $) =$ offset$(P, r+$ $)$.

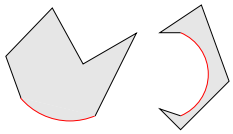
Lemma 2 *Q is -close to* offset(P,r) *if and only if $P \subseteq \Pi$ and $Q \subseteq$* offset$(P, r+$ $)$.

To prove correctness of the algorithm, we have to show that $Q \subseteq$ offset$(\Pi, r+$ $)$ already implies that there also exists a polygonal region $P \subseteq \Pi$ with $Q \subseteq$ offset$(P, r+$ $)$. Indeed, Π is not polygonal in general; we have to study its shape closer to prove that we can approximate it by a polygonal region, maintaining the property that the offset remains -close to Q.

The shape of offsets and insets For a polygonal region Q, it is not hard to figure out the shape of $Q =$ offset$(Q,$ $)$: It is a 2-manifold with boundary that is bounded by straight-line segments and by circular arcs, belonging to a circle of radius . It is important to remark that all circular arcs are *convex*:

Definition 2 *Let $X \subset \mathbb{R}^2$ be a 2-manifold with boundary with some circular arc bounding it. Then, is called* concave *with respect to X, if each segment connecting two distinct points on is not fully contained in X. Otherwise, the arc is called* convex.

We call X a convexly *(resp.* concavely*)* bounded region with radius r, *if ∂X consists of finitely many straight-line segments and convex (resp. concave) circular arcs that are all of radius r, interlinked at the* vertices *of the region.*

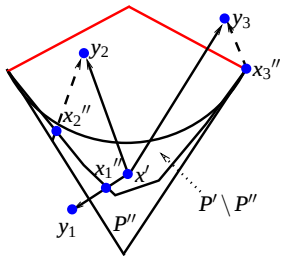


Note that a convexly bounded region (left) is not necessarily convex. The r-offset of a polygonal region P is a convexly bounded region with radius r. The heart of this section is a proof that the same also holds if P is concavely bounded (right) with radius smaller than r:

Theorem 3 *Let P be a concavely bounded region with radius r_1, and let $r_2 > r_1$. Then, there is a polygonal region $P_L \subseteq P$ such that* offset$(P,r_2) =$ offset(P_L,r_2). *In particular,* offset(P,r_2) *is a convexly bounded region with radius r_2.*

Note that the correctness of Algorithm 1 already follows by noticing that Q is a convexly bounded region with radius , and we can apply Theorem 3 to all constructed offsets, since $< r < r+$:

Corollary 4 *Algorithm 1 returns true if and only if there exists a polygonal region P such that* offset(P,r) *is -close to Q.*



We now give a sketch of the proof of Theorem 3. W.l.o.g., we assume that each concave circular arc spans less than half a circle. The arc's *linear cap* is the region enclosed by and the two lines tangent to the circle through the endpoints of . The *extended linear cap* is the region spanned by the two tangents just mentioned and the two normals at the endpoints.

We iteratively replace an arc of a concavely bounded region P' (starting with P) by a polyline ending in the endpoints of , such that the polyline does neither leave P' nor 's linear cap, and such that other boundary parts of P' are not intersected. This yields a concavely bounded region P'' with one arc less.

We show that in each iteration, the r_2-offsets of P' and P'' are the same. For that we consider any point $x' \in P' \setminus P''$, in the region that is cut off by P'', and consider $y = x' + v'$ for an arbitrary $v' \in D_{r_2}$. We show that in each case, y can also be written by $y = x'' + v''$, with $x'' \in P''$, and $v'' \in D_{r_2}$.

The proof then proceeds by studying several cases based on the location of the point y with respect to the extended linear cap of ; see y_1, y_2, y_3 in the previous figure and [2] for full details of the proof.

Theorem 5 *Let P be concavely bounded with radius r_1 having n vertices, and assume $r_2 > r_1$. Then,* offset(P, r_2) *has $O(n)$ vertices and it can be computed in $O(n \log n)$ time.*

Proof. By Theorem 3, it suffices to consider a polygonally bounded P_L instead of P; a trapezoidal decomposition leads to a P_L with $O(n)$ vertices. The Voronoi diagram of P_L's vertices and (open) edges can be computed in $O(n \log n)$ time and has size $O(n)$ [6]. The r_2-offset boundary inside a Voronoi cell is formed by the intersection of the cell with a parallel line (for the cell of an edge of P_L) or a circle (for the cell of a vertex). Because the offset boundary intersects any Voronoi edge only a constant number of times, the number of vertices (and edges) of the offset is proportional to the number of Voronoi edges. The offset is constructed by sweeping the collection of all the boundary curves from all Voronoi cells, which runs in $O(n \log n)$ because of the absence of interior intersections. □

The running time of Algorithm 1 follows by applying Theorem 5 for the first three steps. The fourth step is easily seen to run in $O(n \log n)$ time as well.

3 Convex Polygons

Lemma 6 *If Q is a convex polygonal region, then Π, as computed by Algorithm 1, is also a convex polygonal region, and it can be computed in $O(n)$ time.*

Proof. Q is the intersection of the halfplanes bounded by lines that support the polygon edges. Observe that Π can be constructed by shifting each such line by $r -$ inside the polygon, which shows that Π is convex. For the time complexity, we compute the lower (upper) envelope for the lines supporting upper (lower) edges of Q by dualizing the lines supporting the edges to points and computing their upper (lower) hull by Graham's scan. We exploit the fact that we already know the x-order of these points. □

The next step of Algorithm 1 would be to check $Q \subseteq$ offset$(\Pi, r +$ $)$. Let $q_1, \ldots, q_n$ be the vertices of Q (in counterclockwise order) and define $K_i = D_{r+}(q_i)$. The following lemma together with Lemma 6 implies that Algorithm 2 runs in linear time.

Algorithm 2 Is there any closed polygonal region P such that a given *convex* Q is -close to offset(P, r)?

(1) $Q \leftarrow$ offset$(Q,$ $)$
(2) $\Pi \leftarrow$ inset$(Q$ $, r)$
(3) return $\bigwedge_{i=1}^{n} (K_i \cap \Pi \neq \emptyset)$

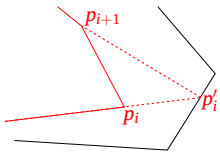
Lemma 7 *Q is -close to* offset(Π, r) *if and only if Π intersects each of the K_i.*

Reducing the number of vertices We assume in the remainder of this section that offset(Π, r) is -close to Q. Since our goal is to find a possibly simple approximation of Q, we look for a $P \subseteq \Pi$ whose offset is -close to Q, but with fewer vertices than Π. Any such P intersects each of the convex (convexly bounded) regions $_i := K_i \cap \Pi, i = 1, \ldots, n$, of radius $r +$, which we call *eyelets* from now on. The converse is also true: Any *convex* polygonal manifold $P \subseteq \Pi$ that intersects all eyelets $_1, \ldots, _n$ has an r-offset that is -close to Q.

Proposition 8 *If* offset(P, r) *is -close to Q, and $P \subseteq P' \subseteq \Pi$, then* offset$(P', r)$ *is -close to Q.*

We call a polygonal region P *(vertex-)minimal*, if its r-offset is -close to Q, and there exists no other such region with fewer vertices. Necessarily, a minimal P must be convex – otherwise, its convex hull CH(P) has fewer vertices and it can be seen by Proposition 8 that offset$(\mathrm{CH}(P), r)$ is also -close to Q.

Lemma 9 *There exists a minimal polygonal region $P \subseteq \Pi$ whose vertices are all on $\partial\Pi$.*

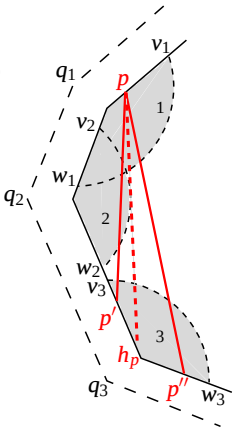


Proof. We pull each vertex $p_i \notin \partial\Pi$ in direction of the ray emanating from p_{i-1} towards p_i until it intersects $\partial\Pi$ in the point p_i' (dragging p_i's incident edges along with it); see the enclosed illustration. For $P' = (p_1, \ldots, p_{i-1}, p_i', p_{i+1}, \ldots, p_m)$: $P \subseteq P' \subseteq \Pi$, offset$(P', r)$ is -close to Q by Proposition 8. □

Thus, we can restrict our search to polygons with vertices on $\partial\Pi$. We call a polygonal region P *good*, if $P \subseteq \Pi$, all vertices of P lie on $\partial\Pi$, and P intersects each eyelet $_1, \ldots, _n$.

Definition 3 *For two points $u, u' \in \partial\Pi$, we denote by $[u, u'] \subset \partial\Pi$ all points that are met when travelling along $\partial\Pi$ from u to u' in counterclockwise order. Likewise, we define half-open and open intervals $[u, u')$, $(u, u']$, (u, u').*

Let $_i = K_i \cap \Pi$ be q_i's eyelet as before. We consider $_i \cap \partial\Pi$. The portion of that intersection set that is visible from q_i (considering Π as an obstacle) defines an interval $[v_i, w_i] \subset \partial\Pi$. We call v_i the *spot* of the eyelet $_i$. Finally, for $u, u' \in \partial\Pi$, we say that the segment $\overline{uu'}$ is *good*, if for all spots $v_i \in (u, u')$, $\overline{uu'}$ intersects the corresponding eyelet $_i$. The figure on the right illustrates these definitions: The segment $\overline{pp'}$ is good, whereas $\overline{pp''}$ is not good, because $v_2 \in (p, p'')$, but it does not intersect $_2$.



Theorem 10 *Let P be a convex polygonal region with all its vertices on $\partial\Pi$. Then, P is good if and only if all its bounding edges are good.*

Proof. Any spot v_i of an eyelet $_i$ either corresponds to some vertex p_ℓ of P, or lies inside some interval $(p_\ell, p_{\ell+1})$. Since $\overline{p_\ell p_{\ell+1}}$ is good, it intersects $_i$. For the converse, assume that $\overline{p_\ell p_{\ell+1}}$ is not good, which encloses with the interval $(p_\ell, p_{\ell+1})$ a polygonal region $R \subseteq \Pi \setminus P$. Hence, there is a spot $v_i \in R$ such that $\overline{p_\ell p_{\ell+1}}$ does not intersect the eyelet $_i$. It follows that the entire $_i$ is inside R (see the above illustration, considering $\overline{pp''}$ and $_2$). Thus, $P \cap _i = \emptyset$, and so P cannot be good. □

For $u \in \partial\Pi$, we define its *horizon* $h_u \in \partial\Pi$ as the maximal point (when travelling from u in counterclockwise order on $\partial\Pi$) such that the segment $\overline{uh_u}$ is good. An example is depicted in the previous figure: The segment $\overline{uh_u}$ is tangential to $_2$, so if going any further than h_u from u, the segment would miss $_2$ and thus become non-good.

Lemma 11 *Let P be a good polygonal region, and $u \in \partial\Pi$. Then, P has a vertex $p \in (u, h_u]$.*

Proof. Assume that P has no such vertex, and let $p_1, \ldots, p_\ell$ be its vertices on $\partial\Pi$. Let p_j be the vertex of P such that $u \in (p_j, p_{j+1})$. Then, also $h_u \in (p_j, p_{j+1})$, because otherwise, $p_{j+1} \in (u, h_u]$. Since P is good, the segment $\overline{p_j p_{j+1}}$ is good, too. It is not hard to see that, consequently, both $\overline{p_j u}$ and $\overline{u p_{j+1}}$ are good. However, the latter contradicts the maximality of the horizon h_u. □

For an arbitrary initial vertex $s \in \partial\Pi$, we finally specify a polygonal region P^s by iteratively defining its vertices. Set $p_1 := s$. For any $j \geq 1$, if the segment $\overline{p_j s}$, which would close P^s, is good, stop. Otherwise, set $p_{j+1} := h_{p_j}$. Informally, we always jump to the next horizon until we can reach s again without missing any of the eyelets. By construction, all segments of P^s are good, so P^s itself is good.

Theorem 12 *Let P be a minimal polygonal region for Q, having* OPT *vertices. Then, for any $s \in \partial\Pi$, P^s has at most* OPT $+ 1$ *vertices*

Proof. We first prove that P^s has the minimal number of vertices among all good polygonal regions that have s as a vertex. Let $s := p_1, \ldots, p_m$ be the vertices of P^s. There are $m-1$ segments of the form $\overline{p_\ell h_{p_\ell}}$. By Lemma 11, any good polygonal region has a vertex inside each of the intervals $(p_\ell, h_{p_\ell}]$. Together with the vertex at s, this yields at least m vertices, thus P^s is indeed minimal among these polygonal regions.

Next, consider any minimal polygonal region $P^\star$. We can assume that all its vertices are on $\partial\Pi$ by Lemma 9. If s is not a vertex of $P^\star$, we add it to the vertex set and obtain a polygonal region P' with at most OPT $+ 1$ vertices that has s as a vertex. P^s has at most as many vertices as P', so $m \leq$ OPT $+ 1$. □

As each visit of an eyelet requires constant time, the construction of a horizon is proportional to the number of visited eyelets, and there are only linearly many eyelets, we can state:

Theorem 13 *For an arbitrary initial vertex s, P^s can be computed in $O(n)$ time.*

Additional Material In the extended version of this paper [2] we also present a new approximation of a polygonal shape's r-offset by line segments and circular arcs aiming at an accurate and compact description. Its vertices are rational and the Hausdorff distance to the exact offset is at most some prescribed . In addition we discuss there some immediate extensions of the algorithms presented here.

References

[1] A. Aggarwal, H. Booth, J. O'Rourke, S. Suri, and C.-K. Yap. Finding minimal convex nested polygons. *Inf. Comput.*, 83(1):98–110, 1989.

[2] E. Berberich, D. Halperin, M. Kerber, and R. Pogalnikova. Polygonal reconstruction from approximate offsets. Manuscript, 2009, available upon request.

[3] R. Drysdale, G. Rote, and A. Sturm. Approximation of an open polygonal curve with a minimum number of circular arcs and biarcs. *Computational Geometry*, 41(1-2):31 – 47, 2008.

[4] M. Heimlich and M. Held. Biarc approximation, simplification and smoothing of polygonal curves by means of Voronoi-based tolerance bands. *Int. J. Comput. Geometry Appl.*, 18(3):221–250, 2008.

[5] R. Wein. Exact and approximate construction of offset polygons. *Computer-Aided Design*, 39(6):518–527, 2007.

[6] C.-K. Yap. An $O(n \log n)$ algorithm for the Voronoi diagram of a set of simple curve segments. *Discrete & Computational Geometry*, 2:365–393, 1987.

The Class Cover Problem with Boxes

S. Bereg* S. Cabello † J.M. Díaz-Báñez‡ P. Pérez-Lantero § C. Seara ¶ I. Ventura‡

Abstract

In this paper we study the following problem: Given sets R and B of red and blue points respectively on the plane, find a minimum-cardinality set $\mathcal{H}$ of axis-aligned open rectangles such that every point in B is covered by at least one rectangle of $\mathcal{H}$, and no rectangle of $\mathcal{H}$ contains a point of R. We prove the NP-hardness of the original version of the stated problem, and give exact or approximated algorithms depending on the type of rectangles considered.

1 Introduction

Let R and B be sets of red and blue points respectively on the plane. Denote $S = R \cup B$, $r = |R|$, and $b = |B|$. The x- and y-coordinates of the point p are denoted by $\mathrm{x}(p)$ and $\mathrm{y}(p)$ respectively. We say that a set X is R-empty if X contains no red points. Given S, a classical problem in data mining and classification problems is the *Class Cover problem* [3]. It consists in finding a minimum-cardinality set of R-empty disks such that every point in B is contained in at least one of the disks. In [3], the authors considered the case in which the disks are centered at some point in B. They showed that this version is NP-hard. In this paper we consider another variation by considering axis-aligned rectangles (i.e. boxes) as the covering objects. It can be defined as follows:

The Boxes Class Cover problem (BCC-problem): *Given $S = R \cup B$, find a minimum-cardinality set $\mathcal{H}$ of R-empty axis-aligned open rectangles such that every point in B is contained in at least one rectangle of $\mathcal{H}$.*

Let $\mathcal{H}$ be a solution to the BCC-problem. We can extend each box $H \in \mathcal{H}$ until each side of H passes through a red point or reaches infinity. Thus we will only consider the set $\mathcal{H}^*$ of all the R-empty open boxes whose sides pass through red points or are in infinity. Up to symmetry, such types of boxes are depicted in Fig. 1.

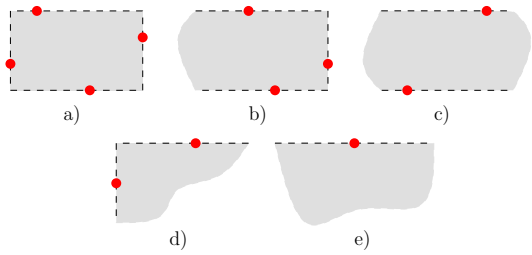


Figure 1: Types of boxes in $\mathcal{H}^*$. a) A rectangle, b) a half-strip, c) a strip, d) a quadrant, e) a half-plane.

The outline of this paper is as follows. In Section 2 we prove that the BCC-problem is NP-hard. In Section 3 we mention related results concerning range spaces and epsilon-nets that give approximation algorithms. In Section 4 we study the BCC-problem when we restrict the boxes to be strips or half-strips. Finally, in Section 5, we consider the version of the BCC-problem in which the boxes are axis-aligned squares, and prove its NP-hardness.

2 Hardness

We prove here that the BCC-problem is NP-hard based on a reduction from the *Rectilinear Polygon Covering problem* (RPC-problem), that is defined as follows: Given a rectilinear polygon P, find a minimum cardinality set of axis-aligned rectangles whose union is exactly P. The RPC-problem is NP-hard [5].

Theorem 1 *The BCC-problem is NP-hard.*

Proof. Suppose we are given a rectilinear polygon P as an instance of the RPC-problem. Let A_1 be the set of all distinct axis-parallel lines that pass through an edge of P. For every two consecutive vertical (resp. horizontal) lines in A_1, draw a vertical (resp. horizontal) line in between. Denote as A_2 these additional lines. Let G be the grid defined by $A_1 \cup A_2$. We put a red (resp. blue) point in every vertex of $G \setminus P$ (resp. $G \cap P$) (see Figure 2). Let S be the

*Department of Computer Science, University of Texas at Dallas, USA, besp@utdallas.edu. Partially supported by project MEC MTM2009-08652.

†Department of Mathematics, FMF, University of Ljubljana, SLOVENIA, sergio.cabello@fmf.uni-lj.si

‡Departamento Matemática Aplicada II, Universidad de Sevilla, SPAIN, {dbanez,iventura}@us.es. Partially supported by project MEC MTM2009-08652.

§Departamento de Computación, Universidad de La Habana, CUBA, pablo@matcom.uh.cu. Partially supported by projects MAEC-AECID and MEC MTM2009-08652.

¶Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, SPAIN, carlos.seara@upc.edu. Partially supported by projects MEC MTM2009-07242 and Gen. Cat. DGR2009GR1040.

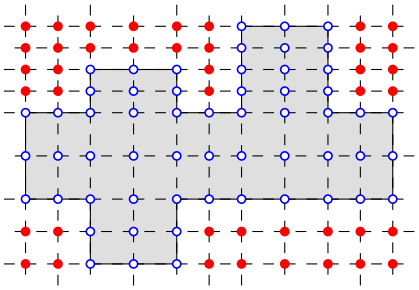


Figure 2: The reduction from an instance of the RPC-problem to an instance of the BCC-problem. Red points are solid dots, and blue points are circles.

above set of red and blue points. Clearly, any covering set of the polygon P corresponds with a solution to the BCC-problem on S with the same cardinality. Conversely, any solution $\mathcal{H}$ for the BCC-problem on S can be adjusted to be a covering set of P. Namely, consider that each box in $\mathcal{H}$ is maximal (i.e. it cannot be enlarged for covering more blue points) and let $\mathcal{H}' = \{\mathrm{BB}(H) \mid H \in \mathcal{H}\}$, where $\mathrm{BB}(H)$ is the axis-parallel bounding box of $H \cap B$. Every box in $\mathcal{H}'$ is fully contained in P, and the union of the boxes in $\mathcal{H}'$ covers P. □

3 Related results

In this section we relate our problem with ε-nets. A finite[1] range space $(X, \mathcal{R})$ is a pair consisting of an underlying finite set X of objects and a finite collection $\mathcal{R}$ of subsets of X called ranges. Given a range space $(X, \mathcal{R})$ the SET COVER problem [6] asks for the minimum-cardinality subset of $\mathcal{R}$ that covers X. Then the BCC-problem is an instance of the SET COVER problem in the range space $(B, \mathcal{H}^*)$. The dual of the SET COVER problem is the HITTING SET problem [6]. Given the range space $(X, \mathcal{R})$, its dual range space is $(\mathcal{R}, X^*)$ where $X^* = \{\mathcal{R}_x \mid x \in X\}$ and $\mathcal{R}_x$ is the set of all ranges in $\mathcal{R}$ that contains x [2]. A set cover in the primal range space is a hitting set in its dual, and vice versa. In [2], a general approach for finding an approximated hitting set for range spaces is given, and it is based on finding small size subsets called ε-nets, as candidate hitting sets, and works for range spaces with finite VC-dimension [2, 7, 9]. In terms of our problem, an ε-net is a subset $B' \subset B$ such that any box in $\mathcal{H}^*$ that contains $\varepsilon|B|$ points covers an element of B'. In the dual space an ε-net is a subset $H \subset \mathcal{H}^*$ that covers all points p of B such that p is covered by at least $\varepsilon|\mathcal{H}^*|$ boxes of $\mathcal{H}^*$. Note that the VC-dimension of our range space $(B, \mathcal{H}^*)$ is at most four. For range spaces with constant VC-dimension, the method in [2] reports a hitting set of size at most a factor of $O(\log c)$ from the optimal size c. Then, since our range space $(B, \mathcal{H}^*)$ has constant VC-dimension (and thus the dual range space does), the technique in [2] can be applied to obtain an $O(\log c)$-approximation algorithm for the BCC-problem, where c is the size of an optimal covering.

[1] A range space can be infinite, but for the purpose of our problem we only define it as finite.

4 Solving particular cases

Here we study some special cases. Namely, we consider only certain boxes of $\mathcal{H}^*$ having at most three points on their boundary. In Subsection 4.1 we use both horizontal and vertical strips for covering. In Subsection 4.2 we consider only one type of half-strips, say top-bottom half-strips, as covering rectangles. Finally, in Subsection 4.3 we prove that the BCC-problem remains NP-hard if we cover with half-strips in the four possible directions, and that there exists a constant factor approximation algorithm.

4.1 Covering with horizontal and vertical strips

A box in $\mathcal{H}^*$ is a strip if it does not contain red points in two consecutive sides (see Fig. 1 c) and e)). Thus we consider as covering objects vertical or horizontal strips, and axis-aligned half-planes. Notice that a covering set exists if and only if every blue point can be covered by an axis-parallel line avoiding red points. Also, if a blue point and a red point lie on the same vertical (resp. horizontal) line then the blue point can only be covered by a horizontal (resp. vertical) strip. First we sort S by x-coordinate and by y-coordinate (two orders) in $O(r \log r + b \log b)$ time. Preprocess in linear time the x-order to assign for each blue point p the references $r_{\mathrm{x}}^-(p)$ and $r_{\mathrm{x}}^+(p)$ to its previous and next red points respectively. Do the same with the y-order to assign $r_{\mathrm{y}}^-(p)$ and $r_{\mathrm{y}}^+(p)$. A solution does not exist if and only if there is a blue point p such that $\mathrm{x}(p) = \mathrm{x}(r_{\mathrm{x}}^-(p))$ or $\mathrm{x}(p) = \mathrm{x}(r_{\mathrm{x}}^+(p))$, and $\mathrm{y}(p) = \mathrm{y}(r_{\mathrm{y}}^-(p))$ or $\mathrm{y}(p) = \mathrm{y}(r_{\mathrm{y}}^+(p))$. It can be checked in $O(r + b)$ time. Thus suppose now that a solution exists. We can assume that there are no red and blue points on the same horizontal or vertical line. Otherwise we can apply the following linear-time preprocessing and after that, solve the same problem for the blue points not yet covered. Given a blue point p let $H_v(p)$ be the vertical strip bounded by $r_{\mathrm{x}}^-(p)$ and $r_{\mathrm{x}}^+(p)$, and $H_h(p)$ be the horizontal strip bounded by $r_{\mathrm{y}}^-(p)$ and $r_{\mathrm{y}}^+(p)$. From each non-covered blue point p, if $\mathrm{x}(p) = \mathrm{x}(r_{\mathrm{x}}^-(p))$ or $\mathrm{x}(p) = \mathrm{x}(r_{\mathrm{x}}^+(p))$ then include in the solution the strip $H_h(p)$. Else, if $\mathrm{y}(p) = \mathrm{y}(r_{\mathrm{y}}^-(p))$ or $\mathrm{y}(p) = \mathrm{y}(r_{\mathrm{y}}^+(p))$ then include $H_v(p)$. Consider a graph G whose set of vertices V is the set of strips, and whose set of edges E is as follows. For each blue point p put an edge between the

horizontal strip $H_h(p)$ and the vertical strip $H_v(p)$; different blue points may define the same edge. The graph G is bipartite, has $O(r)$ vertices and $O(b)$ edges, and can be constructed in $O(r+b)$ time. Since each blue point is covered by exactly two strips the problem reduces to finding a *Minimum Vertex Cover* [6] in G. For bipartite graphs the *Vertex Cover Problem* is equivalent to the *Maximum Matching Problem* because of the König's theorem, and thus it can be solved in $O(\sqrt{|V|}|E|) = O(\sqrt{r}b)$ time [8].

Theorem 2 *The BCC-problem can be solved in $O(r\log r + b\log b + \sqrt{r}b)$ time if we only use axis-aligned strips as covering objects.*

4.2 Covering with half-strips in one direction

A box of $\mathcal{H}^*$ is a half-strip if it contains at most three points on its boundary (see Fig. 1 b), c), d), and e)), and is top-bottom if either contains a red point on its top side or is a vertical strip. We give an exact $O(r\log r + b\log b + b\log r)$-time algorithm based on a simple data structure

Consider the structure of rays that is obtained by drawing a bottom-top red ray starting at each red point. Every time we select the highest blue point p not yet covered, and let s_p be the maximum-length horizontal segment passing through p whose interior does not intersect any red ray. Let p_l (resp. p_r) be the red point such that the left (resp. right) endpoint of s_p is located in the ray corresponding to p_l (resp. p_r). We say that p_l and p_r are the left and the right red neighbors of p. We include in the solution the top-bottom half-strip H_p whose top side is s_p translated upwards until it touches a red point or reaches the infinite, i.e. the top-bottom half-strip in $\mathcal{H}^*$ covering p and the maximum number of other blue points. We finish when all blue points are covered.

The correctness of the algorithm follows from the fact that, if p is the blue point not yet covered that has maximum ordinate, then the strip H_p satisfies that for any other non-covered blue point p' not in H_p, p and p' cannot be covered with the same top-bottom half-strip. This is because any top-bottom half-strip covering p and p' contains at least one of the two red neighbors of p. In the algorithm we first preprocess S to obtain the decreasing y-order and to build two balanced binary search trees T_B and T_R containing respectively the blue and the red points sorted lexicographically. The first one allows deletion of elements and in the second one each node v is labeled with the element of minimum ordinate in the subtree rooted at v. This labeling permits us to obtain the red neighbors for a given blue point p and to determine the top side of H_p both in $O(\log r)$ time. The preprocessing time is $O(r\log r + b\log b)$ in total. Now for each blue point p in the decreasing y-order such that p is not still covered (i.e. p is in T_B) we do: find the left and the right red neighbors p_l and p_r of p, determine H_p, include H_p in the solution, and remove in $O(k_p \log b)$ time the k_p blue points in T_B covered by H_p.

Theorem 3 *The BCC-problem can be solved in $O(r\log r + b\log b + b\log r)$ time if we only use half-strips in one direction as covering objects.*

4.3 Covering with half-strips

Here we study the BCC-problem when the covering boxes are half-strips oriented in any of the four possible directions. We firstly show that this variant is also NP-hard, and afterwards we give a constant-factor approximation algorithm due to results in [2, 4]. We name this version as *The Half-Strip Class Cover problem* (HSCC-problem), and use a reduction from the 3-SAT-problem [6] to prove that it is NP-hard.

Theorem 4 *The HSCC-problem is NP-hard.*

Proof. Given an instance $\mathcal{F}$ of the 3-SAT-problem with t variables $x_1, \ldots, x_t$ and m clauses $C_1, \ldots, C_m$, we construct an instance of the HSCC-problem as follows. Let α be a set of t pairwise disjoint vertical strips of equal width such that the i-th strip from left to right α_i represents the variable x_i. Similarly, let β be a set of $t+m$ pairwise disjoint horizontal strips of equal width, such that the clause C_j is represented by the $(t+j)$-th strip β_{t+j} from bottom to up. Consecutive strips in α and β are well separated. Let δ_i be the dividing line of α_i. We say that the part of the interior of α_i that is to the right (resp. to the left) of δ_i is the true (resp. false) part of α_i. For each variable x_i $(1 \le i \le t)$ we put in $\alpha_i \cap \beta_i$ a set V_i of red and blue points as shown in Fig. 3 a). For each clause C_j $(1 \le j \le m)$ we add a set W_j of bicolored points in the following manner. Suppose that C_j involves the variables x_i, x_k, and x_l $(1 \le i < k < l \le m)$. Define the lines ℓ_1, ℓ_1', ℓ_2, ℓ_2', ℓ_3, and ℓ_3', and add a set of red points as depicted in Fig. 4. Put a blue point in the intersection of ℓ_1' and ℓ_3, and another one in the

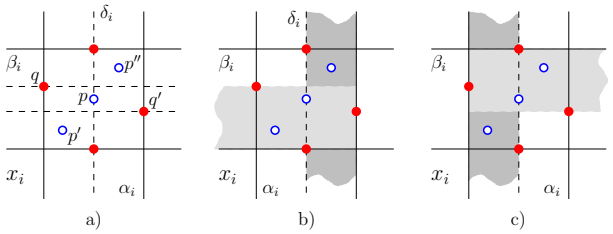


Figure 3: In a) the point set V_i for the variable x_i. In b) and c) the two ways of optimally covering the blue points in V_i. b) x_i is equal to true, c) x_i is equal to false.

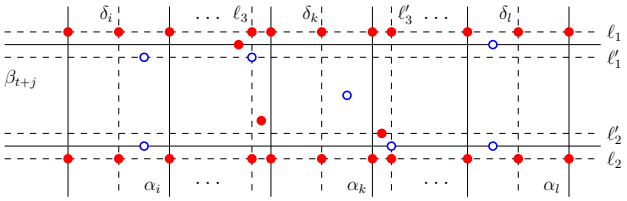


Figure 4: The set W_j of red and blue points for the clause $C_j = (x_i \vee x_k \vee \neg x_l)$.

intersection of ℓ'_3 and the bottom boundary of β_{t+j}. If x_i is not negated in C_j then put in the true part of α_i (otherwise in the false part) two blue points, the first one over ℓ'_1 and the second one over the bottom boundary of β_{t+j}. If x_k is not negated in C_j put one blue point in the center of the intersection of β_{t+j} and the true part of α_k (otherwise in the false part). Finally, if x_l is not negated in C_j put in the true part of α_l (otherwise in the false part) two more blue points, one over the top boundary of β_{t+j} and another one over the bottom boundary.

Let $S = \bigcup_{i=1}^{t} V_i \cup \bigcup_{j=1}^{m} W_j$ be the instance of the HSCC-problem. Note that we can cover optimally the blue points in V_i in two ways, the first one with a right-left half-strip covering the two lowest blue points in V_i and a vertical strip covering the true part of α_i (see Fig. 3 b)), and the second one with a vertical strip covering the false part of α_i and a left-right half-strip that covers the upper two blue points of V_i (see Fig. 3 c)). We say that the first way is a true covering of x_i (i.e. x_i is true), and that the second one is a false covering of x_i (i.e. x_i is false). For each clause C_j $(1 \leq j \leq m)$ that involves the variables x_i, x_k, and x_l $(1 \leq i < k < l \leq m)$ we observe that if at least one variable, say x_i, is such that the covering of V_i covers the blue points in $W_j \cap \alpha_i$ (i.e. the value of x_i makes C_j true), then at least two half-strips are needed to cover $W_j \setminus \alpha_i$. Otherwise, at least three half-strips are needed. We claim that $\mathcal{F}$ is satisfiable if and only if the blue points in S can be covered with $2t + 2m$ half-strips. □

Theorem 5 *There exists a polynomial-time $O(1)$-approximation algorithm for the HSCC-problem.*

Proof. Let $\mathcal{H}_S$ be the set of all half-strips in $\mathcal{H}^*$, and partition $\mathcal{H}_S$ into the subsets $\mathcal{H}_{S_v}$ and $\mathcal{H}_{S_h}$ of all vertical and horizontal half-strips, respectively. Note that $\mathcal{H}_{S_v}$ and $\mathcal{H}_{S_h}$ are families of pseudo-disks. In [4], the authors showed that if the geometric range space $(X, \mathcal{R})$ is such that $\mathcal{R}$ is a family of pseudo-disks, then ε-nets of size $O(\frac{1}{\varepsilon})$ exist for the dual range space, and that a cover of size with a factor $O(1)$ from the optimal can be found in polynomial time. Therefore, given $\varepsilon > 0$, the dual of the range space $(B, \mathcal{H}_{S_v})$ has (by using [4]) an $(\frac{\varepsilon}{2})$-net N_v of size $O(\frac{2}{\varepsilon})$. Analogously, the dual of the range space $(B, \mathcal{H}_{S_h})$ has an $(\frac{\varepsilon}{2})$-net N_h of size $O(\frac{2}{\varepsilon})$. We claim that $N_v \cup N_h$ is an ε-net of size $O(\frac{4}{\varepsilon}) = O(\frac{1}{\varepsilon})$ for the dual of $(B, \mathcal{H}_S)$. In fact, if p is a blue point covered by $\varepsilon|\mathcal{H}_S|$ half-strips, then at least $\frac{\varepsilon}{2}|\mathcal{H}_S|$ of them are either vertical or horizontal. Thus p is covered by a half-strip in $N_v \cup N_h$ since N_v and N_h are $(\frac{\varepsilon}{2})$-nets. Hence, there exists, by [2] and also by [4, Theorem 3.2], a polynomial-time $O(1)$-approximation algorithm. □

5 The Class Cover problem with Squares

In this section we consider the variant of the BCC-problem in which we use axis-aligned squares instead of general rectangles as covering objects. In [1], the following problem is studied: Given an image represented by an array of $\sqrt{n} \times \sqrt{n}$ black-and-white pixels, cover the black pixels with a minimum set of (possibly overlapping) squares. They proved that obtaining a solution for a polygonal binary image with holes is NP-hard. By a reduction from it we can prove:

Theorem 6 *The BCC-problem remains NP-hard if we use squares as covering objects.*

Note that a set of axis-aligned squares is a family of pseudo-disks, thus we can give an $O(1)$-approximation algorithm by using [2, 4].

References

[1] L.J. Aupperle, H.E. Corm, J.M. Keil, and J. O'Rourke. Covering Orthogonal Polygons with Squares. *In Proc. 26th Annu. Allerton Conf. on Comm., Cont. and Comp.*, pp. 97–106, 1988.

[2] H. Brönnimann and M.T. Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete and Computational Geometry*, Vol. 14, pp. 463–479, 1995.

[3] A.H. Cannon and L.J. Cowen. Approximation algorithms for the class cover problem. *Annals of Math. and Art. Int.*, Vol. 40, pp. 215–223, 2004.

[4] K.L. Clarkson and K. Varadarajan. Improved Approximation Algorithms for Geometric Set Cover. *Disc. and Comp. Geom.*, Vol. 37, pp. 43–58, 2007.

[5] J. C. Culberson and R. A.Reckhow. Covering polygons is hard. *J. Algorithms*, Vol. 17, pp. 2—44, 1994.

[6] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., NY, 1979.

[7] D. Haussler and E. Welzl. ε-nets and simplex range queries. *Disc. C. G.*, Vol. 2, pp. 127–151, 1987.

[8] J.E. Hopcroft and R.M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SICOMP*, Vol. 2, pp. 225—231, 1973.

[9] V. N. Vapnik and A. Ya. Červonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory Probab. Appl.*, Vol. 16, pp. 264–280, 1971.

How Alexander the Great Brought the Greeks Together While Inflicting Minimal Damage to the Barbarians

Mark de Berg* Dirk H.P. Gerrits* Amirali Khosravi* Ignaz Rutter†
Constantinos P. Tsirogiannis* Alexander Wolff‡

Abstract

Let $\mathcal{R}$ be a finite set of red point sites in $\mathbb{R}^d$ and let $\mathcal{B}$ be a set of n blue point sites in $\mathbb{R}^d$. We want to establish "safe" connections between the red sites by deleting a minimum number of blue sites such that the region controlled by the red sites is connected. More precisely, we want to find a minimum-size subset $\mathcal{B}_{\text{del}} \subseteq \mathcal{B}$ such that the red cells in the Voronoi diagram of $\mathcal{R} \cup \mathcal{B} \setminus \mathcal{B}_{\text{del}}$ form a connected region. For $|\mathcal{R}| = 2$ we present an optimal $O(n \log n)$-time algorithm for $d = 2$, and an $O(n^{d-1})$-time algorithm for $d \geqslant 3$; we also show that the problem is 3SUM-hard for $d = 3$. Furthermore, we show that the general problem, where the number of red sites is not a constant, is NP-hard.

1 Introduction

Let $\mathcal{R}$ be a set of red cities and let $\mathcal{B}$ be a set of blue cities. Suppose each city controls a subset of the space, namely the set of all points for which it is the closest city. The red people would like to be able to travel safely between any two points in the red region, without having to cross through hostile (blue) territory. This may not always be possible, however, since the red region need not be connected. Then some blue cities will have to be eliminated, in order to make the red region connected. As the red people are a friendly people, they wish to do so by eliminating as few blue cities as possible.

In a more abstract setting, the problem above can be formulated using Voronoi diagrams: we are given a set $\mathcal{R}$ of red point sites in $\mathbb{R}^d$ and a set $\mathcal{B}$ of blue point sites in $\mathbb{R}^d$, and we want to find a subset $\mathcal{B}_{\text{del}} \subseteq \mathcal{B}$ such that the red cells in the Voronoi diagram of $\mathcal{R} \cup \mathcal{B} \setminus \mathcal{B}_{\text{del}}$ form a connected region. We call every such subset a *connectivity set*, and we want to find a connectivity set of minimum size. We call the problem of computing such a set MIN VORONOI CONNECTIVITY. We obtain the following results:

- We solve the problem for two red sites, see Section 2. Our algorithm runs in $O(n \log n + n^{d-1})$ time, where n is the number of blue sites and $d \geq 2$ is the dimension of the underlying space. We show that this is optimal for $d = 2$ and that the problem is 3SUM-hard for $d = 3$.
- We show that the general problem, where the number of red sites is not a constant, is NP-hard, see Section 3.

Terminology and notation. We denote the Voronoi diagram of a set S of sites by $\text{Vor}(S)$. We say that two sites $p, q \in S$ are *neighbors* if the boundaries of their Voronoi cells have a $(d-1)$-dimensional overlap. In other words, two sites are neighbors if they share an edge in the Delaunay graph of S.

We denote the bisector of p and q by $\beta(p, q)$. For a third site r, we call the part of $\beta(p, q)$ that lies inside or on the boundary of the Voronoi cell of r in $\text{Vor}(\{p, q, r\})$—that is, the part of $\beta(p, q)$ lying at least as close to r as to p and q—the *shadow region* of r on $\beta(p, q)$. We say that r *covers* this part of $\beta(p, q)$.

2 The Case of Two Red Sites

In this section we consider the special case that $\mathcal{R}$ consists of only two sites.

Theorem 1 *Let $\mathcal{R}$ be a set of two points in $\mathbb{R}^d$ and $\mathcal{B}$ be a set of n points in $\mathbb{R}^d$. Then* MIN VORONOI CONNECTIVITY *can be solved in $O(n \log n)$ time for $d = 2$, and in $O(n^{d-1})$ time for $d \geqslant 3$.*

Proof. Let $\mathcal{R} = \{p, q\}$, and let $\beta := \beta(p, q)$ be the bisector of p and q. For each site $b \in \mathcal{B}$, the shadow region $\sigma(b)$ on β is a $(d-1)$-dimensional half-space within the $(d-1)$-dimensional space β. Observe that p and q are neighbors in $\text{Vor}(\mathcal{B} \cup \{p, q\})$ if and only if the union of the shadow regions $\sigma(b)$ over all $b \in \mathcal{B}$ does not fully cover β. Hence, we can solve the problem as follows.

1. Let $h(b)$ denote the $(d-2)$-flat bounding $\sigma(b)$, and let $H := \{h(b) : b \in \mathcal{B}\}$. Construct the

*Department of Computer Science, TU Eindhoven, the Netherlands. MdB and CPT were supported by the Netherlands' Organisation for Scientific Research (NWO) under project no. 639.023.301.

†Institute of Theoretical Informatics, Karlsruhe Institute of Technology (KIT), Germany

‡Institute of Computer Science, Universität Würzburg, Germany.

arrangement $\mathcal{A}(H)$ on β induced by the $(d-2)$-flats in H.

2. Compute for each cell of $\mathcal{A}(H)$ in how many shadow regions it is contained. Let C be a cell for which this number is minimum. Report the set $\mathcal{B}_{\text{del}}$ of blue sites whose shadow regions cover C.

For $d \geqslant 3$, we can construct the $(d-1)$-dimensional arrangement in $O(n^{d-1})$ time [2]. For $d = 2$, we have to construct a 1-dimensional arrangement. This boils down to sorting the endpoints of the shadow regions on β, which takes $O(n \log n)$ time. After constructing the arrangement, Step 2 can be done in $O(n^{d-1})$ time, by traversing the dual graph of $\mathcal{A}(H)$ and maintaining the number of shadow regions containing the cells as we move from cell to cell in the dual graph. □

The general algorithm can be adapted so as to handle other types of sites. The time complexity again depends on the running time of the subroutine that computes the arrangement of the shadow-region boundaries on the bisector of the red sites.

Note that even in the plane, for some types of sites such as disks and ellipses, a shadow regions can consist of more than one connected component [3]. If, however, the number of connected components of each shadow region is bounded by a constant, then the number of intervals in the overlay of the blue shadow regions is linear (in the case $d = 2$). Even if β and the blue shadow regions are not of linear algebraic nature, this does not affect the time complexity of the algorithm, assuming that the necessary basic algebraic computations still take constant time.

Next we show lower bounds for Min Voronoi Connectivity.

Theorem 2 Min Voronoi Connectivity *with two red sites in* $\mathbb{R}^2$ *is in* $\Omega(n \log n)$.

Proof. Consider the problem ε-Closeness, which is defined as follows:

ε-Closeness
Input: A set $\mathcal{X}$ of n reals and a real $\varepsilon > 0$.
Output: yes if there are at least two elements in $\mathcal{X}$ whose distance is less than ε, no otherwise.

ε-Closeness has been proven to be in $\Omega(n \log n)$ for the linear decision tree model [4] and with similar arguments the same lower bound can be proven in the fixed-order algebraic decision-tree model [1]. We shall now describe a linear-time reduction from ε-Closeness to Min Voronoi Connectivity for point sites in $\mathbb{R}^2$ with $|\mathcal{R}| = 2$.

Let $(\mathcal{X}, \varepsilon)$ be an instance of ε-Closeness. We create an instance of Min Voronoi Connectivity as follows. Let $\mathcal{R} = \{(0,1),(0,-1)\}$. The bisector $\beta : y = 0$ of the two red sites represents the real axis for the instance of ε-Closeness. For each $\xi \in \mathcal{X}$ we construct two blue sites such that their shadow regions are the rays $\{(x,0) : x \leq \xi - \varepsilon\}$ and $\{(x,0) : x \geq \xi + \varepsilon\}$. Clearly, our reduction takes linear time.

The set $\mathcal{X}$ is a yes instance of ε-Closeness if and only if Min Voronoi Connectivity with input $\mathcal{R}$ and $\mathcal{B}$ has a solution that eliminates less than $n-1$ blue sites. □

Point sites in 3-space. We now prove that Min Voronoi Connectivity in $\mathbb{R}^3$ is 3sum-hard. The class of 3sum-hard problems was introduced by Gajentaan and Overmars [5]. Similar to the conjectured computational intractability of np-hard problems, 3sum-hard problems are conjectured to not allow for subquadratic algorithms (depending on the model of computation). One can show that a problem Π is 3sum-hard by giving a reduction that transforms in $o(n^2)$ time instances of a known 3sum-hard problem Π' to instances of Π.

Theorem 3 Min Voronoi Connectivity *with two red sites in* $\mathbb{R}^3$ *is* 3sum*-hard.*

Proof. We define a *strip* to be the area between two parallel lines. We consider the following problem:

Strips Cover Box
Input: A set $\mathcal{S}$ of n strips in the plane and an axis-parallel rectangle ρ.
Output: yes if the union of the strips completely covers the area of ρ, no otherwise.

Strips Cover Box is 3sum-hard [5]. We give a linear-time reduction from Strips Cover Box to Min Voronoi Connectivity for point sites in $\mathbb{R}^3$ with $|\mathcal{R}| = 2$.

Let $(\mathcal{S}, \rho)$ be an instance of Strips Cover Box. We create an instance of Min Voronoi Connectivity as follows. Let $\mathcal{R} = \{(0,0,1),(0,0,-1)\}$. The bisector $\beta : z = 0$ of the two red sites represents the plane that contains ρ and the strips in $\mathcal{S}$.

For each edge e of ρ, we construct $n+1$ blue sites such that their shadow regions are identical, having a boundary on β that coincides with the support line of e, and they do not contain ρ. We can make multiple sites have the same shadow region by placing them on the perimeter of a circle that lies on a plane orthogonal to β. Thus, every point of β outside ρ is covered by at least $n+1$ sites. For each strip in $\mathcal{S}$ we construct two blue sites such that the intersection of their shadow regions on β coincides with the strip. Clearly, our reduction takes linear time.

The instance $(\mathcal{S}, \rho)$ is a yes-instance for Strips Cover Box if and only if any solution to Min Voronoi Connectivity with input $\mathcal{R}$ and $\mathcal{B}$ eliminates at least $n+1$ blue sites. This shows that in 3-space Min Voronoi Connectivity is 3sum-hard. □

3 The General Case

We now investigate the complexity of the decision version of Min Voronoi Connectivity for point sites in $\mathbb{R}^2$ where we drop the restriction that $|\mathcal{R}| = 2$. In other words, we consider the following problem:

Voronoi Connectivity
Input: Two sets $\mathcal{R}$ and $\mathcal{B}$ of point sites in the plane and a natural number k.
Output: yes if there exists a connectivity set $\mathcal{B}_{\text{del}} \subseteq \mathcal{B}$ with $|\mathcal{B}_{\text{del}}| \leqslant k$, no otherwise.

For a given undirected graph $G = (V, E)$ a set of vertices $C \subseteq V$ is a *connected vertex cover* if the subgraph induced by C is connected and C is a vertex cover of G, that is, C contains at least one endpoint of each edge of G. To show that Voronoi Connectivity is np-hard, we reduce from the following special case of connected vertex cover, which is np-hard [6].

Planar Connected Vertex Cover
Input: A planar 2-connected graph $G = (V, E)$ of maximum degree 4 and a positive integer k.
Output: yes if there exists a connected vertex cover of G that consists of at most k vertices, no otherwise.

The reduction. Let (G, k) be an instance of Planar Connected Vertex Cover. Our approach is as follows. We first construct a rectilinear embedding of G on a grid of size polynomial in $n = |V|$. Then we use the grid coordinates of the vertices of G to place the red and blue sites. We prove that in the induced Voronoi diagram we can connect the Voronoi cells of the red sites by deleting at most k blue sites if and only if G has a connected vertex cover of size at most k.

First we compute a planar grid embedding of G using the algorithm of Tamassia and Tollis [7]. Their linear-time algorithm maps the vertices of G to distinct points of an $O(n^2)$-size section of the integer grid and maps the edges of G to non-intersecting rectilinear paths over grid points. To any grid point $p = (x_p, y_p)$ that appears in this embedding, we assign the square $[x_p - 1/2, x_p + 1/2] \times [y_p - 1/2, y_p + 1/2]$. Let $E_1(G)$ denote the set of grid squares occupied by the resulting embedding of G. We subdivide each square $\delta \in E_1(G)$ into a grid of $(2n + 1) \times (2n + 1)$ squares that we denote by $\mathit{grid}(\delta)$. We denote the set of squares of the latter, refined embedding of G by $E_2(G)$. We place either a red or a blue site in the center of each square of $\mathit{grid}(\delta)$. We call such a square a red or blue square according to the color of the site that we placed in its center. We place the red sites so that the red squares in $E_2(G)$ form a rectilinear embedding of G very similar to $E_1(G)$ yet now the vertices and the rectilinear edge paths have the thickness of one square of $E_2(G)$. In Figure 1 we show the patterns in which we place red and blue sites in the grid squares of $\mathit{grid}(\delta)$ for each $\delta \in E_1(G)$. If δ corresponds to a vertex in G, then we place a blue site in the center square of $\mathit{grid}(\delta)$. We call this site a *vertex site*. If δ does not correspond to a vertex, we place a red site in this square. In each grid square in $E_2(G)$ that is not occupied by a red site or a vertex site, we place a blue site. Hence, on each side of a rectilinear path of red squares, there is a "padding" of n disjoint rectilinear paths of blue squares.

Note that in the induced Voronoi diagram no two red neighbors of a vertex site share a common boundary edge of their cells.

We denote the sets of red and blue sites that we place into the squares of $E_2(G)$ by $\mathcal{R}(G)$ and $\mathcal{B}(G)$, respectively. Note that in the Voronoi diagram induced by $\mathcal{R}(G)$ and $\mathcal{B}(G)$, the Voronoi cells of the red sites form one connected component for each embedded edge of G. We call these components *edge components*. We can construct $\mathcal{R}(G)$ and $\mathcal{B}(G)$ in time polynomial in n.

Lemma 4 *The Voronoi diagram of $\mathcal{R}(G)$ and $\mathcal{B}(G)$ has a connectivity set of size at most k if and only if G admits a connected vertex cover of size at most k.*

Proof. "if": Let C be a connected vertex cover of G, and let $\mathcal{B}_{\text{del}} \subseteq \mathcal{B}(G)$ be the set of vertex sites that correspond to the vertices in C. Clearly, $|C| = |\mathcal{B}_{\text{del}}|$.

We argue that $\mathcal{B}_{\text{del}}$ is a connectivity set for $\text{Vor}(\mathcal{R}(G) \cup \mathcal{B}(G))$. Take any two red sites $s \neq s'$. Each of these lies on the embedding of an edge of G. Since C is a vertex cover, each of the two edges has an endpoint in C. Since C is a connected vertex cover, the graph induced by C contains a path between the two endpoints. Consider the set of sites that lie between s and s' on the embedding of this path plus the two initial edges. All blue sites in this set lie in $\mathcal{B}_{\text{del}}$. Removing $\mathcal{B}_{\text{del}}$ from $\mathcal{B}(G)$ connects the Voronoi cells of all red sites in the set, in particular, those of s and s'. Hence $\mathcal{B}_{\text{del}}$ is a connectivity set.

"only if": Let $\mathcal{B}_{\text{del}} \subseteq \mathcal{B}(G)$ be a connectivity set for $\text{Vor}(\mathcal{R}(G) \cup \mathcal{B}(G))$. We first assume that all sites in $\mathcal{B}_{\text{del}}$ are vertex sites. Let C be the set of vertices in G that correspond to sites in $\mathcal{B}_{\text{del}}$. Then C is a vertex cover of G. Otherwise there is at least one edge component in $\text{Vor}(\mathcal{R}(G) \cup \mathcal{B}(G) \setminus \mathcal{B}_{\text{del}})$ that has not merged with any of the other components. The graph induced by C must also be connected—otherwise the red cells in $\text{Vor}(\mathcal{R}(G) \cup \mathcal{B}(G) \setminus \mathcal{B}_{\text{del}})$ form more than one connected component.

It remains to examine the case that $\mathcal{B}_{\text{del}}$ contains also padding sites. We show that there is a connectivity set $\mathcal{B}'_{\text{del}} \subseteq \mathcal{B}(G)$ with $|\mathcal{B}'_{\text{del}}| \leq |\mathcal{B}_{\text{del}}|$ that contains more vertex sites than $\mathcal{B}_{\text{del}}$. (We can repeat this argument until we have only vertex sites.) Let c and c' be two distinct edge components in $\text{Vor}(\mathcal{R}(G) \cup \mathcal{B}(G))$.

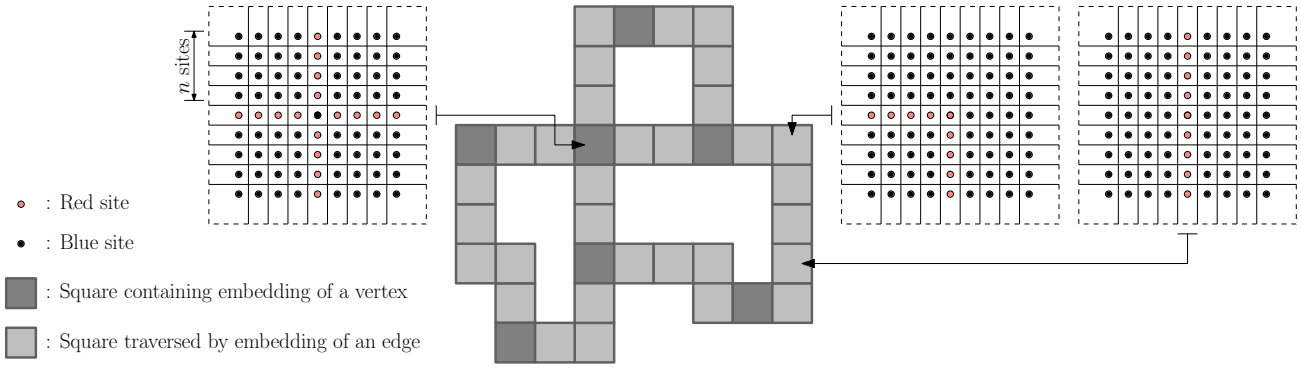


Figure 1: The coarser grid embedding $E_1(G)$ (gray shaded squares) along with the induced Voronoi diagram of the three main possible patterns for $grid(\delta)$ in the reduction of Voronoi Connectivity. The dark gray squares correspond to the vertices of G.

Let s be a site whose cell belongs to c, and let s' be a site whose cell belongs to c'. We consider two cases.

First, suppose the cells of s and s' are adjacent in $\mathrm{Vor}(\mathcal{R}(G) \cup \mathcal{B}(G) \setminus \mathcal{B}_{\mathrm{del}})$. If there is no vertex site whose cell is incident to both c and c' then it is easy to see that at least $2n$ blue sites must be deleted so that the cells of s and s' become adjacent. Thus $|\mathcal{B}_{\mathrm{del}}| \geqslant 2n$, and we simply let $\mathcal{B}'_{\mathrm{del}}$ be the set of all n vertex sites.

Now suppose c and c' are both incident to the square $\delta \in E_1(G)$ of some vertex site s''. If the squares in $E_2(G)$ occupied by s and s' both lie in $grid(\delta)$ then there is at least one blue site s''' in $\mathcal{B}_{\mathrm{del}}$ whose square lies in $grid(\delta)$, too. In that case we let $\mathcal{B}'_{\mathrm{del}} = (\mathcal{B}_{\mathrm{del}} \setminus \{s'''\}) \cup \{s''\}$. If at least one of the squares of s and s' in $E_2(G)$ does not lie in $grid(\delta)$ then again $|\mathcal{B}_{\mathrm{del}}| \geqslant 2n$, and we let $\mathcal{B}'_{\mathrm{del}}$ be the set of all vertex sites. In each case, we have a new connectivity set $\mathcal{B}'_{\mathrm{del}}$ with more vertex sites than $\mathcal{B}_{\mathrm{del}}$ and with $|\mathcal{B}'_{\mathrm{del}}| \leq |\mathcal{B}_{\mathrm{del}}|$. □

We have just proved that Voronoi Connectivity is NP-hard since Planar Connected Vertex Cover is NP-hard. Voronoi Connectivity is in NP since, given $\mathcal{R}$ and $\mathcal{B}$, we can guess a potential solution $\mathcal{B}_{\mathrm{del}}$ with positive probability and then check in time polynomial in $|\mathcal{R}|+|\mathcal{B}|$ whether $\mathcal{B}_{\mathrm{del}}$ is indeed a solution by computing $\mathrm{Vor}(\mathcal{R} \cup \mathcal{B} \setminus \mathcal{B}_{\mathrm{del}})$.

Theorem 5 Voronoi Connectivity *is* NP*-complete.*

Remark 6 The proof of Theorem 5 also holds if the sites in $\mathcal{B}(G)$ and $\mathcal{R}(G)$ are perturbed slightly away from the centers of the squares in $E_2(G)$. Hence, the theorem is also applicable for non-degenerate distributions of the input sites.

4 Concluding remarks

We have introduced the problem Min Voronoi Connectivity, and shown how it can be solved in $O(n \log n + n^{d-1})$ time for two red sites and n blue sites in $\mathbb{R}^d$. The running time of our algorithm is optimal for $d = 2$, and also for $d = 3$ if the conjectured lower bound for 3sum holds. Our algorithm can also be used to compute all connectivity sets which are minimal under inclusion. This allows us to find an optimal connectivity set for any constant number of red sites: for every spanning tree of the complete graph on the red sites, try every combination of inclusion-minimal connectivity sets for the edges. For a non-constant number of red sites, the problem is NP-hard. $O(|\mathcal{B}|)$- and $O(|\mathcal{R}|)$-approximations are not difficult, but we haven't come up with an $O(1)$-approximation.

References

[1] M. Ben-Or. Lower bounds for algebraic computation trees. *Proc. 15th Annu. ACM Symp. Theory Comput. (STOC)*, pp. 80–86, 1983.

[2] H. Edelsbrunner, J. O'Rouke and R. Seidel. Constructing arrangements of lines and hyperplanes with applications. *SIAM J. Comput.*, 15:341–363, 1986.

[3] I.Z. Emiris and M.I. Karavelas. The predicates of the Apollonius diagram: Algorithmic analysis and implementation. *Comput. Geom. Theory Appl.*, 33(1–2):18–57, 2006.

[4] M.L. Fredman and B. Weide. On the complexity of computing the measure of $\bigcup[a_i, b_i]$. *Commun. ACM*, 21(7):540–544, 1978.

[5] A. Gajentaan and M. Overmars. On a class of $O(n^2)$ problems in Computational Geometry. *Comput. Geom. Theory Appl.*, 5(3):165–185, 1995.

[6] P.K. Priyadarsini and T. Hemalatha. Connected vertex cover in 2-connected planar graph with maximum degree 4 is NP-complete. *Int. J. Math. Phys. Eng. Sci.*, 2(1):51–54, 2008

[7] R. Tamassia and I.G. Tollis. Planar grid embedding in linear time. *IEEE Trans. Circuits Syst.*, 36(9):1230–1234, 1989.

Approximation algorithms for free-label maximization

Mark de Berg* Dirk H.P. Gerrits*

Abstract

Inspired by applications where moving objects have to be labeled, we consider the following (static) point labeling problem: given a set P of n points in the plane and labels that are unit squares, place a label with each point in P in such a way that the number of free labels (labels not intersecting any other label) is maximized. We develop efficient constant-factor approximation algorithms for this problem, as well as PTASs, for various label-placement models.

1 Introduction

Map labeling involves associating textual *labels* with certain *features* on a map such as cities (points), roads (polylines), and lakes (polygons). Manually performing this task is estimated to take cartographers 50% of the total time in creating a map [11]. It is therefore unsurprising that map labeling was listed as an important research area in the ACM Computational Geometry Impact Task Force report [4], and has generated a lot of algorithmic research, especially for point features. See for instance the on-line Map Labeling Bibliography [15], currently containing 371 references.

Label models. A good labeling for a point set has legible labels, and an unambiguous association between the labels and the points. The latter puts restrictions on the shape of labels and the way they can be placed in relation to points. Various such *label models* have been proposed, most often with labels assumed to be axis-aligned rectangles slightly larger than the text they contain.

In the *fixed-position models*, every point has a finite number of *label candidates* (often 4 or 8), each being a rectangle having the point on its boundary. In particular, in the 1-position (1P) model one designated corner of the label must coincide with the point, in the 2-position (2PH, 2PV) models there is a choice between two adjacent corners, and the 4-position (4P) model allows any corner of the label to coincide with the point (see the upper-left 2x2 block in Figure 1). The *slider models*, introduced by Van Kreveld et al. [14] generalize this. In the 1-slider (1SH, 1SV) models one side of the label is designated, but the label may contain the point anywhere on this side. In the 2-slider (2SH, 2SV) models there is a choice between two opposite sides of the label, and in the 4-slider (4S) model the label can have the point anywhere on its boundary (see the fourth row and column in Figure 1). Erlebach et al. [5] introduced terminology analogous to the slider models for fixed-position models with a non-constant number of positions (1MH, 1MV, 2MH, 2MV, 4M; see the third row and column in Figure 1).

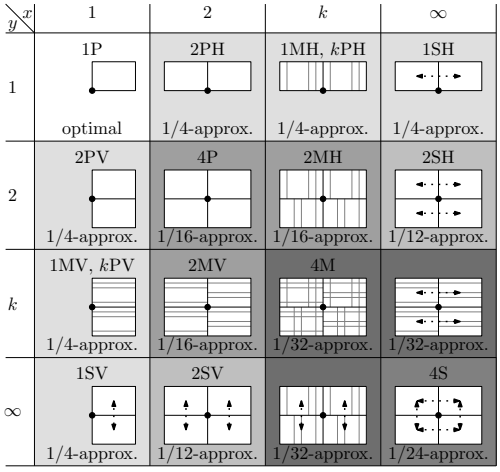


Figure 1: The fixed-position and slider models, and our approximation results for them for the free-label-maximization problem (assuming unit-square labels). The x-axis (y-axis) indicates the number of allowed horizontal (vertical) positions for a label.

Static labeling. Intersecting labels and small font sizes hinder legibility. The *size-maximization problem* asks to label all points with pairwise non-intersecting labels of maximum size. For a given placement of the labels it is a fairly simple geometric task to find the optimal scale factor, so the problem can be solved optimally for the 1P model. For two or more label candidates the problem is APX-hard, even for unit-square labels [6]. Constant-factor approximation algorithms exist for various label models [6, 9].

The more widely studied *number-maximization problem* asks to label a maximum-cardinality *subset* of the n points with pairwise non-intersecting labels of *given* dimensions. Even if all labels are unit squares, this problem is known to be strongly NP-hard for the 1P [7], 4P [6, 10], and 4S models [14]. A generalization of this problem concerns weighted points [12] and asks for a maximum-weight subset

*Dept. of Computer Science, TU Eindhoven, the Netherlands, mdberg@win.tue.nl, dirk@dirkgerrits.com

of the points to be labeled so that, for example, a big city will more likely get a label than a small town. For unit-height rectangular labels this problem admits a polynomial-time approximation scheme (PTAS) for static points in all fixed-position and slider models, both in the weighted [5, 12] and the unweighted case [1, 14]. For arbitrary rectangles in the unweighted case an $O(1/\log\log n)$-approximation algorithm is known for the fixed-position models [3], but the slider models, the weighted case, and the (non-)existence of a PTAS remain open problems.

Labeling of moving points. Mobile devices with interactive displays and GPS have vastly increased both the availability of motion data and our ability to view them. An important aspect of displaying such data is the association of textual labels with points of interest. Yet, despite the large body of work on labeling static points, virtually no results have been published on labeling moving points. Been et al. [2] studied the unweighted number-maximization problem for static points under continuous zooming in and out by the viewer, which can be seen as points moving on a very specific kind of trajectories. Rostamabadi and Ghodsi [13] studied how to quickly flip and scale the labels of static points to avoid one moving point.

There is not only a lack of results on labeling moving points, but in fact the size- and number-maximization problems are ill-suited to moving points. Continuously scaling labels under size maximization would be hard to read, and the (dis)appearance of a label under number maximization can be disturbing for the viewer. We instead propose the *free-label-maximization problem*, where the labels have given dimensions (as in the number-maximization problem), but all points need to be labeled (as in the size-maximization problem). Instead of disallowing intersections, we want to maximize the number of labels which are not intersected, and call such labels *free*. Ideally, an algorithm for free-label maximization on moving points would move the labels continuously in such a way that the number of free labels is close to the static optimum at all times.

Our results. As a first step towards this goal we have studied the free-label-maximization problem for static points. For unit-square labels we have developed a simple $O(n \log n)$-time, $O(n)$-space constant-factor approximation algorithm, as well as a PTAS. This makes free-label maximization easier than size maximization, as the latter is APX-hard even for unit-square labels. In contrast, techniques used for (approximate) number maximization for unit-square labels easily extend to unit-height labels of different widths, which seems not to be the case for free-label maximization. Thus the complexity of free-label maximization seems to fall in between that of the size- and number-maximization problems.

We present our constant-factor approximation algorithm in the next section, leaving our PTAS to the full version of this paper. The algorithm's approximation guarantees for the various label models are listed in Figure 1. We prove them for the 2PH, 4P, 1SH, 2SH, and 4S label models, the other models being analogous. We assume that no two points have the same x- or y-coordinate, and that labels are open sets (their boundaries may intersect). Neither assumption is essential, but they make our exposition simpler.

2 Constant-factor approximations for unit squares

Consider the algorithm GreedySweep, which works as follows. Going through the points from left to right, we label them one-by-one. We call a label candidate ℓ for a point being processed *freeable* if none of the previously placed labels intersect ℓ, and every point still to be labeled has at least one label candidate that does not intersect ℓ or any previously placed freeable label. We always choose a freeable label candidate if possible, and then also call the resulting label freeable. If a point has no freeable label candidate we pick a non-freeable label candidate that does not intersect any previously placed freeable label (which is always possible by the definition of freeable). In case of ties, we pick the label candidate farthest to the left. (Further ties between equally leftmost label candidates can be broken arbitrarily.)

Lemma 1 *For the free-label-maximization problem with unit-square labels, algorithm* GreedySweep *gives a* 1/4*-approximation for the 2PH and 1SH models and this ratio is tight.*

Proof. Let Opt be some optimal solution, and let Alg be the solution computed by GreedySweep. Now suppose a point p is labeled with a free label ℓ_p^{Opt} in Opt, but that the label candidate ℓ_p^{Opt} was not freeable when p was being processed by GreedySweep. Call a label candidate for a point *rightmost* if it is farthest to the right of all label candidates for that point, and define *leftmost* analogously. Since p and all points that already have a label lie to the left of every unprocessed point p', their labels cannot intersect the rightmost label candidate for p' without intersecting all other label candidates for p' as well. Thus all unprocessed points can be labeled with their rightmost label candidate without intersecting ℓ_p^{Opt}. Hence, ℓ_p^{Opt} not being freeable must be caused by a label $\ell_{p'}^{\text{Alg}}$ (either freeable or not) that was placed earlier. We note that $\ell_{p'}^{\text{Alg}}$ cannot be leftmost. (If the leftmost label candidate for a point p' left of p intersects ℓ_p^{Opt}, then all other label candidates for p' do as well, contradicting that ℓ_p^{Opt} is free in Opt.) That $\ell_{p'}^{\text{Alg}}$ is not leftmost can mean two things. Either $\ell_{p'}^{\text{Alg}}$ is freeable, in which case we charge ℓ_p^{Opt} to $\ell_{p'}^{\text{Alg}}$, or

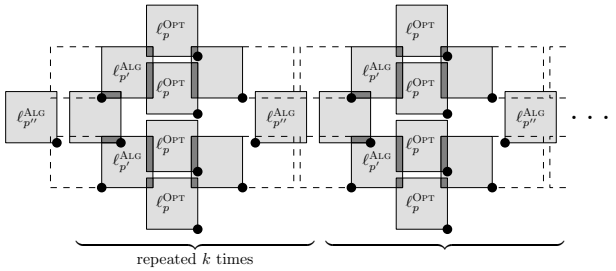


Figure 2: A labeling computed by GreedySweep for the 2PH model, where the $k+1$ labels marked $\ell_{p''}^{\text{ALG}}$ are free. In the optimal solution the $4k$ labels marked ℓ_p^{OPT} are free. Thus the 1/4-approximation is tight for the 2PH model, and the same can be shown for the 1SH model by moving the points closer together horizontally.

making $\ell_{p'}^{\text{ALG}}$ leftmost will cause it to intersect some freeable label $\ell_{p''}^{\text{ALG}}$, in which case we charge ℓ_p^{OPT} to $\ell_{p''}^{\text{ALG}}$. With a careful case analysis one can argue that at most four free labels of Opt get charged to a single freeable label of Alg by the above scheme. Fig. 2 shows that resulting approximation ratio is tight.

We still need to consider the case where a point p has a free label ℓ_p^{OPT} in Opt that is also a freeable label candidate when p is being processed by GreedySweep. Then ℓ_p^{ALG} must be a freeable label, and we charge ℓ_p^{OPT} to ℓ_p^{ALG}. The label ℓ_p^{ALG} can at most be as far to the right as ℓ_p^{OPT}, otherwise GreedySweep would have picked ℓ_p^{OPT} over ℓ_p^{ALG}. One can argue that this implies ℓ_p^{ALG} will only be charged once. □

Already for the 4P model, GreedySweep can be as bad as an $O(1/\sqrt{n})$-approximation. We instead take the best solution over running GreedySweep several times with different sweep directions. For the 4P model we do one left-to-right sweep (as before) and one right-to-left sweep (preferring rightmost label candidates). For the 2SH model we do one top-to-bottom sweep (preferring topmost label candidates) and one bottom-to-top sweep (preferring bottommost label candidates). For the 4S model we sweep in all four of these directions. This yields the following:

Theorem 2 *There are $O(n \log n)$-time and $O(n)$-space algorithms for free-label maximization on n points with unit-square labels, having the following approximation ratios:* 1/4 *(tight) for the 2PH and 1SH models,* 1/12 *for the 2SH model,* 1/16 *for the 4P model, and* 1/24 *for the 4S model.*

Proof. We will prove the approximation ratio for the 4P model; the proofs for the 2SH and 4S models are similar, and the ratio for the 2PH and 1SH models was proved in Lemma 1. Let Opt be an optimal solution for the 4P model, and consider the solution Alg computed in the left-to-right sweep. We can assume that at least half of the labels in Opt are placed in one of the two rightmost positions. (If not, at least half must be placed in one of the two leftmost positions and we can instead consider the right-to-left sweep in a completely symmetric way.) We will argue that the rightmost free labels in Opt can be charged to free labels of Alg so that no label receives more than eight charges, yielding the stated 1/16-approximation.

Suppose p is a point with a rightmost free label ℓ_p^{OPT} in Opt, but with a non-free label ℓ_p^{ALG} in Alg. At the time p was being processed, the label candidate ℓ_p^{OPT} must not have been freeable, either because some unprocessed point would inevitably get a label intersecting ℓ_p^{OPT}, or because some processed point already had a label intersecting ℓ_p^{OPT}. We consider these two cases separately.

(1) Suppose every label candidate of some unprocessed point p' intersects either ℓ_p^{OPT} or some previously placed freeable label. (This cannot occur in the 2PH and 1SH models.) Of the rightmost label candidates for p' one must be topmost, say $\ell_{p'}^{\wedge}$, and one must be bottommost, say $\ell_{p'}^{\vee}$. Since p and all points that already have a label lie to the left of p', if ℓ_p^{OPT} or a freeable label intersects a rightmost label candidate for p', then it also intersects the label candidate(s) for p' with the same y-coordinate but lying more to the left. So if all rightmost label candidates for p' are intersected by previously placed freeable labels, then all label candidates for p' are intersected by previously placed freeable labels, meaning that at least one of them was in fact not freeable. Thus ℓ_p^{OPT} must intersect some rightmost label candidate of p'. This implies that ℓ_p^{OPT} does not intersect the horizontal line through p', for otherwise ℓ_p^{OPT} would contain p'. Thus ℓ_p^{OPT} intersects either $\ell_{p'}^{\wedge}$ or $\ell_{p'}^{\vee}$ but not both, so there must be a freeable label $\ell_{p''}^{\text{ALG}}$ in Alg which intersects $\ell_{p'}^{\vee}$ if ℓ_p^{OPT} intersects $\ell_{p'}^{\wedge}$, or vice versa. Charge ℓ_p^{OPT} to $\ell_{p''}^{\text{ALG}}$. One can argue that any freeable label can be charged at most twice this way (see Figure 3(b)).

(2) Suppose some already processed point p' has a label $\ell_{p'}^{\text{ALG}}$ (either freeable or not) that intersects ℓ_p^{OPT}. Because ℓ_p^{OPT} is rightmost, $\ell_{p'}^{\text{ALG}}$ cannot be leftmost. So either $\ell_{p'}^{\text{ALG}}$ is freeable, and we charge ℓ_p^{OPT} to $\ell_{p'}^{\text{ALG}}$, or making $\ell_{p'}^{\text{ALG}}$ leftmost will cause it to intersect some freeable label $\ell_{p''}^{\text{ALG}}$, and we charge ℓ_p^{OPT} to $\ell_{p''}^{\text{ALG}}$. One can argue that any freeable label can be charged at most six times this way for the 4P model (see Figure 3(a)).

Combining the charges of these two cases yields at most eight charges per free label for the 4P model, and we argued that at least one half the free labels in Opt could be charged, yielding the claimed 1/16-approximation. We have not yet charged free labels in Opt which label points that also have a free label in Alg. One can argue that charging such labels does

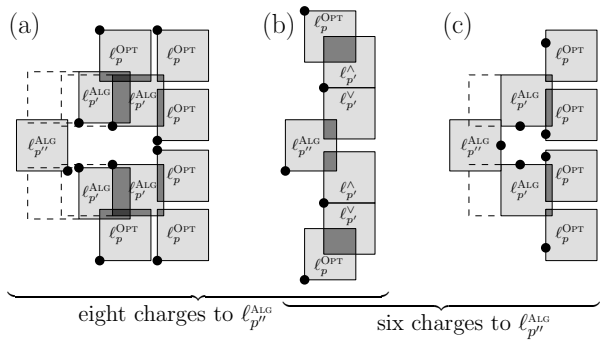


Figure 3: (b) If every ℓ_p^{OPT} charged to $\ell_{p''}^{\text{ALG}}$ is intersected by labels placed *later*, $\ell_{p''}^{\text{ALG}}$ is charged at most twice. If every ℓ_p^{OPT} charged to $\ell_{p''}^{\text{ALG}}$ is intersected by labels placed *earlier*, $\ell_{p''}^{\text{ALG}}$ is charged (a) at most six times for the 4P model, and (c) at most four times for the 2SH and 4S models.

not cost us extra charges, as one of the charges to $\ell_{p''}^{\text{ALG}}$ in Figure 3(b) must disappear if $\ell_{p''}^{\text{OPT}}$ is free.

The proofs for the 2SH and 4S models are similar, but each free label can get at most six charges (see Figure 3(b)–(c)). In the 2SH model every free label in OPT is either topmost or bottommost so that we can again charge at least half of them, but in the 4S model a label can also be leftmost or rightmost so that we can charge only one fourth.

With some clever use of standard data structures, similar to the 1/2-approximation algorithm for number maximization by Van Kreveld et al. [14], GREEDY-SWEEP can be implemented to run in $O(n \log n)$ time and $O(n)$ space. We omit the details. □

3 Conclusion

We have presented a simple and efficient constant-factor approximation algorithm for a new variant of the labeling problem motivated by the wish to label moving points. Our algorithm works for the case where all labels are unit squares (or, equivalently, if all labels are rectangles of the same dimensions). For this case we also developed a PTAS using a variation on the "shifting technique" due to Hochbaum and Maass [8]. Details can be found in the full version of this paper. The cases of labels being unit-height rectangles or arbitrary rectangles are still open. For the number-maximization problem these cases allow, respectively, a PTAS and an $O(1/\log\log n)$-approximation, and it would be interesting to see if these results can be matched. If not, the free-label-maximization problem is strictly harder than the number-maximization problem, while easier than the size-maximization problem. The weighted version of the free-label-maximization problem is another interesting direction for future research.

References

[1] P. Agarwal, M. van Kreveld, and S. Suri. Label placement by maximum independent set in rectangles. *Comput. Geom. Theory Appl.*, 11:209–218, 1998.

[2] K. Been, M. Nöllenburg, S.-H. Poon, and A. Wolff. Optimizing active ranges for consistent dynamic map labeling. *Comput. Geom. Theory Appl.*, 43(3):312–328, 2010.

[3] P. Chalermsook and J. Chuzhoy. Maximum independent set of rectangles. In C. Mathieu, editor, *Proc. 20th ACM-SIAM Sympos. on Discrete Algorithms (SODA'09)*, pages 892–901, New York, 2009.

[4] B. Chazelle and 36 co-authors. Application challenges to computational geometry: CG impact task force report. Technical Report TR-521-96, Princeton University, 1996.

[5] T. Erlebach, T. Hagerup, K. Jansen, M. Minzlaff, and A. Wolff. Trimming of graphs, with an application to point labeling. In S. Albers and P. Weil, editors, *Proc. 25th Internat. Sympos. Theoretical Aspects Comput. Sci. (STACS'08)*, pages 265–276, Bordeaux, 2008.

[6] M. Formann and F. Wagner. A packing problem with applications to lettering of maps. In *Proc. 7th Annu. ACM Sympos. Comput. Geom. (SoCG'91)*, pages 281–288, North Conway, 1991.

[7] R. Fowler, M. Paterson, and S. Tanimoto. Optimal packing and covering in the plane are NP-complete. *Information Processing Letters*, 12:133–137, 1981.

[8] D. Hochbaum and W. Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *Journal of the ACM*, 32(1):130–136, 1985.

[9] M. Jiang, S. Bereg, Z. Qin, and B. Zhu. New bounds on map labeling with circular labels. In *Proc. 15th Annu. Internat. Sympos. Algorithms Comput. (ISAAC'04)*, pages 606–617, 2004.

[10] J. Marks and S. Shieber. The computational complexity of cartographic label placement. Technical Report TR-05-91, Harvard CS, 1991.

[11] J. Morrison. Computer technology and cartographic change. In D. Taylor, editor, *The Computer in Contemporary Cartography*. Johns Hopkins University Press, 1980.

[12] S.-H. Poon, C.-S. Shin, T. Strijk, T. Uno, and A. Wolff. Labeling points with weights. *Algorithmica*, 38(2):341–362, 2003.

[13] F. Rostamabadi and M. Ghodsi. A fast algorithm for updating a labeling to avoid a moving point. In *Proc. 16th Canadian Conf. Comput. Geom. (CCCG'04)*, pages 204–208, 2004.

[14] M. van Kreveld, T. Strijk, and A. Wolff. Point labeling with sliding labels. *Comput. Geom. Theory Appl.*, 13:21–47, 1999.

[15] A. Wolff and T. Strijk. The Map Labeling Bibliography. `http://liinwww.ira.uka.de/bibliography/Theory/map.labeling.html`, 2009.

On Rectilinear Partitions with Minimum Stabbing Number*

Mark de Berg† Amirali Khosravi†

Abstract

Let S be a set of n points in $\mathbb{R}^d$, and let r be a parameter with $1 \leqslant r \leqslant n$. A rectilinear partition for S is a collection $\Psi(S) := \{(S_1, b_1), \ldots, (S_t, b_t)\}$ such that the sets S_i form a partition of S, each b_i is the bounding box of S_i, and $n/2r \leqslant |S_i| \leqslant 2n/r$ for all $1 \leqslant i \leqslant t$. The (rectilinear) stabbing number of $\Psi(S)$ is the maximum, over all axis-parallel hyperplanes h, of the number of bounding boxes used in $\Psi(S)$ that are intersected by h. We study the problem of finding an optimal rectilinear partition—a rectilinear partition with minimum stabbing number—for a given set S. We obtain the following results.

- Computing an optimal partition is NP-hard.
- There are point sets such that any partition with disjoint bounding boxes has stabbing number $\Omega(r^{1-1/d})$, while the optimal partition has stabbing number 2.
- A 2-approximation algorithm for computing optimal partitions, running in polynomial time if r is a constant.

1 Introduction

Motivation. Range searching is one of the most fundamental problems in computational geometry. In its basic form it can be stated as follows: preprocess a set S of objects in $\mathbb{R}^d$ into a data structure such that the objects intersecting a query range can be reported (or counted) efficiently. A range-searching data structure that is popular in practice is the *bounding-volume hierarchy*, or BVH for short. This is a tree in which each object from S is stored in a leaf, and each internal node ν stores a bounding volume $b(\nu)$ of the objects in its subtree. Often the bounding volume being used is a *bounding box*: the smallest axis-aligned box containing the objects in the subtree. When a BVH is stored on external memory, one usually uses a B-tree [3, Chapter 18] as underlying tree structure; the resulting structure (with bounding boxes as bounding volumes) is then called an *R-tree*. R-trees are one of the most widely used external-memory data structures for spatial data, and they have been studied extensively—see for example the book by Manolopoulos *et al.* [5]. In this paper we study a problem related to the construction of R-trees, as explained next.

There are two main strategies to construct R-trees: top-down and bottom-up. A top-down construction algorithm will partition S into a number of subsets S_i, and then recursively construct a subtree $\mathcal{T}_i$ for each S_i. Thus the number of subsets corresponds to the degree of the R-tree. When a range query with a range Q is performed, one has to recursively search in the subtrees $\mathcal{T}_i$ for which the bounding box of S_i (denoted b_i) intersects Q. Note that if $b_i \subset Q$, then all objects from S stored in the subtree $\mathcal{T}_i$ lie inside Q; if, however, b_i intersects ∂Q (the boundary of Q) then we do not know if the objects stored in $\mathcal{T}_i$ intersect Q. Thus the overhead of the search algorithm is determined by the bounding boxes intersecting ∂Q. If Q is a box, as is often the case, then the number of bounding boxes b_i intersecting ∂Q is bounded, up to a factor $2d$, by the maximum number of bounding boxes intersecting any axis-parallel plane. Thus we would like to partition S into subsets so as to minimize the number of bounding boxes intersecting any axis-parallel plane.

Further background and problem statement. Let S be a set of n points in $\mathbb{R}^d$, and let r be a parameter with $1 \leqslant r \leqslant n$. A *rectilinear partition* for S with respect to r is a collection $\Psi(S) := \{(S_1, b_1), \ldots, (S_t, b_t)\}$ such that the sets S_i form a partition of S, each b_i is the bounding box of S_i, and $n/2r \leqslant |S_i| \leqslant 2n/r$ for all $1 \leqslant i \leqslant t$. Note that even though the subsets S_i form a (disjoint) partition of S, the bounding boxes b_i need not be disjoint. The *stabbing number* of an axis-parallel plane h with respect to $\Psi(S)$ is the number of boxes b_i whose interior intersects h, and the *(rectilinear) stabbing number* of $\Psi(S)$ is the maximum stabbing number of any axis-parallel plane h. Observe that our rectilinear partitions are the axis-parallel counterpart of the (fine) simplicial partitions introduced by Matoušek [6].

It is easy to see that there are point sets S for which any rectilinear partition has stabbing number $\Omega(r^{1-1/d})$; this is for example the case when the points in S form a grid of size $n^{1/d} \times \cdots \times n^{1/d}$. Moreover, any set S admits a rectilinear partition with stabbing number $O(r^{1-1/d})$; such a rectilinear partition can be obtained by a construction similar to a kd-tree. Thus from a worst-case and asymptotic point of view the problem of computing rectilinear partitions with

*This research was supported by the Netherlands' Organisation for Scientific Research (NWO) under project no. 639.023.301 and project no. 612.000.631.

†Department of Computing Science, TU Eindhoven. P.O. Box 513, 5600 MB Eindhoven, the Netherlands.

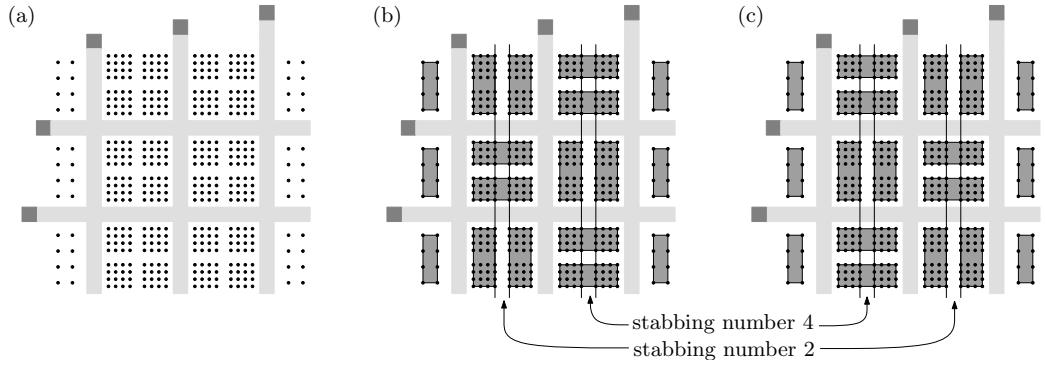


Figure 1: (a) The variable gadget. The dark grey squares are barrier gadgets, and the light grey rectangles indicate that no other bounding box can cross that strip. (b) True setting. (c) False setting.

low stabbing number is solved. However, any specific point set may admit a rectilinear partition with a much lower stabbing number than $O(r^{1-1/d})$. For instance, if the points from S are all collinear, then there is a rectilinear partition with stabbing number 1. The question now arises: given a point S and a parameter r, can we compute a rectilinear partition that is *optimal for the given input set* S, rather than worst-case optimal? In other words, we want to compute a rectilinear partition that has the minimum stabbing number over all rectilinear partitions for S.

Our results. In Section 2 we show that already in $\mathbb{R}^2$, finding an optimal partition is NP-hard. We then turn our attention to approximation algorithms. We show in Section 3 that algorithms only considering disjoint partitions cannot have a good approximation ratio: there are point sets such that any partition with disjoint bounding boxes has stabbing number $\Omega(r^{1-1/d})$, while the optimal partition has stabbing number 2. Finally, in Section 4 we give a 2-approximation algorithm for computing optimal partitions, which runs in polynomial time if r is a constant.

2 Finding optimal rectilinear partitions is NP-hard

We prove the following problem to be NP-hard.

Optimal Rectilinear Partition
Input: A set S of n points in $\mathbb{R}^2$ and parameters r, k.
Output: YES if S admits a rectilinear partition w.r.t. r with stabbing number at most k, NO otherwise.

Our reduction (from 3-SAT) is similar to the proof by Fekete *et al.* [4] of the NP-hardness of minimizing the stabbing number of a matching on a planar point set.

Let $\mathcal{U} := \{x_1, \ldots, x_m\}$ be a set of m boolean variables, and let $\mathcal{C} := C_1 \wedge \cdots \wedge C_s$ be a CNF formula defined over these variables, where each clause C_i is the disjunction of three variables. The 3-SAT problem is to decide whether such a boolean formula is satisfiable; 3-SAT is NP-hard [7]. Our reduction will be such that there is a rectilinear partition with stabbing number 5 for the Optimal Rectilinear Partition instance if and only if the 3-SAT instance is satisfiable. We only sketch the construction, leaving the proof of correctness to the full version of the paper. We first describe the various gadgets we need, and then explain how to put them together.

The barrier gadget. A barrier gadget is constructed by taking a tiny square, partitioning it into 5×5 subsquares, and placing $2n/r$ points in each of the subsquares. Obviously we can partition these $25 \cdot (2n/r)$ points into 25 subsets in such a way that both the horizontal and the vertical stabbing number of the 25 bounding boxes is 5. We need that any partitioning of these points with stabbing number 5 in fact has horizontal and vertical stabbing number at least 5. (Of course we should take care that "borrowing" points from other parts of the construction is not possible, or at least does not change the argument.) Thus no other bounding box can cross the horizontal strip defined by the lines through the top and bottom of the square, and similarly for the vertical strip defined by the left and right edge.

The variable gadget. Fig. 1(a) shows the variable gadget. The three subsets in the left part of the construction, and the tree subsets in the right part, each contain $n/2r$ points. Because of the barrier gadgets, the points from one subset cannot be combined with other points and must be put together into one rectangle in the partition. The six subsets in the middle part of the construction each contain $4n/r$ points. To make sure the stabbing number does not exceed 5, these subsets can basically be grouped in two different ways. One grouping corresponds to setting the variable to true, the other grouping to false—see Fig. 1(b) and (c). Note that the gadget defines two vertical *slabs*. If the variable is set to true then the

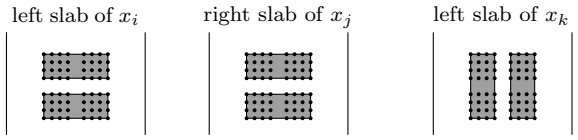


Figure 2: A clause gadget for $(x_i \vee \overline{x_j} \vee x_k)$, and one possible grouping of the points.

left slab has stabbing number 2 and the right slab has stabbing number 4, otherwise the opposite is the case.

The clause gadget. A clause gadget consists of three subsets of $4n/r$ points, arranged as shown in Fig. 2, and placed in the left or right slab of the corresponding variables: a positive literal is placed in the left slab, a negative lateral in the right slab. If the stabbing number of the slab is already 4, which is the case when the literal evaluates to false, then the subset of $4n/r$ points in the clause gadget must be grouped into two "vertical" rectangles. Hence, not all literals in a clause can evaluate to false if the stabbing number is to be at most 5.

The global structure. Fig. 3 shows the global structure. The variable gadgets are placed diagonally. The clause gadgets are placed below the variables. We also place barriers separating the clause gadgets from each other and from the variable gadgets; these are not shown. Finally, the gadgets for occurrences of the same variable in different clauses should be placed such that they are not stabbed by a common vertical line. This concludes our sketch of the construction, which gives the following theorem.

Theorem 1 Optimal Rectilinear Partition *is* np*-complete.*

3 Arbitrary versus disjoint rectilinear partitions

Since computing optimal rectilinear partitions is np-hard, we should look at approximation algorithms. It may be easier to develop an approximation algorithm considering rectilinear partitions with disjoint

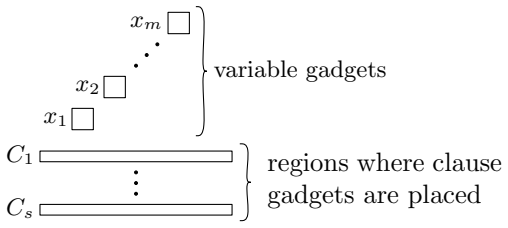


Figure 3: The global structure.

bounding boxes. The next theorem shows that such an approach will not give a good approximation ratio.

Theorem 2 *Let d be a constant, and assume $r < \sqrt{2n}$. Then there is a set S of n points in $\mathbb{R}^d$ whose optimal rectilinear partition has stabbing number 2, while any rectilinear partition with disjoint bounding boxes has stabbing number $\Omega(r^{1-1/d})$.*

Proof. Let $\mathcal{G}$ be a $(r/4)^{1/d} \times \cdots \times (r/4)^{1/d}$ grid in $\mathbb{R}^d$. (We assume for simplicity that $(r/4)^{1/d}$ is an integer.) We put each grid point in S. We call these points *black points*, and we call the hyperplanes forming the grid $\mathcal{G}$ *black hyperplanes*. Note that there are $\Theta(r)$ black points. Fig. 4 shows an example for $d = 2$ with $r = 64$. Next we refine the grid using $d((r/4)^{1/d} - 1)$ additional axis-parallel *grey hyperplanes*; see Fig. 4. At each of the new grid points—the grey dots in the figure—we put a tiny cluster of $2n/r$ points, which we also put in S. If the cluster lies on one or more black hyperplanes, then all points from the cluster lie in the intersection of those hyperplanes, as shown in Fig. 4. (So far we used less than n points; the remaining points can be placed far enough from the construction, not influencing the coming argument.) Next, we rotate the whole construction slightly so that no two points have the same coordinate in any dimension. This rotated set is our final point set S.

To obtain a rectilinear partition with stabbing number 2, we make each of the clusters into a separate subset S_i, and we put the black points into one separate subset; the latter is allowed since $r < 2n/r$. (If $r < n/2r$ we can use some of the remaining points to fill up the subset.) If the clusters are small enough, then the rotation we have applied to the point set guarantees that no axis-parallel hyperplane can intersect two clusters at the same time. Hence, the stabbing number of this rectilinear partition is 2.

We claim that any disjoint rectilinear partition for S has stabbing number $\Omega(r^{1-1/d})$. To see this, observe that no subset S_i in a disjoint rectilinear partition can contain two black points. Indeed, the bounding box of any two black points contains at least one full cluster and, hence, too many points. We conclude that each black point is assigned to a different bounding box. Let B be the collection of these bounding boxes. Now consider a set H of $O(r^{1/d})$ axis-parallel hyperplanes such that each bounding box in B intersects at least one hyperplane from H. (Such a set can be found by duplicating each of the black hyperplanes, and moving the two duplicates of each black hyperplane slightly apart.) Then the total number of intersections between the boxes in B and the hyperplanes in H is r, which implies that there is a hyperplane in H with stabbing number $\Omega(r^{1-1/d})$. □

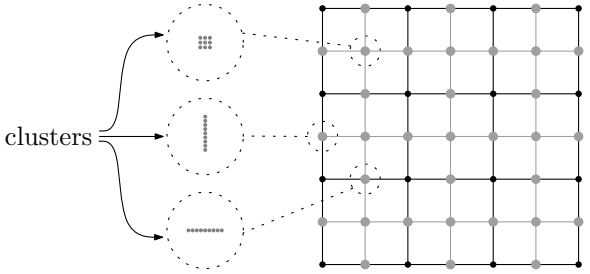


Figure 4: Every rectilinear partition with disjoint bounding boxes has stabbing number $\Omega(\sqrt{r})$ while there exists a partition with stabbing number 2.

4 A 2-approximation algorithm

We present a polynomial-time algorithm that finds, for a given set S and parameter r, a rectilinear partition with stabbing number at most twice the optimal stabbing number. Our algorithm works as follows.

1. For $1 \leqslant i \leqslant d$, let H_i be a collection of $3r$ hyperplanes orthogonal to the x_i-axis such that there are at most $n/3r$ points from S in between any two consecutive hyperplanes in H_i. Let C be the set of all boxes defined by the hyperplanes in $H := H_1 \cup \cdots \cup H_d$. Note that $|C| = O(r^{2d})$.

2. For each t with $r/2 \leqslant t \leqslant 2r$, proceed as follows. Consider all $O(r^{2dt})$ possible subsets $B \subset C$ with $|B| = t$. Check whether B induces a valid solution, that is, whether we can assign the points in S to the boxes in B such that (i) each point is assigned to a box containing it, and (ii) each box is assigned between $n/2r$ and $2n/r$ points. How this is done will be explained later.

3. Over all sets B that induce a valid solution, take the set with the smallest stabbing number. Replace each box in it with the bounding box of the points assigned to it, and report the partition.

Lemma 3 *The above algorithm reports a rectilinear partition with stabbing number at most twice the optimal stabbing number.*

Proof. Let $\Psi := \{(S_1, b_1), \ldots, (S_t, b_t)\}$ be an optimal rectilinear partition for S, and let OPT denote the stabbing number of Ψ. Expand every b_j in all directions until each facet of b_j is contained in a hyperplane from H. Let $\overline{b}_j$ denote the expanded box, and let $\overline{\Psi} := \{(S_1, \overline{b}_1), \ldots, (S_t, \overline{b}_t)\}$. The set $\{\overline{b}_1, \ldots, \overline{b}_t\}$ is one of the subsets B considered in Step 2, and it induces a valid solution. Hence, the stabbing number of the reported partition is at most the stabbing number of $\overline{\Psi}$.

Now consider any axis-parallel hyperplane h. Assume without loss of generality that h is orthogonal to the x_1-axis and that h lies in between hyperplanes $h_i, h_{i+1} \in H$. Let $\overline{b}_j$ be a box intersecting h. Note that b_j must intersect h_i or h_{i+1} (or both), otherwise b_j contains too few points. Hence, the h intersects at most $2 \cdot \text{OPT}$ boxes $\overline{b}_j$. □

To implement Step 2 we construct a flow network with node set $\{v_{\text{source}}, v_{\text{sink}}\} \cup S \cup B$. The source node v_{source} has an arc of capacity 1 to each point $p \in S$, each $p \in S$ has an arc of capacity 1 to every $b_j \in B$ that contains p, and each $b_j \in B$ has an arc of capacity $2n/r$ to the sink node v_{sink}. The arcs from the boxes to the sink also have (besides the upper bound of $2n/r$ on the flow) a lower bound of $n/2r$ on the flow. The set B induces a valid rectilinear partition if and only if there is an integer flow of n units from v_{source} to v_{sink}. Such a flow problem can be solved in $O(\min(V^{3/2}, E^{1/2})E \log(V^2/E + 2) \log c)$ time [1], where V is the number of vertices in the network, E is the number of edges, and c is the maximum capacity of any arc. We have $V = O(n)$, $E = O(nr)$, and $c = 2n/r$. Since we have to check $O(r \cdot r^{4dr})$ subsets B, the running time is $O(r \cdot r^{4dr} \cdot \min(V^{3/2}, E^{1/2})E \log(V^2/E+2) \log c)$ which is polynomial (assuming r is a constant). Note that by enumerating all the partitions , the running time would already be $\Omega(2^{n/2})$ for $r = 2$.

Theorem 4 *Let S be a set of n points, and r a constant. Then we can compute in polynomial time a rectilinear partition with stabbing number at most* 2OPT*, where* OPT *is the minimum stabbing number of any rectilinear partition for S.*

References

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall, 1993.

[2] B. Chazelle and E. Welzl. Quasi-optimal range searching in spaces of finite VC-dimension. *Discr. Comput. Geom.* 4:467–490 (1989).

[3] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms (2nd edition).* MIT Press and McGraw-Hill, 2001.

[4] S.P. Fekete, M.E. Lübbecke, and H. Meijer. Minimizing the stabbing number of matchings, trees, and triangulations. *Discr. Comput. Geom.* 40: 595–621 (2008).

[5] Y. Manolopoulos, A. Nanopoulos, Y. Theodoridis, and A. Papadopoulos. *R-trees: Theory and Applications.* Series in Adv. Inf. and Knowledge Processing, Springer, 2005.

[6] J. Matoušek. Efficient partition trees. *Discr. Comput. Geom.* 8:315–334 (1992).

[7] M.R. Garey, D.S. Johnson, Computers and Interactibility: A Guide to the Theory of NP-Completness. W.H. Freeman and Co. (1979).

Finding Structures on Imprecise Points

Mark de Berg* Elena Mumford† Marcel Roeloffzen‡

Abstract

An imprecise point is a point in $\mathbb{R}^2$ of which we do not know the location exactly; we only know for each point a region in $\mathbb{R}^2$ containing it. On such a set of imprecise points, structures like the closest pair or convex hull are not uniquely defined. This leads us to study the following problem: Given a structure of interest, a set $\mathcal{R}$ of regions and a subset $\mathcal{C} \subseteq \mathcal{R}$, we want to determine if it is possible to place a point in each region of $\mathcal{R}$ such that the points placed in regions of $\mathcal{C}$ form the structure of interest. We study this problem for the convex hull, with various types of regions. For each variant we either give a NP-hardness proof or a polynomial-time algorithm.

1 Introduction

Many geometric problems involve finding a structure in a set of points in $\mathbb{R}^2$, such as the *convex hull* or the *closest pair*. These structures are well defined for any given set of points and many algorithms are available for computing these structures efficiently [1].

Most real-world data, however, is not exact. It often comes from finite-precision computations or imprecise measurements, such as GPS coordinates. Even though such data specifies coordinates for each point, we do not know the exact location: We only know for each point that it is within some region around the given coordinates. This imprecision can be modeled using a region in space for each point. For example, discs can be used to model the imprecision caused by measurement errors, whereas squares can be used to model imprecision on the coordinates caused by finite precision computations. The input is then a set of regions in $\mathbb{R}^2$, where somewhere inside each region a point is located. Now different placements of points inside their regions can lead to different convex hulls and closest pairs on those points. Thus it is not immediately clear what we want to compute on a set of such imprecise points.

Related work. Löffler and van Kreveld [4] try to find the placement of points which maximizes or minimizes the area or perimeter of the convex hull. They describe algorithms for this with running times ranging from $O(n \log n)$ to $O(n^{13})$ for various restrictions on the input regions. They also prove NP-hardness for finding the convex hull with the largest area and for finding the convex hull with largest perimeter. They did similar work for the smallest bounding box, smallest enclosing circle, width, diameter and closest pair of a set of imprecise points [3].

Löffler and van Kreveld [3, 4] focus on some numerical values, such as the area or distance, of the structures that can be made from a set of imprecise points. They do not consider the combinatorial properties. This is the focus of our work: To determine if a set of imprecise points can induce a structure with certain combinatorial properties. For example, given a set S of imprecise points, we want to determine if it is possible that two given points $p, q \in S$ form the closest pair. Next, we define this more precisely.

Problem description. The problem we study is defined as follows. We have a structure of interest, such as the convex hull or a closest pair, a set $\mathcal{R}$ of regions in $\mathbb{R}^2$ and a subset $\mathcal{C} \subseteq \mathcal{R}$. The question we then want to answer is: Is it possible to place a single point in each region of $\mathcal{R}$ such that the points placed in the regions of $\mathcal{C}$ form the structure of interest?

We use sets of line segments, sets of squares or sets of discs as the input set $\mathcal{R}$. For different structures we may impose additional constraints on these regions, for example that regions have to be disjoint. For the rest of this paper we use $\mathcal{R}$ to denote the input set of regions and $\mathcal{C}$ to denote the given subset of $\mathcal{R}$. We focus on the convex hull as the structure of interest.

Results. We consider three variations of the convex hull problem: the *exact convex hull*, *subset convex hull* and *superset convex hull*. In the exact convex hull problem the points placed in regions of $\mathcal{C}$ should be exactly its vertices, whereas for the subset (and superset) convex hull problem the points placed in regions of $\mathcal{C}$ should be a subset (or superset) of the vertices.

The results for the decision problem to determine if it is possible to place a point in each region such that the points placed in regions of $\mathcal{C}$ form a given structure are summarized in Table 1. Note that $n = |C|$ and $k = |R|$.

The results for the exact and subset convex hull problems can be found in Sections 2 and 3. Results

*Department of Computing Science, TU Eindhoven, m.t.d.berg@tue.nl

†Department of Computing Science, TU Eindhoven, e.mumford@tue.nl

‡Department of Computing Science, TU Eindhoven, m.j.m.roeloffzen@tue.nl

Problem	Regions	Restrictions	Complexity
Possible Closest Pair	line segments	disjoint, unit length, parallel	NP-hard
	squares	disjoint, unit size, axis aligned	NP-hard
	discs	disjoint, unit size	NP-hard
Exact Convex Hull	line segments	none	NP-hard
Superset Convex Hull	line segments	none	NP-hard
	line segments	disjoint, unit size, parallel	$O(k^2 n \log n)$
Subset Convex Hull	line segments	disjoint, parallel	$O(n \log n + k^2)$
	rectangles	disjoint, axis aligned	$O(n \log n + k^{4+\varepsilon})$

Table 1: Results on the decision problems.

on the superset convex hull and closest pair can be found in the master thesis by Roeloffzen [6].

Preliminaries. Given a set $\mathcal{R}$ of regions, a choice of points, one from each region of $\mathcal{R}$, will be called a *placement*. Unless otherwise indicated, a placement refers to a choice of points from the regions of $\mathcal{R}$. Formally, a placement is a function $\pi : \mathcal{R} \to \mathbb{R}^2$ that maps each region $P \in \mathcal{R}$ to a point $\pi(P) \in P$.

We will use calligraphic letters $(\mathcal{R}, \mathcal{C})$ to indicate sets of regions, capital letters (P, Q) to indicate regions, and lower case letters (p, q) to indicate points. Furthermore when a point corresponds to a certain region, the region and point will be indicated by the same letter $(p \in P, q \in Q)$. We denote $|\mathcal{C}|$ by k and $|\mathcal{R}|$ by n. Lastly if S is a set of point then $CH(S)$ is the convex hull of S and $CH_{vert}(S)$ is the set of vertices of $CH(S)$.

2 Exact convex hull

In this section we show that Exact Convex Hull is NP-hard for arbitrary line segments. The convex hull problem on imprecise points is defined as follows.

Exact Convex Hull
Input: A set of regions $\mathcal{R}$ and a subset $\mathcal{C} \subseteq \mathcal{R}$.
Output: yes if there is a placement π for $\mathcal{R}$ such that $\pi(\mathcal{C}) = CH_{vert}(\pi(\mathcal{R}))$, no otherwise.

To prove that Exact Convex Hull is NP-hard we use a reduction from 3-SAT. For a given 3-sat formula ϕ we will construct a set $\mathcal{R}(\phi)$ of line segments and define a subset $\mathcal{C}(\phi) \subseteq \mathcal{R}(\phi)$ such that Exact Convex Hull returns yes on $(\mathcal{R}(\phi), \mathcal{C}(\phi))$ if and only if ϕ is satisfiable.

Literal gadget. Literal gadgets are placed around a circle as indicated by the grey areas in Figure 1a. The gadgets are placed such that the literals of the same clause are next to each other. Each gadget consists of two point regions, P and Q, on the endpoints of the circle arc, a segment region A and a point region L. A point placed at the endpoint a_t of A will correspond to the variable being true, and a point placed at a_f corresponds to the variable being false. All regions

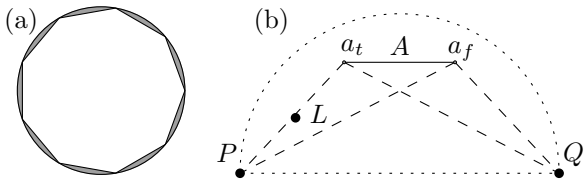


Figure 1: (a) Placement of literal gadgets. (b) Non-negated literal gadget.

except L are in $\mathcal{C}(\phi)$; this also includes the regions in variable and clause gadgets which are described next. Figure 1b shows the gadget for a non-negated literal. A negated literal has basically the same gadget except the endpoints a_t and a_f switch places.

Variable gadget. We want to enforce different occurrences of the same variable to have the same placement: either all should be in the true position (a_t) or all should be in the false position (a_f). We do this by putting in line segments $V_i = w_i v_i$ between consecutive occurrences in a circular manner; Figure 2a and 2b illustrate this for the three occurrences of the variable a. The endpoints v_i and w_i are placed such that any point placed on A will cause at least one of these endpoints, v_i or w_{i+1}, to be inside the convex hull. Since V_i is in $\mathcal{C}(\phi)$ a point on it should be a vertex of the convex hull. Let A_m and A_l be in the literal gadgets where v_i and w_i are located respectively. If a point is placed at $a_{t,m}$ then a point cannot be placed near $a_{f,l}$ since then V_i could not have a point occurring as a vertex of the convex hull. Since all literals of the same variable are connected in a circular manner this implies that either all points are placed around the true endpoints or all at the false endpoints.

Lemma 1 *For all literal gadgets of the same variable, either all points on the line segments A have to be placed near the a_t endpoint or they all have to be placed near the a_f endpoint.*

Clauses. Consider a clause involving three literals, a, b, c. Recall that for each of these literals we have a gadget as in Figure 2b and recall that the point

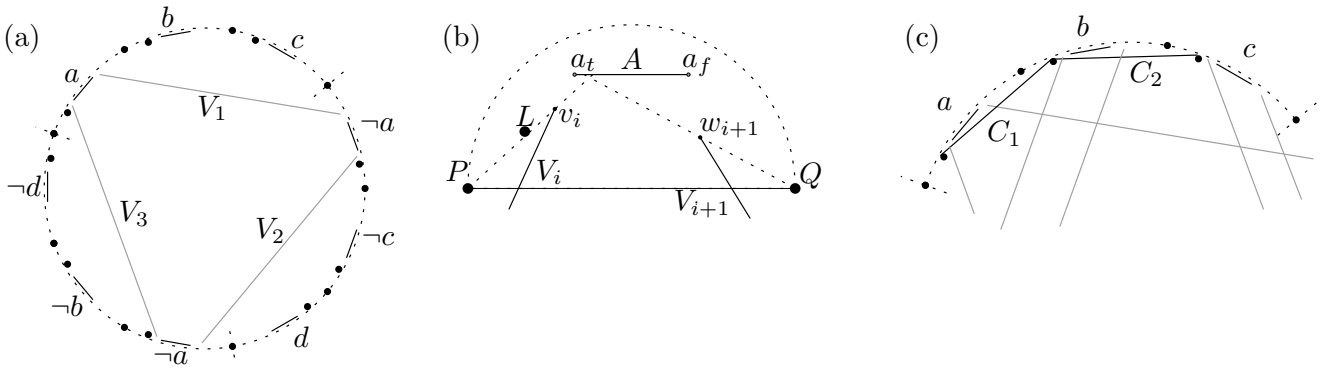


Figure 2: (a) Variable line segments for the variable a. (b) Non-negated literal. (c) Clause line segments.

region L should not be on the convex hull. For the clause involving a, b, c we now add two more segment regions, C_1 and C_2, one going between the gadgets for a and b, and one going between the gadgets for b and c—see Figure 2c. By placing the points for these two segments inside the gadget for a (or b or c) we can ensure that the point for L in a (or b or c) does not show up on the convex hull, as required. However, we only have two clause segments, so we can only enforce this for two of the three literals a, b, c. Preventing the region L for the third literal from showing up on the convex hull can only be done using the A region in that gadget, and requires that literal to be true.

Lemma 2 *For each clause there is a placement that ensures that the point L in any of the literal gadgets of this clause is not a vertex of the convex hull if and only if at least one literal is true.*

Theorem 3 *There is a placement π for $\mathcal{R}(\phi)$, such that $\pi(\mathcal{C}(\phi)) = CH_{vert}(\pi(\mathcal{R}(\phi)))$, if and only if ϕ is satisfiable.*

Proof. If ϕ is satisfiable then it follows from Lemma 1 and 2 that for the placement π_s corresponding to a satisfying assignment it holds that $\pi_s(\mathcal{C}) = CH_{vert}(\pi_s(\mathcal{R}(\phi)))$. If ϕ is not satisfiable then every possible valuation of the variables has a clause that is false. Hence, for every placement for the literal gadgets corresponding to such a valuation there is a clause in which none of the literals is true. Lemma 2 implies that no correct placement is possible for that valuation. Lemma 1 implies that other placements for the literal gadgets, that do not correspond to a specific valuation also do not lead to a solution. □

For every literal only a constant number of line segments is added to $\mathcal{R}(\phi)$. These line segments can be computed in polynomial time, so we conclude:

Theorem 4 Exact Convex Hull *is NP-hard when the input regions are arbitrary line segments.*

3 Subset convex hull for vertical line segments

For Subset Convex Hull we do not require the points placed in regions of $\mathcal{C}$ to be exactly the vertices of the convex hull, but merely a subset of the vertices.

Subset Convex Hull
Input: A set of regions $\mathcal{R}$ and a subset $\mathcal{C} \subseteq \mathcal{R}$.
Output: yes if there is a placement π for $\mathcal{R}$ such that $\pi(\mathcal{C}) \subseteq CH_{vert}(\pi(\mathcal{R}))$, no otherwise.

In this section we describe an algorithm that solves Subset Convex Hull for the case when $\mathcal{R}$ is a set of disjoint vertical line segments. Our algorithm relies on the following observation.

Observation 1 *Let π be a placement for $\mathcal{R}$ such that there is a region P with $\pi(P) \notin CH_{vert}(\pi(\mathcal{R}))$. If P has a point outside $CH(\pi(\mathcal{R}))$ then we can define a placement π' such that $CH_{vert}(\pi'(\mathcal{R})) = CH_{vert}(\pi(\mathcal{R})) \cup \{\pi'(P)\}$. In π' all points are placed the same as in π except the point for P, which is placed such that $\pi'(P) \notin CH(\pi(\mathcal{R}))$.*

Because of this observation it suffices to look at *minimal convex hulls*. A convex hull of a placement π is minimal if there is no placement π' such that $CH(\pi'(\mathcal{R})) \subset CH(\pi(\mathcal{R}))$. We look at the following problem:

Minimal Convex Hull
Input: A set of regions $\mathcal{R}$ and a subset $\mathcal{C} \subseteq \mathcal{R}$.
Output: yes if there is a placement π such that $CH(\pi(\mathcal{R}))$ is minimal and for every region $P \in \mathcal{C}$, either $P \backslash CH(\pi(\mathcal{R})) \neq \emptyset$ or $\pi(P) \in CH_{vert}(\pi(\mathcal{R}))$, no otherwise.

Mukhopadhyay et al. [5] introduce the notions of a bottom and top chain on a set of vertical line segments. The bottom chain is the lower boundary of the convex hull of all upper endpoints of the line segments, whereas the top chain is the upper boundary of the convex hull of lower endpoints. In Figure 3a the top and bottom chains are indicated with dashed lines for a set of vertical line segments.

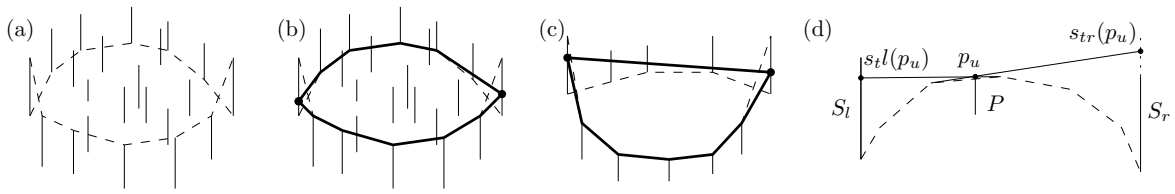


Figure 3: (a) Top and bottom chains. Minimal convex hulls (b) where the top boundary contains part of the top chain and (c) where it is a line between s_l and s_r. (d) $s_{tl}(p_t), s_{tr}(p_t)$ and tangent lines through p_t.

With these chains we can specify every minimal convex hull with two points, one on the leftmost line segment S_l and one on the rightmost line segment S_r. Given a point $s_l \in S_l$ and a point $s_r \in S_r$ we define $CH_{min}(s_l, s_r)$ as follows. Let C_t be the vertices of the top chain that are above the line $s_l s_r$ and C_b the vertices of the bottom chain that are below the line $s_l s_r$ then $CH_{min}(s_l, s_r) = CH(C_t \cup C_b \cup \{s_l, s_r\})$. The proof that shows that every minimal convex hull is equal to $CH_{min}(s_l, s_r)$ for some $s_l \in S_l$ and $s_r \in S_r$ can be found in Roeloffzen's thesis [6].

If we now represent the placement of s_l on S_l and of s_r on S_r by parameters in $[0, 1]$, then we can represent every minimal convex hull by a point in $[0, 1]^2$. We call this the *solution space*. In the Minimal Convex Hull problem we look for a minimal convex hull such that every region $P \in \mathcal{C}$ either contributes a vertex to the convex hull or has some part outside. We define constraints on s_l and s_r such that these constraints are satisfied if an only if P either contributes a vertex or has some part outside the minimal convex hull. The constraints define an area S_P of the solution space. The intersection of S_P over all regions $P \in \mathcal{C}$ gives us the area of the solution space which holds solutions to Minimal Convex Hull.

A region P has a part outside the convex hull if the top endpoint p_t is above the convex hull or the bottom endpoint p_b is below the convex hull. If p_t is below the top chain or on it but not a vertex, then it cannot be above the convex hull. For a top endpoint p_t which is above the top chain or a vertex on it, we define the constraints C_1–C_3. To this end, define $s_{tl}(p_t)$ to be the point on S_l or the vertical extension of it, such that the line through $s_{tl}(p_t)$ and p_t is tangent to the top chain, where the tangent point is not between $s_{tl}(p_t)$ and p_t—see Figure 3d.

(C_1) s_l is below $s_{tl}(p_t)$
(C_2) s_r is below $s_{tr}(p_t)$
(C_3) p_t is above the line $s_l s_r$

With these constraints the following lemma holds. The proof is omitted here due to space limitations.

Lemma 5 *The top endpoint p_t of a line segment P is above the convex hull $CH(s_l, s_r)$ if and only if constraints C_1–C_3 hold, assuming p_t is above the top chain or a vertex of it.*

These constraints (and symmetric ones for the lower endpoint p_b) define the region S_P of the solution space. Since each of the constraints defines a half plane in the solution space S_P is bounded by a constant number of half planes. Therefore, the intersection $S_{sol} = \bigcap_{P \in C} S_P$ can be found by computing the arrangement of the half planes that define the regions S_P and traversing that arrangement. This can be done in $O(k^2)$ time by using a topological sweep [2]. The half planes themselves can be computed in $O(\log n)$ time, because the points $s_{tl}(p_t)$ and $s_{tr}(p_t)$ can be computed in $O(\log n)$ time.

Theorem 6 Subset Convex Hull *can be solved in $O(n \log n + k^2)$ time when the input regions in R are parallel line segments.*

References

[1] M. de Berg, O. Cheong, M. van Kreveld and M. Overmars. Computational Geometry: Algorithms and Applications. Springer-Verlag, 2008.

[2] H. Edelsbrunner and L. Guibas. Topologically sweeping an arrangement. *J. Comput. Syst. Sci.* 38(1):165–194, 1989.

[3] M. Löffler and M. van Kreveld. Largest Bounding Box, Smallest Diameter, and Related Problems on Imprecise Points. In *Proc. WADS*, p. 446–457, 2007.

[4] M. Löffler and M. van Kreveld. Largest and Smallest Convex Hulls for Imprecise Points. *Algorithmica*, 2008.

[5] A. Mukhopadhyay, C. Kumar, E. Greeneand and B. Bhattacharya. On intersecting a set of parallel line segments with a convex polygon of minimum area. *Inf. Proc. Lett.* 105(2):58–64, 2008.

[6] M. Roeloffzen. Finding structures on imprecise points. MSc thesis, TU Eindhoven, 2009, http://www.win.tue.nl/∼emumford/misc/MRoeloffzen.pdf

The time-optimal helicopter trajectory is a circle segment

André Berger * Alexander Grigoriev * Natalya Usotskaya *

Abstract

This paper addresses the problem of determining a time-optimal helicopter trajectory between two points in three-dimensional space without any obstacles. The absolute value of the helicopter speed decreases linearly in altitude, i.e., $v(y) = max\{v_0 - qy, 0\}$, where v_0 is the helicopter velocity at the ground level and q is the velocity loss per one meter altitude. Although intuitively one might think that the optimal trajectory is a straight line, in most of the cases this is not true. We show that the time-optimal trajectory is, in fact, a circle segment. For the simplicity of the proof, we restrict ourselves to the class of continuously differentiable trajectories, although the circle segment is optimal even in the class of continuous functions, a natural assumption to any helicopter trajectory.

1 Introduction

The classical trajectory optimization problems in computational geometry usually assume a constant speed of the moving objects. Moreover, many fundamental geometric problems assume that a moving object, a *mover*, is a single point in two- or three-dimensional space. Under these two assumptions, length-optimal and time-optimal trajectories are the same. Therefore, dealing with a time-optimization problem, we can apply, for instance, the shortest-paths algorithms; see, for instance, [1, 2]. The situation is totally different when we assume that the speed of the mover depends on the position in the space. In this case, length-optimal and time-optimal trajectories can deviate from each other significantly. In this paper we investigate one such problem.

Given a mover in three-dimensional Euclidean space, assume that the absolute value of the mover's speed decreases or increases in one of the space coordinates. This is the real-life setting in any helicopter flight. The greater the altitude of the helicopter, the less the atmospheric pressure and, consequently, the less the air density. In turn, the reduction in air density will reduce the power available, and then, the maximum speed of the helicopter decreases with altitude. Though the (concave) function of the helicopter's maximum velocity in altitude is quite complex and hardly admits a closed analytical form, it is widely accepted to approximate it by linear functions or piece-wise linear functions with one breakpoint [3]. Therefore, we can formulate the *basic helicopter problem* as follows. Given is a source point A and a destination point B in three-dimensional Euclidean space, where, traditionally, the vertical y-axis represents points altitudes while the x- and z-axes represent the surface coordinates. No obstacles (also no ground level) are present. The absolute value of the mover's speed decreases linearly in altitude, i.e., $v(y) = max\{v_0 - qy, 0\}$, where $v_0 > 0$ is some speed intercept (for instance, the helicopter's maximum velocity at the ground level) and $q \geq 0$ is the velocity degradation rate. One has to find a time-optimal trajectory to fly the mover (*helicopter*) from A to B. Notice that this three-dimensional problem can be easily reduced to the two-dimensional problem. This is because the time-optimal trajectory clearly belongs to the plane orthogonal to the surface and containing points A and B.

The *general helicopter problem* reads: given points A and B in three-dimensional Euclidean space, along with a set of polyhedral obstacles, find a time-optimal trajectory to fly the helicopter from A to B without hitting the interior of any of the obstacles. It is noticeable that the general helicopter problem is a generalization of the classical three-dimensional Euclidean shortest-path problem: if $q = 0$, the problems are equivalent. It is well known that the two-dimensional Euclidean shortest-path problem is polynomially solvable [9], the three-dimensional problem is NP-hard [6], but admits polynomial time approximation schemes [1]. Notice that one can easily derive a polynomial-time approximation scheme to the three-dimensional general helicopter problem with obstacles by discretizing/scaling the space and constructing a weighted complete graph, where the edge weight is the time to travel between two vertices (points in the discrete space) using the straight line trajectory. The presence of obstacles can be easily taken into account setting the edge weight to positive infinity if the straight line between two vertices hits interior of an obstacle. Now, we can search for the shortest path in the obtained graph.

In the time-optimization setting, to tackle the general problem with obstacles, one might need a complete characterization of the set of optimal solutions to the basic problem without obstacles, the set of,

*Department of Quantitative Economics, Maastricht University, Netherlands
{a.berger,a.grigoriev,n.usotskaya}@maastrichtuniversity.nl

so-called, *motion primitives*. A typical illustration of lifting from motion primitives to solutions to the general problem can be found, for example, in robotics. In this paper we concentrate on deriving a complete characterization for the basic helicopter problem. We show that the trajectory following a certain circle segment with endpoints A and B is the time-optimal trajectory in the class of continuously differentiable functions. The proof uses Euler-Lagrange equations from the calculus of variations. One can see this also as a variant of the Pontryagin Maximum (Minimum) Principal [10]. This type of techniques is quite common in optimal control theory in general, and in robotics in particular; see, e.g., [4, 5]. This paper leaves two interesting open questions: 1) whether the two-dimensional general helicopter problem admits a polynomial time algorithm; and 2) whether the basic helicopter problem with piece-wise linear velocity degradation admits a polynomial time algorithm.

2 The unique time-optimal trajectory is the circle segment

First, we notice that if the points $A(x_1, y_1)$ and $B(x_2, y_2)$ lay on the same vertical line (i.e., they share the same $x-$coordinate: $x_1 = x_2$), then the optimal trajectory is just a piece of the straight line $x = x_1$ between A and B. We omit the proof as it is straightforward. We also assume that $q > 0$, for otherwise we are in the well-studied setting of the classical Euclidean shortest-path problem. Thus, from now on we consider only the case where $x_1 \neq x_2$ and $q > 0$. We show that in this case the circle segment is the unique time-optimal trajectory. The first observation is about the convexity of any time-optimal trajectory.

Lemma 1 *For any two points A and B in $\mathbb{R}^2 \cap \{(x, y) : v_o - qy > 0\}$, a time-optimal trajectory between A and B is a convex function of x.*

Proof. The proof is based on comparison of the straight line segment between A and B and any concave trajectory above this line. The distance to travel along the straight line is smaller and the velocity increases with the drop of altitude. Hence, the straight line trajectory is better than any concave function. □

Now, we are ready to present the main theorem of this section claiming that there is a unique time-optimal trajectory between A and B, which is a circle segment. The concise proof of Theorem 2 relies on the fact that we choose the polar system of coordinates with an observation point being in the center of the optimal circle segment.

Theorem 2 *Let C be the intersection point of the line $y = \frac{v_0}{q}$ and the line equidistant from A and B. The segment T_{AB} of the circle with center C and radius $R = |CA| = |CB|$ is the unique time-optimal trajectory between A and B. The time needed to travel along T_{AB} is*

$$t_{opt} = t_{T_{AB}} = \frac{1}{q} \ln \left| \frac{\tan \frac{\beta}{2} + \tan \frac{\gamma}{2}}{\tan \frac{\beta}{2} - \tan^2 \frac{\beta}{2} \tan \frac{\gamma}{2}} \right|, \qquad (1)$$

where β is the angle between CA and the x-axis, and γ is the angle between CA and CB.

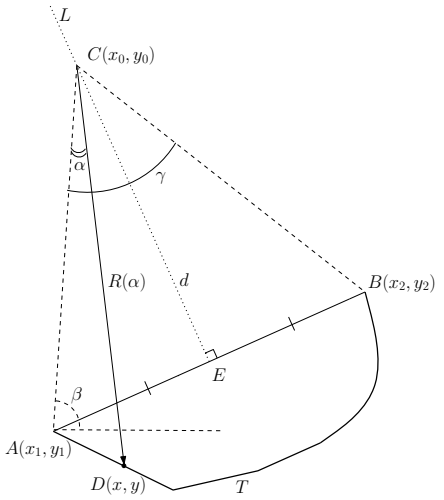


Figure 1: Convex trajectory T in the polar system of coordinates

Proof. By Lemma 1, we can restrict our search for time-optimal trajectories to the class of convex (in x) functions. Let T be an arbitrary continuously differentiable convex trajectory between A and B. Let L be the line equidistant from A and B. Since T is convex in x, it is possible to observe every point of T from any point C on L that lies above the line segment AB. We consider the trajectory T in the polar system of coordinates with observation point $C = (x_0, y_0)$, where $C \in L$ is chosen in such a way that the velocity in C is 0, i.e., $y_0 = \frac{v_0}{q}$.

Consider an arbitrary point $D = (x, y)$ of the trajectory T. In the chosen polar system of coordinates, the point D is completely determined by α and $R(\alpha)$, where α is the angle between CA and CD, and $R(\alpha)$ is the length of the interval CD:

$$x = x_0 - R(\alpha)\cos(\alpha + \beta), \qquad (2)$$
$$y = y_0 - R(\alpha)\sin(\alpha + \beta). \qquad (3)$$

For illustration see Figure 1.

First, we write the integral representing the time needed to travel along T. For a sufficiently small piece of the trajectory, we may assume that the velocity v remains constant within the piece. Let the

length of the piece be denoted by Δx, then the time to travel along the piece is $\Delta t \approx \frac{\Delta x}{v}$. The velocity v is completely determined by the altitude of $D = (x, y)$. Therefore, by Equation (3), we have $v = v_0 - qy = qR(\alpha)\sin(\alpha + \beta)$, as $v_0 - qy_0 = 0$ by choice of C. From mechanics we know that for continuously differentiable trajectories the length of the piece Δx is determined by $\Delta x = \sqrt{R'^2(\alpha) + R^2(\alpha)}\Delta\alpha$. Therefore, the time needed to travel along T is determined by the following integral:

$$t_T = \int_0^\gamma \frac{\sqrt{R'^2(\alpha) + R^2(\alpha)}\, d\alpha}{qR(\alpha)\sin(\alpha + \beta)}, \tag{4}$$

$$t_T = \frac{1}{q}\int_0^\gamma \sqrt{\left(\frac{R'(\alpha)}{R(\alpha)}\right)^2 + 1}\,\frac{d\alpha}{\sin(\alpha + \beta)}.$$

Notice, this integral is well defined as $0 < \alpha+\beta < \pi$ for any $\alpha \in [0, \gamma]$ (β and γ are two of the three angles of the triangle ABC). Furthermore, for the trajectory T we have that $R(0) = R(\gamma) = R = |CA| = |CB|$, as the point C is equidistant from A and B.

Now, we replace $R(\alpha)$ with the function $f(\alpha) = \ln(R(\alpha))$. Since $R(\alpha)$ is continuously differentiable, the function $f(\alpha)$ is continuously differentiable as well. Furthermore, $f'(\alpha) = \frac{R'(\alpha)}{R(\alpha)}$, and the following boundary conditions hold: $f(0) = f(\gamma) = \ln R$, where $R = |CA| = |CB|$. For the function $f(\alpha)$ the travel time is calculated as follows:

$$t_T = \frac{1}{q}\int_0^\gamma \sqrt{1 + f'^2}\,\frac{d\alpha}{\sin(\alpha + \beta)}. \tag{5}$$

Now, we have to find all minimizers of the functional (5). The following theorem of the calculus of variation is used:

Theorem 3 ([8], p. 21; see also [7]) *Given a functional $S(y) = \int_a^b F(t, y(t), y'(t))dt$, where $y : [a, b] \subset \mathbb{R} \to X$ is differentiable and $y(a) = y_a, y(b) = y_b$, y' is the continuous derivative of y and F is a real-valued function with continuous first partial derivatives. If the function $y(t)$ is a stationary point of the functional $S(y)$, then it satisfies the equation:*

$$F_y(t, y(t), y'(t)) - \frac{d}{dt}F_{y'}(t, y(t), y'(t)) = 0. \tag{6}$$

Notice that the trajectory is a circle segment if and only if the radius-vector $R(\alpha)$ is constant, which is possible if and only if the function $f(\alpha) = \ln(R(\alpha))$ is also a constant. Therefore, we can concentrate on Equation (5) proving that $f(\alpha) = const$ is the unique minimizer of the functional.

1) We prove first that the circle segment with the center C and radius $R = |CA|$ is a time-optimal trajectory.

   Consider an arbitrary trajectory T. $f'^2(\alpha) \geq 0$ for any T, therefore, the following lower bound exists:

   $$t_T \geq \frac{1}{q}\int_0^\gamma \frac{d\alpha}{\sin(\alpha + \beta)}.$$

   On the other hand, the last integral represents the time needed to travel along the circle segment T_{AB} with the center C and radius $R = |CA|$. Therefore, the derived circle segment is at least as good as any other trajectory.

2) Now, we show that T_{AB} is the unique time-optimal trajectory in the class of continuously differentiable functions. Consider the functional (5) with the integrand $F(\alpha, f, f') = \frac{\sqrt{1+f'^2(\alpha)}}{\sin(\alpha+\beta)}$. Suppose $f(\alpha)$ is a minimizer of the functional. Therefore, the *Euler-Lagrange Equation* (6) holds for $f(\alpha)$, because it is a necessary condition for the stationary point of any functional. The continuous partial derivatives of the integrand $F(\alpha, f, f')$ are:

   $$\begin{aligned} F_\alpha &= -\frac{\sqrt{1+f'^2}\cos(\alpha+\beta)}{\sin^2(\alpha+\beta)}, \\ F_f &= 0, \\ F_{f'} &= \frac{f'}{\sqrt{1+f'^2}\sin(\alpha+\beta)}. \end{aligned}$$

   We obtain from Equation (6) that $\frac{d}{d\alpha}(F_{f'}) = 0$. Hence,

   $$\frac{f'}{\sqrt{1 + f'^2}\sin(\alpha + \beta)} = const. \tag{7}$$

   Since $\sqrt{1 + f'^2} \geq 1$ and $\sin(\alpha + \beta) > 0$, $(0 < \alpha + \beta < \pi)$, the denominator in Equation (7) is always positive. Now, we consider three cases:

   1. If $const = 0$, then $f' = 0$ and $f(\alpha) = const$. Therefore, $R(\alpha) = e^{f(\alpha)} = const$. The value of this constant is determined by the boundary condition $R = |CA| = |CB|$. This is exactly the circle segment T_{AB} with the center C and the radius $R = |CA|$.
   2. If $const > 0$, then $f' > 0$ and the function $f(\alpha)$ is strictly increasing. Therefore, $f(\gamma) > f(0)$. This contradicts the boundary conditions $f(0) = f(\gamma) = \ln R$. Thus, there are no optimal trajectories, when the constant is strictly positive.
   3. If $const < 0$, then $f' < 0$ and the function $f(\alpha)$ is strictly decreasing. Once again, the contradicts the boundary conditions as $f(\gamma) < f(0)$. Therefore, there are no optimal trajectories, when the constant is strictly negative.

   Summarizing all three cases, there are no other time-optimal radius-vector functions but the circle segment T_{AB} with the center C and the radius $R = |CA| = |CB|$.

The formula for the time needed to travel along the circle segment T_{AB} can be easily computed as follows:

$$\begin{aligned} t_{opt} &= \frac{1}{q}\int_0^{\gamma}\frac{d\alpha}{\sin(\alpha+\beta)} = \frac{1}{q}\int_{\beta}^{\beta+\gamma}\frac{d\tau}{\sin\tau} = \\ &= \frac{1}{q}\int_{\tan\frac{\beta}{2}}^{\tan\frac{\beta+\gamma}{2}}\frac{dt}{t} = \frac{1}{q}\ln\left|\frac{\tan\frac{\beta+\gamma}{2}}{\tan\frac{\beta}{2}}\right| = \\ &= \frac{1}{q}\ln\left|\frac{\tan\frac{\beta}{2}+\tan\frac{\gamma}{2}}{\tan\frac{\beta}{2}-\tan^2\frac{\beta}{2}\tan\frac{\gamma}{2}}\right|. \end{aligned}$$

Here, $\tan\frac{\gamma}{2}$ and $\tan\frac{\beta}{2}$ can be determined in terms of coordinates $A(x_1,y_1)$ and $B(x_2,y_2)$. Point $C(x_0,y_0)$ has an altitude $y_0=\frac{v_0}{q}$. Since C belongs to the line L equidistant from A and B, we have that

$$x_0 = -\frac{y_2-y_1}{x_2-x_1}\frac{v_0}{q} + \frac{x_1^2+y_1^2-x_2^2-y_2^2}{2(x_1-x_2)}.$$

Straightforwardly,

$$R = |CA| = \sqrt{(x_0-x_1)^2+(y_0-y_1)^2},$$

$$d = |CE| = \sqrt{\left(x_0-\frac{x_1+x_2}{2}\right)^2+\left(y_0-\frac{y_1+y_2}{2}\right)^2}.$$

Now, $\tan\frac{\gamma}{2}=\frac{\sqrt{R^2-d^2}}{d}$. Finally, since $\sin\beta=\frac{y_0-y_1}{R}$ and $\cos\beta=\frac{x_0-x_1}{R}$, we derive $\tan\frac{\beta}{2}=\frac{\sin\beta}{1+\cos\beta}=\frac{y_0-y_1}{R+x_0-x_1}$. □

3 Conclusion

In this paper we have addressed the problem of determining a time-optimal helicopter trajectory between two points in three-dimensional Euclidean space, where the speed of the helicopter depends on the flying altitude. We have characterized the time-optimal trajectories which are either line or circle segments.

Two interesting research directions can be followed using the results of this paper. First, it is an open question whether the basic helicopter problem with piece-wise linear velocity degradation admits a polynomial time algorithm (here we have considered linear velocity degradation).

Moreover, the time-optimal line or circle segments are optimal if no obstacles are present. Hence, the second open problem is whether the two-dimensional general helicopter problem with obstacles admits a polynomial time algorithm. As mentioned in the introduction the three-dimensional problem with obstacles is NP-hard as it is a generalization of the Euclidean shortest-path problem with obstacles in three-dimensional space.

Acknowledgments

We thank the organizers in advance for the tasty cookies!

References

[1] P.K. Agarwal, S. Har-Peled, M. Sharir, and K.R. Varadarajan. *Approximating Shortest Paths on a Convex Polytope in Three Dimensions.* JACM, 44(4), 567–584, 1997.

[2] P.K. Agarwal, P. Raghavan, and H. Tamaki. *Motion planning for a steering constrained robot through moderate obstacles.* In Proc. ACM Symposium on Computational Geometry, 343-352, 1995.

[3] *Basic helicopter handbook.* US Department of Transportation, Federal Aviation Administration, Advisory Circular 61-13A, 1973.

[4] J.T. Betts. *Practical Methods for Optimal Control Using Nonlinear Programming.* SIAM Press, Philadelphia, Pennsylvania, 2001.

[5] J.T. Betts. *Survey of Numerical Methods for Trajectory Optimization.* J. of Guidance, Control, and Dynamics, 21(2), 193-207, 1998.

[6] J. Canny and J.H. Reif. *New lower bound techniques for robot motion planning problems.* In Proc. of the 28th Annual IEEE Symposium on Foundations of Computer Science, IEEE, New York, 49–60, 1987.

[7] I.M. Gelfand and S.V. Fomin. *Calculus of Variations.* Dover Publications, Inc., NY, 2000.

[8] I.M. Gelfand and S.V. Fomin. *Calculus of Variations.* Gos. Izd. fiz.-mat. lit., Moscow, 1961.

[9] J.S.B. Mitchell and C.H. Papadimitriou. *Planning shortest paths.* In Proc. SIAM Conference on Geometric Modeling and Robotics, New York, 1–21, 1985.

[10] L.S. Pontryagin, V.G. Boltyanskii, R.V. Gamkrelidze, and E.F. Mishchenko. *The Mathematical Theory of Optimal Processes.* John Wiley, 1962.

A Traveller's Problem

Florian Berger* Rolf Klein*

Abstract

A traveller is planning a tour from some start position, s, to a goal position g in d-dimensional space. Transportation is provided by n carriers. Each carrier is a convex object that results from intersecting finitely many closed linear subspaces; it moves at constant speed along a line. Different carriers may be assigned different velocity vectors. While using carrier C, the traveller can walk at innate speed $v \geq 0$ in any direction, like a passenger on board a vessel. Whenever his current position on C is simultaneously contained in some other carrier C', the traveller can change from C to C', and continue his tour by C'.

Given initial positions of the carriers and of s and g, is the traveller able to reach g starting from s? If so, what minimum travel time can be achieved?

We provide the following answers. For a $1\frac{1}{2}$-dimensional situation similar to the "Frogger" game, where the traveller has to cross a river on which n consecutive rectangular barges move at m different speeds, we provide an $O(n \log m)$ solution. In dimension 8 and higher, the Traveller's Problem is undecidable, even for innate speed zero. An interesting case is in dimension 2. We prove that the problem is NP-hard, even if all carriers are vertical line segments. It turns out that an s-to-g path of finite duration may require an infinite number of carrier changes. Despite this difficulty, we can show that the two-dimensional problem is decidable. In addition, we provide a pseudo-polynomial approximation algorithm.

Keywords: Affine mappings, computational geometry, continuous Dijkstra, frogger, motion planning, NP-hardness, partition, pseudo-polynomial approximation, undecidability.

1 Problem description

Motion planning in dynamic environments is one of the challenging problems in computational geometry. A classical, and important question is how to avoid collision with obstacles that move in time. In this paper we take a different view and consider a situation where moving objects can be used as a means of transportation.

*University of Bonn, Institute of Computer Science I, D-53117 Bonn, Germany.

A *carrier* in dimension d is a non-empty intersection of finitely many closed linear subspaces. Thus, in dimension 2, a carrier can be a point, a line segment, a half-line, a line, a possibly unbounded convex polygon, or the plane itself. Each carrier is assigned its own velocity vector, causing it to move linearly at constant speed. The *traveller* is modelled as a point, p. At time $t = 0$, his journey begins at a start point, s, which is located on some carrier C_s. Like a passenger on board a cruiser, traveller p will be moved along with C_s. In addition, he can walk on C_s (and any other carrier) at maximum speed v, called *innate speed* for short, as long as he does not fall off. If, at some time $t > 0$, traveller p is located in the intersection of C_s and some other carrier, C, he may decide to change from C_s to C, and continue his journey by C. The traveller's ultimate goal is to reach a goal point, g, located on some carrier C_g.

We observe that our definition of changing the carrier allows for the following implementations. First, the traveller can change at some point where two carriers touch each other, since such touch point belongs to either carrier. Second, we could introduce a special carrier C_0 that equals the whole plane and does not move. Then the traveller could, in our model, get off his current carrier anytime and wait for another carrier to arrive or, if $v > 0$, walk some distance in the plane, and board another carrier.

In general it is not clear if g can be reached from s at all. Thus, we are interested in the following questions.

- Given initial positions for the carriers at time $t = 0$, is it possible for the traveller to reach g when starting from s?
- If so, what is the quickest way to get there, using only the carriers for transportation?

2 Crossing a river

As a warm-up example, we consider a special case that resembles the well-known "Frogger" game. Our traveller wants to cross a river whose banks are modelled as two horizontal line-shaped carriers that do not move.

On the river, n rectangular barges are sailing horizontally, at individual speeds and in both directions; see Figure 1.

The ith barge in bottom-up order is B_i. The strips defined by B_i and B_{i+1} touch each other; B_1 and B_n

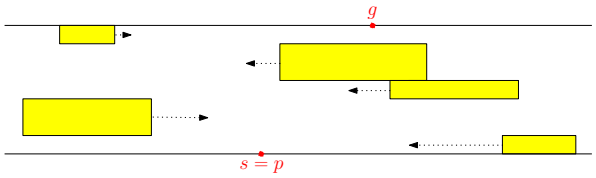


Figure 1: Barges on a river.

touch the lower resp. the upper lines, where start point s resp. goal point g are located. The traveller cannot swim. But, as in the well-known "Frogger" game, he can walk on the barges in X- or Y-direction at maximum innate speed $v = 1$. Also, where two barges touch, he can jump from one to the next.

Starting from s at time $t = 0$, is the traveller able to reach g? If so, what is the earliest possible arrival time? Clearly, this is a special case of the general Traveller's Problem defined in the Introduction.

Using the continuous Dijkstra technique [4] and applying an argument of [3], we obtain

Theorem 1 *Suppose that the n barges are running at m different speeds. Then Traveller's Problem can, in this case, be solved in time $O(n \log m)$.*

3 Higher dimensions

The ease of the solution stated in theorem 1 is owed to the special situation. In this section, we will establish that the general Traveller's Problem is undecidable in dimension $d \geq 8$.

To this end we employ the following recent result by Bell and Potapov [1].

Theorem 2 *The following problem is undecidable [1]. Given five affine mappings $f_1, f_2, \ldots f_5$ from $\mathbb{Q}^2$ to $\mathbb{Q}^2$ and two rational vectors $q = (x, y)$ and $q' = (x', y')$. Is there a finite product of mappings from $\{f_1, f_2, \ldots f_5\}$ that maps q to q'?*

Now we use this result to prove the following.

Theorem 3 *Traveller's Problem with innate speed zero is undecidable in dimension $d \geq 8$.*

Proof. First, we demonstrate how to simulate the application of a single linear function of one variable, $f(X) = a \cdot X$, where $a > 0$, using seven unbounded carriers moving in $XYZV$-space. In our construction, only the directions of their speed vectors matter, not their lengths.

Figure 2 depicts the projection of the corresponding journey to XYZ-space. We start with the traveller at an arbitrary point $e > 0$ on the X-axis. Our first carrier is the XY-plane itself, moving upwards. The traveller can use it to get up to the graph of

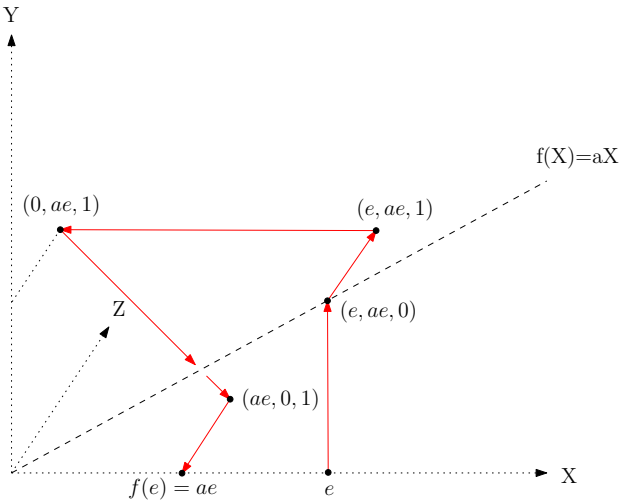


Figure 2: A linear mapping $f : \mathbb{Q} \to \mathbb{Q}$ simulated by carrier movements.

the mapping $f(X) = aX$. Here he can change to the plane $\{(x, ax, z)|x, z \in \mathbb{R}\}$ that moves into direction $(0, 0, 1)$. At $Z = 1$, he can get off, and change to the third carrier—the plane $\{(x, y, 1)|x, y \in \mathbb{R}\}$ moving leftwards. It gets the traveller to $(0, ae, 1)$. At this point, he can enter dimension 4 by means of the plane $\{(0, y, 1, v)|y, v \in \mathbb{R}\}$, which moves in positive V-direction.[1] At $(0, ae, 1, 1)$ the traveller can board the plane $\{(x, y, 1, 1)|x, y \in \mathbb{R}\}$ whose velocity vector equals $(1, -1, 0, 0)$. It gets him to the point $(ae, 0, 1, 1)$, where he can change to the sixth carrier, the plane $\{(x, 0, 1, v)|x, v \in \mathbb{R}\}$ moving in negative V-direction. Back to 3-space at $(ae, 0, 1)$, he can finally catch the plane $\{(x, 0, z)|x, z \in \mathbb{R}\}$; it moves in negative Z-direction and gets the traveller back to the X-axis, at point $ae = f(e)$.

Two observations are crucial. While the definition of carriers does depend on the coefficient a of mapping f, it works for any argument e.[2] Second, if the traveller leaves the X-axis (at some point e), then his only chance of returning to the X-axis is to ride the seven carriers the way explained above. This gets him to point $f(e)$.

Our construction generalizes to two-dimensional affine mappings $f(X_1, X_2)$. Here, we need to make sure that the traveller, after starting from some point (e, g) in the X_1X_2-plane, can return to this plane only at the point $(r, s) = f(e, g)$. This can be achieved by introducing extra dimensions, so that the computations of r and s take place in disjoint affine subspaces and, consequently, do not interfere. We can

[1] This part of the journey is not visible in Figure 2, as it projects onto the point $(0, ae, 1)$. The same holds for the ride back on the 6th carrier.

[2] We must add three alternative carriers to deal with arguments $e < 0$.

deal similarly with five such affine mappings acting independently on the X_1X_2-plane, and thus simulate the problem of Theorem 2. It turns out that dimension 8 provides us with sufficiently many disjoint affine subspaces for this purpose. □

4 Dimension two

4.1 NP-hardness

Our NP-hardness reduction is from the NP-complete problem

PARTITION Given n natural numbers $a_1, a_2, \ldots, a_n$, let $S := \sum_{i=1}^{n} a_i$. Does there exist a subset $I \subset \{1, \ldots, n\}$ such that $S/2 = \sum_{i \in I} a_i$?

Theorem 4 *Traveller's Problem in dimension two is NP-hard for arbitrary innate speed v. There is no constant factor approximation, unless P=NP.*

Proof. Let us first assume that $v = 0$ holds. Our construction uses vertical carriers C, C' and $A_i, B_i, 1 \leq i \leq n$, of height S each. C and C' are aligned and do not move. The bottommost point of C equals s, while g is the center point of C; see Figure 3. For each i, carrier A_i moves in such a way that, at some time, it is congruent with C, and has gained a_i in height on reaching C' afterwards. B_i traverses the horizontal strip containing C, C' from right to left. These carriers pass over C, C' in the order $A_1, B_1, A_2, B_2, \ldots, A_n, B_n$, with plenty of time in between so that no interference is possible.

The traveller, located at some point on C, may choose to board a carrier A_i. It will get him to C', at a point by a_i higher than his point of departure from C. Then, he can use B_i, or one of the later carriers B_j, to return to C while maintaining his height. In other words, for each index i, the traveller has the option to move a distance a_i upwards on C. Thus, he is able to reach its middle point, g if, and only if, a partition of the given numbers is possible.

Now assume that a partition is impossible. Then the traveller misses g by a distance at least $1/2$, since all a_i are natural numbers. If $v = 0$ the traveller will never get to g. Now let $v > 0$, and assume that there exists an approximation algorithm with approximation factor $< \alpha$. By speeding up the carriers we can ensure that $T = \frac{1}{2\alpha v}$ holds for the time T where B_n hits C. At this point, the traveller has walked a total distance of at most $Tv = \frac{1}{2\alpha}$, so that he is at least $d := \frac{1}{2} - \frac{1}{2\alpha}$ away from g, if he has reached C at all. To walk distance d takes time at least $\frac{d}{v} = d2\alpha T = (\alpha - 1)T$, so that the whole journey needs time αT at least. Thus, the approximation algorithm would decide PARTITION. □

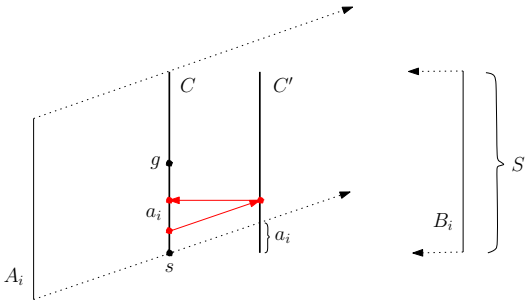


Figure 3: Proving Traveller's Problem NP-hard.

4.2 Decidability

The main difficulty in showing decidability comes from the fact that there are scenes where the goal can be reached in finite time, but only by an infinite number of carrier changes. An example is shown in Figure 4. The three line segments A, B, C are very long, but bounded. Their velocity vectors have large axial and small lateral components. This causes the segments to intersect in point z at some time T. The fast point-shaped carrier D will pass through z at time T, too, and then speed on to meet carrier E that consists just of the goal point, g. The traveller's

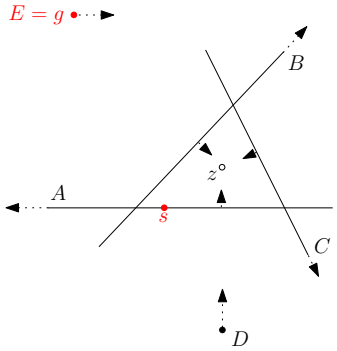


Figure 4: The traveller must pass through the cycle A, B, C infinitely often before he can board carrier D at Zeno point z.

innate speed is zero. He sets out from point s on A. In order to reach g, he needs to catch carrier D, because A's upward movement is too slow, and because staying on A would get the traveller too far to the left, anyway. The only occasion when one of A, B, C contains D is at time T in point z. Until then, the traveller must keep cycling through the ever contracting triangle formed by A, B, C — an infinite number of times! Using a notation from the field of hybrid automata [2], we call z a *Zeno* point, because the situation resembles the paradox of Achilles and the tortoise. At least three carriers are necessary to give rise to a Zeno point and each triple of carriers causes

at most one Zeno point. Hence, their total number is in $O(n^3)$.

Although Zeno points cause complications, they are not strong enough to provide universal computing power in our model. Indeed, we have the following result.

Theorem 5 *Traveller's Problem is decidable in the plane (if all carriers are lines, half-lines, or line segments).*[3]

4.3 Pseudo-polynomial approximation

Now we present a pseudo-polynomial approximation algorithm for the case of bounded carriers in the plane. Let $v = 1$. Let us assume that an s-to-g paths exists in our model, and let P^* be a path of minimum travel time, W. We will establish an algorithm that computes an s-to-g path of almost minimum travel time. However, this path may only be feasible in a *relaxed model*, where the traveller can walk at innate speed $1+\varepsilon$, and use a carrier although it is a distance of μ away. Both relaxation parameters, ε and μ, can be chosen arbitrarily small.

Our algorithm works by discretising time and space. The time difference between two consecutive discrete time points is $\Delta := \frac{\mu}{4+8v_{\max}}$, where $v_{\max}$ denotes the maximum speed of all carriers. Space is discretised by a regular grid of width $\kappa := \frac{1}{4}\min(\mu, \varepsilon\Delta)$.

It is easy to show the following lemma.

Lemma 6 *There exists a function $Q : [0, W] \to \mathbb{R}^2$, such that*

- *$\|Q(t)\| \le \frac{\sqrt{2}}{2}\kappa$ for all $t \in [0, W]$, and*
- *P^*+Q is a path visiting grid points at all discrete time points $j\Delta$, and*
- *$P^* + Q$ is feasible in the relaxed model.*

Thanks to Lemma 6, we need only consider such paths that visit grid points at all times $j\Delta$. This will get us to the goal location at most Δ later than W.

Suppose, the traveller is located at the grid point p at time $j\Delta$. Which grid points can be reached from there at time $(j+1)\Delta$? The crucial idea is to restrict this path planning to a set of *θ-usable* carriers, which contains all carriers having distance at most $\theta := \frac{\sqrt{2}}{2}\kappa + (1+2v_{\max})\Delta$ from p at time $j\Delta$.

If a carrier has distance greater than θ from p at time $j\Delta$, it is redundant by Lemma 6. Namely, the traveller's distance to this carrier exceeds $\frac{\sqrt{2}}{2}\kappa$ not only at time $j\Delta$, but throughout a time interval of length Δ, during which traveller and carrier could get closer by $(1+2v_{\max})\Delta$. On the other hand, if a carrier is at most θ away from p at time $j\Delta$, then the distance cannot exceed $\theta + (1+2v_{\max})\Delta < \frac{3}{4}\mu$ during the time interval from $j\Delta$ to $(j+1)\Delta$. Hence, no matter how the traveller changes between the θ-usable carriers, or walks on them, all extended carrier restrictions are respected.

This freedom allows to linearly combine carrier motions. It turns out that the reachability from p at time $j\Delta$ to q at time $(j+1)\Delta$ is related to the convex hull of the velocity vectors of the θ-usable carriers.

Finally, we obtain:

Theorem 7 *A feasible path in the relaxed model with travel time at most $W + \Delta$ can be computed in running time:*

$$O\left(n\ L\log(L)\ W\ \frac{G}{H}\right)$$

where L denotes the total number of edges of the n carriers, G equals the maximum carrier diameter squared times the maximum speed to the 5th power, and $H = \varepsilon^4\mu^3$.

5 Conclusion

We have introduced a new motion planning problem and shown that its complexity ranges from near linear, in simple cases, to undecidable in higher dimensions. For dimension 2, the problem is decidable, but NP-hard. Its "true" complexity in dimension 2 and 3 remains open. We believe that our pseudo-approximation algorithm can be generalized to higher dimensions. Whether it can be strengthened is another interesting question.

6 Acknowledgement

The authors are grateful to Frank Dehne, Günter Rote, and Jörg-Rüdiger Sack for very fruitful discussions.

References

[1] P. Bell and I. Potapov. On Undecidability Bounds for Matrix Decision Problems. Theor. Comput. Sci. 391(1-2), pp. 3–13, 2008.

[2] K. H. Johanson, J. Lygeros, S. Sastry, and M. Egerstedt. Simulation of Zeno Hybrid Automata. IEEE Conference on Decision and Control, Phoenix, 1999.

[3] F. Leber. Diploma Thesis, in preparation. Bonn, 2009.

[4] J. S. B. Mitchell. L_1-shortest paths among polygonal obstacles in the plane. Algorithmica 8(1), pp. 55–88, 1992.

[3] We put this requirement in brackets because we do not think that it is essential.

The edge rotation graph

Javier Cano* Mayra Corvera Espinoza* José Miguel Díaz-Báñez† Joel Espinosa Longi*
Clemens Huemer‡ Jorge Urrutia§

Abstract

For a given point set V in the plane in general position, consider the set $PG(V,k)$ of non-crossing geometric graphs on V with a fixed number k of edges. For a given non-crossing geometric graph $G=(V,E)$ and an edge $\overline{xy} \in E$, an *edge rotation* of $\overline{xy}$ around x replaces $\overline{xy}$ by an edge $\overline{xw} \notin E$, if the open triangle $\triangle xyw$ does not intersect any edge $e \in E$ nor does it contain any vertex of V. The edge rotation graph has vertex set $PG(V,k)$ and two vertices are adjacent if they differ by an edge rotation. We show that if $PG(V,k)$ contains no triangulations, then the edge rotation graph is connected and has diameter $O(|V|^2)$. This also generalizes to edge-labeled geometric graphs and directed geometric graphs.

1 Introduction

Let V be a finite point set on the plane in general position. A non-crossing geometric graph (also called plane straight line graph) on V is a graph whose vertex set is V and whose edges are straight-line segments with pairwise disjoint interior, joining pairs of elements in V. For the case when the number of edges is maximal, this graphs are triangulations on V. Transformations of triangulations by replacement of edges (edge-flips) has been widely studied among other things, for their applications on generating high quality meshes on V like the Delaunay triangulation [7, 8], or in enumeration of all triangulations on V [2]. We study non-crossing geometric graphs with fewer edges than in a triangulation and consider another operation to locally transform geometric graphs.

*Posgrado en Ciencia e Ingeniería de la Computación, UNAM, México, j_cano@uxmcc2.iimas.unam.mx, m_corvera@uxmcc2.iimas.unam.mx, j_espinosa@uxmcc2.iimas.unam.mx

†Departamento de Matemática Aplicada II, Universidad de Sevilla, Spain, dbanez@us.es. Partially supported by Project MEC MTM2009-08652.

‡Departament de Matemàtica Aplicada IV, Universitat Politècnica de Catalunya, clemens.huemer@upc.edu. Partially supported by Projects MEC MTM2009-07242 and Gen. Cat. DGR 2009SGR1040.

§Instituto de Matemáticas, Universidad Nacional Autónoma de México, urrutia@matem.unam.mx, partially supported by grants CONACyT CB-2007/80268, and FEDER-MTM2006-03909.

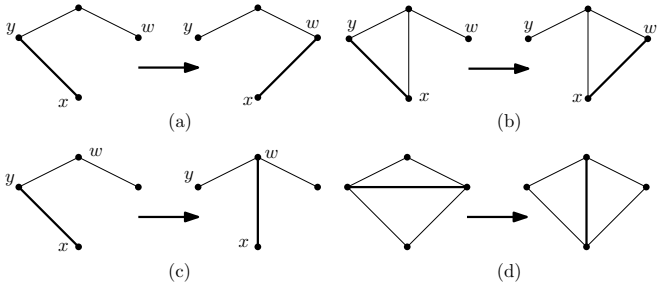


Figure 1: Three types of edge rotations (a-c) and an edge flip (d).

Given a non-crossing geometric graph $G=(V,E)$ and an edge $\overline{xy} \in E$, an *edge rotation* of $\overline{xy}$ around x replaces $\overline{xy}$ by an edge $\overline{xw} \notin E$, if the open triangle $\triangle xyw$ does not intersect any edge $e \in E$ nor does it contain any vertex of V.

Figure 1 shows an edge rotation for a geometric graph (a). Variants of the definition of edge rotations are possible, for example neglecting the conditions imposed on the triangle $\triangle xyw$, Case (b) in Figure 1. A more restrictive rotation allows us to replace $\overline{xy}$ by $\overline{xw}$ only if y and w are consecutive vertices in the cyclic order of visible (as seen from x) vertices around x, see Case (c). An edge flip is shown in (d). We will use the definition of Case (a); the same arguments can be applied to show connectivity of the edge rotation graph for (b) and (c).

Edge rotations are a well known operation for graphs, see [5] for example. In the geometric setting, edge rotations have been considered for the class of trees by Bose, Czyzowicz and Hurtado, see [3]. A transformation for trees are edge-slides, studied by Aichholzer and Reinhardt [1]. We refer to the survey [3] for more details and related works.

We are concerned with the question whether and how fast two geometric graphs can be transformed into each other by means of edge rotations. In terms of graph theory, we are asking for connectivity and diameter of the edge rotation graph.

Let V be a set of points in the plane in general position, and let $PG(V,k)$ be the set of all non-crossing geometric graphs on V with k edges. The *edge rotation graph* $\mathcal{ERG}(V,k)$ has vertex set $PG(V,k)$ and two vertices are joined by an edge if and only if they differ by an edge rotation.

To make things easier to understand, in what follows, we will augment a non-crossing geometric graph to a triangulation by adding extra edges which we call *absent edges*. These additional edges do not influence edge rotations in geometric graphs. In what follows, we will also assume that the geometric graphs in $PG(V, k)$ are not triangulations, and thus have at least one absent edge. Hence, k can be any value in the range between 1 and the number of edges of a triangulation on V minus 1.

This work is organized as follows. In Section 2 we prove that the edge rotation graph is connected and give tight asymptotic bounds on its diameter. In Sections 3 and 4 we show that this also holds for non-crossing geometric graphs with labeled edges and for directed non-crossing geometric graphs. Some proofs are omitted or not given in full detail; we refer to [4] for a more detailed study on edge rotations.

2 Connectivity

The following two lemmas will be the main tools for proving connectivity.

Let T be a triangulation, and let a and e be two edges of T.

Lemma 1 *There is a sequence of edge rotations that transform $T - a$ to $T - e$.*

Proof. Let T' be the dual graph of T and let P be a shortest path in T' between a face of T containing a and a face containing edge e. The proof proceeds by induction over the path length of P.

Base case: The path length is zero. Hence, a and e are edges of a triangle of T. By definition, we can rotate the edge e to the position of a using the vertex incident to both edges as pivot.

Inductive case: The path length is $\ell > 0$. The path P induces a strip of triangles in T connecting a with e. Consider the edge c of T incident to the triangle that contains a and the next triangle in the P. Since a and c are in the same triangle, we can exchange the positions of a and c. This gives a shorter path P' whose length is $\ell-1$. By induction hypothesis we can move the edge a to the edge e. □

In other words, Lemma 1 tells us the following: Let T be a triangulation with an absent edge a. Then a can be exchanged with any other edge e of T by a sequence of edge rotations. Clearly, this also holds for triangulations with more absent edges.

Lemma 2 *Given a triangulation with at least one absent edge, any edge flip can be simulated with edge rotations.*

Proof. Recall that an edge flip replaces a diagonal of a convex quadrilateral by the other one, see Figure 1 (d). Note first that a diagonal d (that is flippable) that is an absent edge can simply be replaced by the other diagonal, because the absent edge is not part of the geometric graph. Assume that d is not an absent edge. We can simulate the edge flip of d by a sequence of edge rotations:

- Using Lemma 1 move an absent edge a to the boundary of the convex quadrilateral; call its vertices p, q, r, s in cyclic order. Say $a = \overline{pq}$ and $d = \overline{pr}$.
- Rotate d to the position of a.
- Replace the (absent) diagonal $a' = \overline{pr}$ by the (absent) diagonal $a'' = \overline{qs}$.
- Rotate the edge $d' = \overline{pq}$ to $d'' = \overline{qs}$.
- Finally, the absent edge can be moved back to its original position by reversing the sequence of edge rotations from Lemma 1.

□

We now prove:

Theorem 3 *$\mathcal{ERG}(V, k)$ is connected.*

Proof. Recall that we are assuming that the graphs in $PG(V, k)$ are not triangulations, for otherwise no edge rotation is possible. We now show that given two plane straight line graphs G_1 and G_2 in $PG(V, k)$, we can convert G_1 into G_2 using edge rotations.

First complete G_1 and G_2 to triangulations T_1 and T_2 by adding absent edges. Given any two triangulations on the same point set, it is always possible to convert one triangulation into the other one using edge flips [6, 7]. By Lemma 1 we can move an absent edge to any place in the triangulation. By Lemma 2, we can simulate an edge flip with edge rotations. Therefore, simulating a sequence of edge flips, T_1 can be transformed into T_2 using edge rotations: move an absent edge to the quadrilateral where we want to do a flip; simulate this edge flip with rotations; and move an absent edge to the next quadrilateral where we want to perform a flip. Once T_1 is transformed into T_2, we can reallocate the absent edges in T_2 by Lemma 1, such that we are left with G_2 when the absent edges are ignored. □

Corollary 4 *Under edge rotations of type (b) and (c), $\mathcal{ERG}(V, k)$ remains connected.*

Proof. For edge rotations of type (b) we only need to observe that an edge rotation of type (a) is also of type (b). For edge rotations of type (c) we only need to observe that an edge rotation of type (a) from $\overline{xy}$

to $\overline{xw}$ can be replaced by a sequence of edge rotations of type (c), visiting all visible vertices (as seen from x) from y to w in the cyclic order around x. □

2.1 Diameter bounds

In this section we show tight asymptotic bounds on the diameter of the edge rotation graph.

Theorem 5 *The diameter of $\mathcal{ERG}(V,k)$ has an upper bound of $O(|V|^2)$.*

Proof. The procedure described in Theorem 3 tells us how to transform an arbitrary plane straight line graph into any other one. Essentially, we simulate edge flips by edge rotations. Any triangulation on a point set V can be transformed into the Delaunay triangulation by applying at most $O(|V|^2)$ edge flips [7, 8]. This gives an upper bound of $O(|V|^3)$ for the diameter of $\mathcal{ERG}(V,k)$. We now show how to reduce this to $O(|V|^2)$.

When considering edge rotations, only a constant number of rotations is necessary to simulate an edge flip if the considered quadrilateral contains an absent edge. In general this is not the case, and one might need to do a linear number of edge rotations to simulate a flip.

An amortized counting will show that at most a linear number of times an absent edge has to be moved along more than a constant number of faces. We exploit the local nature of the propagation of flips in the Delaunay edge flip algorithm. Recall that an edge e of a triangulation can be flipped by the Delaunay flip algorithm if the circumcircle of a triangle incident to e contains the opposite corner of the other triangle incident to e in its interior.

We simulate a Delaunay edge flip with edge rotations. Initially we may have to perform a number k of edge flips to simulate such a Delaunay flip. Consider the edges of the quadrilateral containing the flipped edge. If for at least one of them, the circumcircle of a triangle containing this edge contains a fourth point of V, then the absent edge only needs to be moved along a constant number of faces to simulate the next Delaunay flip. Otherwise, if a circumcircle of three points does not contain any point of V in its interior then the triangle Δ formed by these three points is already *a triangle of the Delaunay triangulation*. Charge the initial k edge rotations to Δ, and restart the counting argument. Since any triangulation contains only $O(|V|)$ triangles, this situation happens at most $O(|V|)$ times. Since k is $O(|V|)$, it follows that $O(|V|^2)$ edge rotations are sufficient. □

We remark that for edge rotations of type (c) this arguments yield a bound for the diameter of $O(|V|^3)$.

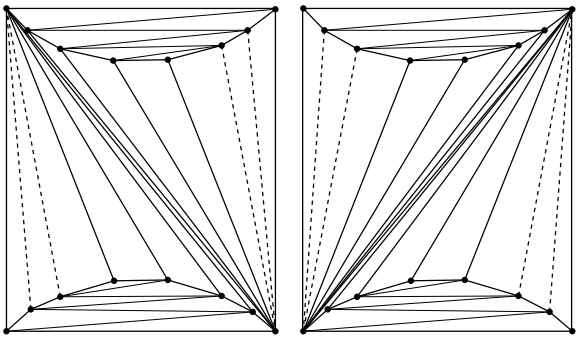


Figure 2: An example that requires $\Omega(|V|^2)$ edge rotations.

Theorem 6 *There exist point sets V such that the diameter of $\mathcal{ERG}(V,k)$ is $\Omega(|V|^2)$, for k equal the number of edges of a triangulation on V minus a constant.*

Proof. We look at the example for the $\Omega(|V|^2)$ bound of the edge flip graph from [6], see Figure 2. Now, for edge rotations, we remove a constant number of edges from the left and from the right in the example, that is, the dashed edges in Figure 2. One can verify that the same arguments as for edge flips from [6] can be applied to edge rotations here. We omit the details. □

3 Labeled graphs

In this section we consider plane geometric graphs where the edges are labeled; each edge has a different label. Let V be a set of points on the plane in general position, and let $PG_L(V,k)$ be the set of all non-crossing geometric graphs on V with k edges labeled $e_1,\ldots,e_k$. Clearly, $|PG_L(V,k)| = k!\,|PG(V,k)|$.

The *labeled edge rotation graph* $\mathcal{ERG}_{\mathcal{L}}(V,k)$ is the graph whose vertex set is $PG_L(V,k)$ and two vertices are joined by an edge if and only if they differ by an edge rotation, see Figure 3.

Lemma 7 *Let G be a graph in $PG_L(V,k)$, and e_i and e_j two edges in G. Then the labels of e_i and e_j*

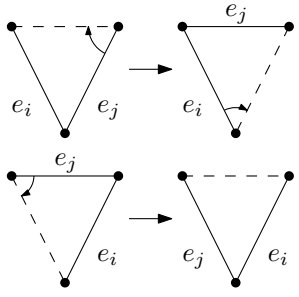


Figure 3: Exchanging labels using edge rotations.

can be exchanged, leaving the labels of the remaining edges in G unchanged.

Proof. Let e_i and e_j denote the labeled edges to be interchanged. As in the proof of Lemma 1 we consider the dual graph of the triangulation (obtained from G by adding absent edges) and find a path between the edge e_i and the edge e_j. We prove the lemma using induction over the length of a shortest path (in the dual graph) between them.

Base case $\ell = 0$: The case when e_i and e_j are edges of the same triangle Δ, and it has an absent edge is easy to check. If Δ containing e_i and e_j has no absent edge, we move an absent edge to the position of the third edge of Δ (using Lemma 1). After e_i and e_j are interchanged, we reverse the sequence of moves of the absent edge to its original position. This guarantees that the structure of the triangulation stays the same and only the labels of e_i and e_j have been exchanged.

Inductive case: If the path between e_i and e_j has length $\ell > 0$, we first exchange the label of e_i with that of the next edge, say e_m of the triangle strip induced by the path between e_i and e_j. We can do this without modifying the triangulation and the label assignment using the base case. Then we have a shorter path between e_i and e_j and by induction, we can exchange the labels of e_i and e_j. We then exchange the labels of e_j and e_m. □

Theorem 8 *$\mathcal{ERG}_{\mathcal{L}}(V,k)$ is connected.*

Proof. Suppose we have two labeled graphs G and H in $\mathcal{ERG}_{\mathcal{L}}(V,k)$. By Theorem 3, ignoring the labels on the edges of H and G, transform G to H. At this point the labels on the edges of H are permuted. Now using Lemma 7, we move each labeled edge to its final position. □

We remark that when considering *edge flips* in labeled triangulations, the *labeled edge flip graph* is not necessarily connected any more, as can be seen by assigning labels to the graph shown in Figure 2.

4 Directed graphs

In this section we consider plane geometric graphs with oriented edges. Let V be a set of points in the plane in general position, and let $PG_D(V,k)$ be the set of all non-crossing geometric graphs on V with k oriented edges. Clearly, $|PG_D(V,k)| = 2^k\ |PG(V,k)|$. The *directed edge rotation graph* $\mathcal{ERG}_{\mathcal{D}}(V,k)$ is the graph whose vertex set is $PG_D(V,k)$ and two vertices are joined by an edge if and only if they differ by an edge rotation.

Theorem 9 *$\mathcal{ERG}_{\mathcal{D}}(V,k)$ is connected.*

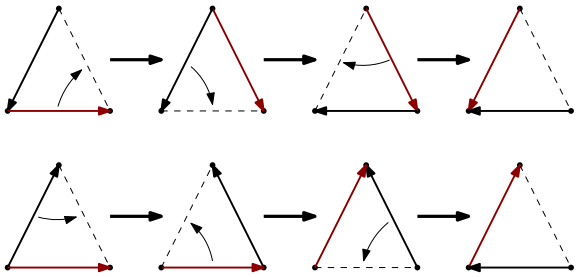


Figure 4: Two cases for inverting the direction of an edge with a sequence of edge rotations.

The theorem can be proved similar to the proceeding section. We omit the proof and only show here two elementary cases of reorienting an edge inside a triangle with an absent edge, see Figure 4.

5 Concluding remarks

We have proved that transforming geometric graphs with edge rotations has, in general, the same complexity as transformations using edge flips. Our proofs strongly rely on the connectivity of the edge flip graph. It would be of interest to study problems related to the hamiltonicity of edge rotation graphs. Some results on hamiltonicity of these graphs are given in [4]. For instance, it is straightforward to prove that $\mathcal{ERG}(V,k)$ is hamiltonian for $k = 1, 2$ and edge rotations of type (c) but it is not if we restrict the rotation angle to be less than π.

References

[1] O. Aichholzer and K. Reinhardt. A quadratic distance bound on sliding between crossing-free spanning trees. *Computational Geometry Theory and Applications*, 37(3):155–161, 2007.

[2] D. Avis and K. Fukuda. Reverse Search for Enumeration. *Discrete Applied Mathematics*, 65:21–46, 1996.

[3] P. Bose and F. Hurtado. Flips in planar graphs. *Computational Geometry Theory and Applications*, 42(1):60–80, 2009.

[4] J. Espinosa Longi, *La gráfica de rotaciones de aristas en gráficas geométricas planas*, Master Thesis, Universidad Nacional Autónoma de México, 2009.

[5] W. Goddard and H. C. Swart. Distances between graphs under edge operations. *Discrete Mathematics*, 161:121–132, 1996.

[6] F. Hurtado, M. Noy, and J. Urrutia. Flipping edges on triangulations. *Discrete and Computational Geometry*, 22:333–346, 1999.

[7] C. L. Lawson. Transforming triangulations. *Discrete Mathematics*, 3:365-372, 1972.

[8] C. L. Lawson. Software for C^1 surface interpolation. In *Mathematical Software III*, 161–194, 1977.

Delaunay Triangulations of Point Sets in Closed Euclidean d-Manifolds

Manuel Caroli* Monique Teillaud*

Abstract

We give a definition for Delaunay triangulations of point sets in closed Euclidean d-manifolds and describe an algorithmic test to check whether the Delaunay triangulation in a closed Euclidean d-manifold exists for a given point set. We provide an algorithm to compute the Delaunay triangulation if it exists. Otherwise the algorithm returns the Delaunay triangulation in a finitely sheeted covering space.

1 Introduction

The Delaunay triangulation of a point set in $\mathbb{E}^d$ is a well-studied structure in computational geometry. Efficient algorithms are known and there exist various implementations. We extend the definition of the Delaunay triangulation of a point set to closed Euclidean d-manifolds and give an algorithm to compute it.

This abstract is a generalization of [4], which discusses the case of the three-dimensional flat torus $\mathbb{T}^3$. We start with a short summary of it in Section 2. In Section 3, we introduce closed Euclidean d-manifolds and their properties. Finally, in Section 4, we present the generalization of [4] to closed Euclidean d-manifolds and describe briefly the algorithm.

2 Review of Triangulations in $\mathbb{T}^3$

Let $\mathbb{E}^3$ denote the three-dimensional Euclidean space, and $\mathcal{P}$ be a finite point set in $\mathbb{E}^3$. The three-dimensional flat torus $\mathbb{T}^3$ is defined as the quotient space $\mathbb{E}^3/\mathcal{G}$ with $\mathcal{G} = (\mathbb{Z}^3, +)$. Let $\pi : \mathbb{E}^3 \to \mathbb{T}^3$ denote the quotient map and $DT(\mathcal{GP})$ denote the Delaunay triangulation of $\mathcal{GP} = \{p + z \mid p \in \mathcal{P}, z \in \mathbb{Z}^3\}$ in $\mathbb{E}^3$.

We now give a summary of [4].

Definition 1 *If $\pi(DT(\mathcal{GP}))$ is a simplicial complex in $\mathbb{T}^3$, then we call it the* Delaunay triangulation of $\pi(\mathcal{P})$ in $\mathbb{T}^3$.

For example, if $\mathcal{P}$ consists of only one point, then $\pi(DT(\mathcal{GP}))$ is not a simplicial complex.

Theorem 1 *If the 1-skeleton[1] of $\pi(DT(\mathcal{GP}))$ does not contain cycles of length ≤ 2, then $\pi(DT(\mathcal{GP}))$ is a triangulation in $\mathbb{T}^3$.*

This yields a geometric criterion for $\pi(DT(\mathcal{GP}))$ to be a triangulation.

Corollary 2 *Let $\mathbb{B}$ denote the largest 3-ball in $\mathbb{E}^3$ that does not contain points of $\mathcal{GP}$ in its interior. If $\mathbb{B}$ has diameter $< \frac{1}{2}$, then $\pi(DT(\mathcal{GP}'))$ is a triangulation in $\mathbb{T}^3$ for any finite $\mathcal{P}' \supseteq \mathcal{P}$.*

Note that the geometric criterion also holds for supersets of $\mathcal{P}$, which will be useful for the algorithm later on. Consider the quotient space $\mathbb{T}^3_{27} := \mathbb{E}^3/\mathcal{G}_{27}$ with $\mathcal{G}_{27} := ((3\mathbb{Z})^3, +)$. Then $\mathbb{T}^3_{27}$ is a 27-sheeted covering space of $\mathbb{T}^3$.[2]

Theorem 3 *For any finite point set $\mathcal{P}$ in $\mathbb{E}^3$, the projection of the Delaunay triangulation of $\mathcal{GP}$ in $\mathbb{E}^3$ onto $\mathbb{T}^3_{27}$ is a triangulation.*

Theorem 3 and Corollary 2 lead to a modified version of the incremental algorithm for computing Delaunay triangulations in $\mathbb{E}^3$ [3]: It starts with inserting 27 copies per input point, computing their Delaunay triangulation in $\mathbb{T}^3_{27}$. Once the largest 3-ball not containing any vertex in its interior has diameter smaller than $\frac{1}{2}$, the algorithm switches to computing in $\mathbb{T}^3$ and inserts each of the remaining points only once.

While computing in $\mathbb{T}^3_{27}$, 27 copies of points of $\mathcal{P}$ are inserted one by one. So actually the following extended version of Theorem 3 is needed:

Theorem 4 *Theorem 3 still holds if we replace $\mathcal{GP}$ by $\mathcal{GP} \cup \mathcal{G}_{27}Q$ for any $Q \subseteq \mathcal{G}p$ with any $p \in \mathbb{E}^3$.*

3 Closed Euclidean manifolds

This section is dedicated to introducing closed Euclidean manifolds, their properties, and how to construct them. Most concepts mentioned in this section are taken from [7].

A *closed manifold* is a compact manifold without boundary. A d-manifold is called *Euclidean* or *flat*, if every point has a neighborhood isometric to a neighborhood in $\mathbb{E}^d$.

We need some more notions: Let $\mathcal{G}$ be a group and $\mathcal{H}$ denote a subgroup of $\mathcal{G}$. $\mathcal{H}$ is called *normal* in $\mathcal{G}$ if it is invariant under conjugation, i.e., if for all $h \in \mathcal{H}$ and $g \in \mathcal{G}$, $ghg^{-1} \in \mathcal{H}$. Note that a normal subgroup $\mathcal{H}'$ of $\mathcal{H}$ is again normal in $\mathcal{G}$. For a group element

*INRIA Sophia Antipolis – Méditerranée, {Manuel.Caroli,Monique.Teillaud}@sophia.inria.fr

[1] The graph of all vertices and edges

[2] See e.g. [1] for a discussion on covering spaces.

$g \in \mathcal{G}$ the set $\{gh \mid h \in \mathcal{H}\}$ is called a *coset* of $\mathcal{H}$ in $\mathcal{G}$. The *index of a subgroup* $\mathcal{H}$ in $\mathcal{G}$ is defined as the number of cosets of $\mathcal{H}$ in $\mathcal{G}$.

A d-dimensional *Bieberbach group* $\mathcal{G}_B$ is a discrete group of isometries of $\mathbb{E}^d$ with compact quotient space $\mathbb{E}^d/\mathcal{G}_B$. Such groups are also called *crystallographic groups* or *space groups*.

Theorem 5 (Bieberbach [5]) *Let $\mathcal{G}_B$ be a d-dimensional Bieberbach group. Then*

- *There is a group $\mathcal{G}_T$ of d linearly independent translations that is a normal subgroup of $\mathcal{G}_B$ of finite index. We call $\mathcal{G}_T$ the translational subgroup of $\mathcal{G}_B$.*
- *For any d there is only a finite number of d-dimensional Bieberbach groups.*

Note that the quotient space $\mathbb{E}^d/\mathcal{G}_B$ is not necessarily a manifold: If $\mathcal{G}_B$ leaves points fixed, these points do not have a neighborhood in $\mathbb{E}^d/\mathcal{G}_B$ that is homeomorphic to a neighborhood in $\mathbb{E}^d$. The quotient space $\mathbb{E}^d/\mathcal{G}_B$ can always be described by the more general concept of an *orbifold* [2, 8]. For the quotient space to be a manifold, the group must not have fixed points. In other words the group must be *torsion-free*, i.e., the identity must be the only element of finite order.

If $\mathcal{G}_T$ is a subgroup of d independent translations of $\mathcal{G}_B$, then $\mathbb{E}^d/\mathcal{G}_T$ is a d-torus.

Theorem 6 ([7]) *Any closed Euclidean d-manifold corresponds up to diffeomorphism to exactly one quotient space $\mathbb{E}^d/\mathcal{G}_B$, where $\mathcal{G}_B$ is a torsion-free d-dimensional Bieberbach group.*

This means that it is sufficient to consider torsion-free Bieberbach groups to completely classify closed Euclidean manifolds.

According to Theorem 5, there are only finitely many d-dimensional Bieberbach groups. In dimension 2 there are 17, in dimension 3 there are 230.[3] In two dimensions there are only two torsion-free Bieberbach groups and thus two closed Euclidean manifolds: the torus and the Klein bottle. In three dimensions there are 10 closed Euclidean manifolds, four of which are non-orientable.

4 Triangulations in closed Euclidean manifolds

The goal of this section is to generalize the main results of [4] given in Section 2.

Let $\mathcal{G}_F$ be a torsion-free d-dimensional Bieberbach group, $\mathcal{P}$ a finite point set in $\mathbb{E}^d$, $\mathbb{X} := \mathbb{E}^d/\mathcal{G}_F$ a closed Euclidean manifold with quotient map $\pi : \mathbb{E}^d \to \mathbb{X}$, and $DT(\mathcal{G}_F\mathcal{P})$ the Delaunay triangulation of $\mathcal{G}_F\mathcal{P}$ in $\mathbb{E}^d$. We first adapt Definition 1 to the Delaunay triangulation of $\pi(\mathcal{P})$ in $\mathbb{X}$:

Definition 2 *If $\pi(DT(\mathcal{G}_F\mathcal{P}))$ is a simplicial complex in $\mathbb{X}$, then we call it the* Delaunay triangulation of $\pi(\mathcal{P})$ in $\mathbb{X}$.

For the discussions below we need the following two values:

1. The minimum distance $\delta(\mathcal{G})$ by which a group $\mathcal{G}$ moves a point:
$$\delta(\mathcal{G}) = \min_{p \in \mathbb{E}^d, g \in \mathcal{G}, g \neq 1_\mathcal{G}} \operatorname{dist}(p, gp),$$
where $1_\mathcal{G}$ denotes the unit element of $\mathcal{G}$. Note that if $\mathcal{G}$ is torsion-free and discrete, then $\delta(\mathcal{G}) > 0$ holds.
2. The diameter $\Delta(\mathcal{S})$ of the largest d-ball $\mathbb{B}$ in $\mathbb{E}^d$ that does not contain any point of a set $\mathcal{S}$ in its interior.

We now generalize Theorem 1:

Theorem 7 *If the 1-skeleton of $\pi(DT(\mathcal{G}_F\mathcal{P}))$ does not contain cycles of length ≤ 2 then $\pi(DT(\mathcal{G}_F\mathcal{P}))$ is a triangulation in $\mathbb{X}$.*

Most parts of the proof of Theorem 1 are completely combinatoric and do not depend on the space, so they extend directly to $\mathbb{X}$ and we omit them in this abstract. We only prove the generalized version of Lemma 4.1 of [4]:

Lemma 8 *Let $\mathcal{K}$ be a set of simplices in $\mathbb{E}^d$ with the following properties:*

(i) The vertices of $\mathcal{K}$ are exactly the elements of $\mathcal{G}_F\mathcal{P}$.
(ii) If $\sigma \in \mathcal{K}$ and τ is a face of σ, then $\tau \in \mathcal{K}$.
(iii) If $\sigma, \tau \in \mathcal{K}$, then $\sigma \cap \tau \in \mathcal{K}$.
(iv) If $\sigma \in \mathcal{K}$, then there is a d-ball circumscribing σ that does not contain any other point of $\mathcal{G}_F\mathcal{P}$ in its interior.

Then $\mathcal{K}$ is locally finite, i.e., each point p in $\mathbb{E}^d$ has a neighborhood $U(p)$ such that the number of elements in $\{U(p) \cap \sigma \mid \sigma \in \mathcal{K}\}$ is finite.

Proof. Assume there is a vertex $v \in \mathcal{K}$ with an infinite number of incident simplices and thus an infinite number of incident edges in $\mathcal{K}$. Since $\mathcal{P}$ contains only a finite number of points, there must be at least one point q in $\mathcal{P}$ such that infinitely many points of the discrete point set $\mathcal{G}_F q$ are adjacent to v. Note that $\delta(\mathcal{G}_F) > 0$ and $\Delta(\mathcal{G}_F q) < \infty$ hold because $\mathcal{G}_F$ is a torsion-free Bieberbach group. Projecting all the edges from v to points in $\mathcal{G}_F q$ onto the unit d-sphere S centered in v yields an infinite point

[3] The number of Bieberbach groups by dimension is assigned the id A006227 in the On-Line Encyclopedia of Integer Sequences [6]. The number of *torsion-free* Bieberbach groups is assigned the id A059104.

set $\mathcal{P}_S$. As S is bounded, $\mathcal{P}_S$ must have an accumulation point. We choose q_1 and q_2 from $\mathcal{G}_F q$ such that the distance between their projections onto S is smaller than ε for some $\varepsilon < \frac{\delta(\mathcal{G}_F)^3}{\Delta(\mathcal{G}_F q)^3}$ and $\varepsilon > 0$. W.l.o.g. we assume $\text{dist}(v, q_2) \geq \text{dist}(v, q_1)$. We give a lower bound on the diameter D of the circumcircle of the triangle vq_1q_2 (see Fig. 1): D is given by the product of the three edge lengths divided by twice the triangle's area. The three edge lengths are each at least $\delta(\mathcal{G}_F)$. Also $\text{dist}(v, q_2) \leq \Delta(\mathcal{G}_F q)$ and the height of the triangle corresponding to the segment vq_2 is at most $\Delta(\mathcal{G}_F q)\varepsilon$. This yields $D \geq \frac{\delta(\mathcal{G}_F)^3}{\Delta(\mathcal{G}_F q)^2 \varepsilon} > \frac{\delta(\mathcal{G}_F)^3}{\Delta(\mathcal{G}_F q)^2 \frac{\delta(\mathcal{G}_F)^3}{\Delta(\mathcal{G}_F q)^3}} = \Delta(\mathcal{G}_F q)$. So the d-ball B_{vq_2} with v and q_2 on its boundary must have diameter larger than $\Delta(\mathcal{G}_F q)$ to not contain q_1 in its interior. As the largest empty d-ball has diameter $\Delta(\mathcal{G}_F q)$, B_{vq_2} cannot be empty, which is a contradiction to condition *(iv)*.

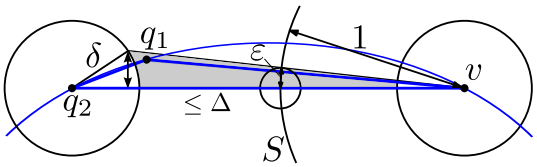


Figure 1: The gray area shows the possible positions of q_1

Let us now consider a point p in $\mathbb{E}^d$ that is not a vertex in $\mathcal{K}$. Let σ denote the simplex that contains p in its interior and let v_σ denote a vertex of σ. Let $St(v_\sigma)$ denote the set of simplices that v_σ is incident to. Above we have shown that $St(v_\sigma)$ contains only finitely many elements. The set $St(\sigma)$ of simplices that σ is incident to is a subset of $St(v_\sigma)$ and thus finite. There is a neighborhood $U(p)$ that has non-empty intersection with exactly the elements $St(\sigma)$. □

This concludes the proof of Theorem 7.

Corollary 2 mentions the threshold $\frac{1}{2}$, which depends on the group $\mathcal{G}$. The generalized version of this Corollary follows by simple geometric reasoning from Theorem 7.

Corollary 9 *If $\Delta(\mathcal{G}_F\mathcal{P}) < \frac{\delta(\mathcal{G}_F)}{2}$, then $\pi(DT(\mathcal{G}_F\mathcal{P}'))$ is a triangulation in $\mathbb{X}$ for any finite $\mathcal{P}' \supseteq \mathcal{P}$.*

For any torsion-free Bieberbach group there are point sets such that the condition of Corollary 9 is fulfilled because δ is strictly positive and Δ can be made arbitrarily small by the choice of the point set.

Finally, we give a generalized version of Theorem 4.

Theorem 10 *There is a normal subgroup $\mathcal{G}_C$ of $\mathcal{G}_F$ of finite index such that the projection of the Delaunay triangulation of $\mathcal{G}_F\mathcal{P} \cup \mathcal{G}_C Q$ in $\mathbb{E}^d$ onto $\mathbb{X}_C = \mathbb{E}^d/\mathcal{G}_C$ is a triangulation for any finite point set $\mathcal{P}$ in $\mathbb{E}^d$ and any $Q \subseteq \mathcal{G}_F p$ with any $p \in \mathbb{E}^d$.*

Proof. According to Theorem 5, there is a group $\mathcal{G}_T$ of d linearly independent translations that is a normal subgroup of $\mathcal{G}_F$ with finite index h'. We choose generators $g_1, \ldots, g_d$ of $\mathcal{G}_T$ in the following way: Let g_1 be the shortest translation in $\mathcal{G}_T$. Let g_{i+1} be the shortest translation in $\mathcal{G}_T$ that is linearly independent of the translations $g_1, \ldots, g_i$. Note that $\Delta(\mathcal{G}_T p)$ does not depend on a specific choice of p and thus can be considered constant. For each g_i we can find an integer coefficient c_i such that $\text{dist}(q, g^{c_i} q) > 2 \cdot \Delta(\mathcal{G}_T p)$ for any $q \in \mathbb{E}^d$. The group $\mathcal{G}_C$ generated by $g_1^{c_1}, \ldots, g_d^{c_d}$ is a normal[4] subgroup of $\mathcal{G}_T$ of index $c_1 \cdot \ldots \cdot c_d$ with the property $\text{dist}(q, gq) > 2\Delta(\mathcal{G}_F p)$ for any $g \in \mathcal{G}_T$ and any $q \in \mathbb{E}^d$. Followingly, $\mathcal{G}_C$ is a normal subgroup of $\mathcal{G}_F$ with index $h = h' \cdot c_1 \cdot \ldots \cdot c_d$. According to Corollary 9 the projection of the Delaunay triangulation of $\mathcal{G}_F\mathcal{P}$ onto $\mathbb{X}_C$ forms a triangulation. By adding further points the diameter of the largest empty ball cannot grow. Thus the claim holds. □

Theorem 10 means that there exists a space $\mathbb{X}_C$, in which the Delaunay triangulation of the point set $\pi(\mathcal{P})$ is defined. The space $\mathbb{X}_C$ is a covering space of $\mathbb{X}$ with a finite number of sheets [1]. Theorem 10 can also be understood by constructing $\mathbb{X}_C$ from $\mathbb{X}$ directly, as follows.

Definition 3 *A* fundamental domain *for a discrete group $\mathcal{G}$ of isometries in $\mathbb{E}^d$ with quotient map $\pi : \mathbb{E}^d \to \mathbb{E}^d/\mathcal{G}$ is a closed and convex subset $D_\mathcal{G}$ of $\mathbb{E}^d$ such that*

- *$D_\mathcal{G}$ contains at least one point of the preimage by π of any point in $\mathbb{E}^d/\mathcal{G}$.*
- *If $D_\mathcal{G}$ contains more than one point of the same preimage, then all points of this preimage lie on the boundary of $D_\mathcal{G}$.*

For example the unit cube is a fundamental domain of $\mathbb{T}^3$ as defined in Section 2.

Each closed Euclidean d-manifold has a d-torus as covering space with a finite number of sheets. This follows from Theorem 5 as discussed above. A fundamental domain of the d-torus is a d-dimensional hyperparallelepiped. By glueing two of these hyperparallelepipeds together we get a new covering space that is again a d-torus. We can construct $\mathbb{X}_C$ by glueing as many copies of the fundamental domain as necessary to fulfill the condition in Corollary 9, i.e., $\Delta(\mathcal{G}_C\mathcal{G}_F\mathcal{P}) = \Delta(\mathcal{G}_F\mathcal{P}) < \frac{\delta(\mathcal{G}_C)}{2}$. See Figure 2 for an illustration in two dimensions.

As an example we consider the flat Klein bottle. The Klein bottle is $\mathbb{E}^2/\mathcal{G}_K$, where $\mathcal{G}_K$ is the group

[4]Every subgroup of an abelian group is normal.

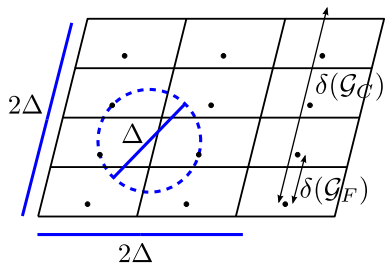


Figure 2: Sufficient number of copies of the fundamental domain

generated by a translation g_t and a glide-reflection g_g, that is a reflection together with a translation perpendicular to the reflection (see Figure 3). The group generated by g_t and g_g^2 is a translational subgroup of $\mathcal{G}_K$ of index 2. Now we can choose a subgroup of this translational subgroup with finite index that fulfills the condition of Theorem 10 as in Figure 2.

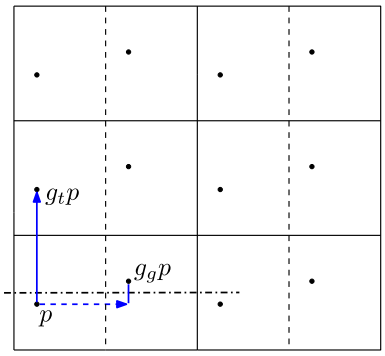


Figure 3: A part of the infinite point grid $\mathcal{G}_K p$.

Note that in both Corollary 9 and Theorem 10 we deal with the condition of the form $\Delta < \frac{\delta}{2}$. In Corollary 9 we adapt the point set $\mathcal{P}$ to decrease Δ, in Theorem 10 we adapt the group $\mathcal{G}_C$ to increase δ.

Algorithm

The algorithm described at the end of Section 2 generalizes to $\mathbb{X} = \mathbb{E}^d/\mathcal{G}_F$ using the results given in this section. It starts computing in an h-sheeted covering space $\mathbb{E}^d/\mathcal{G}_C$ as in Theorem 10, inserting h copies per input point. Once the condition of Corollary 9 is met for the current point set, it switches to computing in $\mathbb{X}$ and continues to insert each of the remaining points only once.

There appear two issues, namely, how to store the current triangulation and how to insert a point.

The triangulation can be stored as a graph in the following way: Fully-dimensional simplices are stored with a list of their vertices and neighbors. Each vertex contains the coordinates of the point it corresponds to. Additionally, each d-cell stores the information on how to map it isometrically into $\mathbb{E}^d$, i.e., one element of the cell's preimage under the quotient map π.

The point insertion can be split in a combinatorial and a geometric part. The combinatorial part does not need to be adapted. The geometric part mainly consists of computing the orientation of $d+1$ points or computing whether a point is situated inside or outside a d-ball given by $d+1$ points on its boundary. Both of these so-called predicates are evaluated in $\mathbb{E}^d$: For each d-cell, on which we need to evaluate a predicate, we take its preimage under π from the data structure and evaluate the predicate on this preimage. This works exactly the same way even if $\mathbb{X}$ is non-orientable: The orientation of a preimage under π can be computed using the orientation predicate in $\mathbb{E}^d$ because $\mathbb{E}^d$ is oriented.

5 Conclusion

We extended the work of [4] to arbitrary dimensions and to any closed Euclidean manifold. The approach does not extend to general orbifolds because it cannot handle Bieberbach groups with fixed points. However, from the Bieberbach theorem we know that any such orbifold has a finitely sheeted covering space that is a closed Euclidean manifold and on which our approach works.

References

[1] M. A. Armstrong. *Basic Topology.* Springer-Verlag, 1982.

[2] M. Boileau, S. Maillot, and J. Porti. *Three-dimensional orbifolds and their geometric structures.* Société mathémathique de France, Paris, 2003.

[3] A. Bowyer. *Computing Dirichlet tessellations.* The Computer Journal, 24:162–166, 1982.

[4] M. Caroli and M. Teillaud. *Computing 3D Periodic Triangulations.* Proceedings of the 17th Annual European Symposium on Algorithms (ESA). Springer-Verlag, LNCS 5757, pages 59–70, 2009. Full version: Research Report 6823, `http://hal.inria.fr/inria-00405478/en`, INRIA, 2009.

[5] L. Bieberbach. *Über die Bewegungsgruppen des n-dimensionalen euklidischen Raumes mit einem endlichen Fundamentalbereich.* Göttinger Nachrichten, 75, 1910.

[6] N. J. A. Sloane. *The On-Line Encyclopedia of Integer Sequences.* `http://www.research.att.com/~njas/sequences/`.

[7] W. P. Thurston. *Three Dimensional Geometry and Topology, vol. I.* Princeton University Press, Princeton, 1997.

[8] W. P. Thurston. *The Geometry and Topology of Three-Manifolds.* Chapter 13, Electronic version 1.1, `http://www.msri.org/publications/books/gt3m`, 2002.

Certified Computation of planar Morse-Smale Complexes

A. Chattopadhyay * S.J. Holtman† G. Vegter‡

Abstract

The Morse-Smale complex is an important tool for global topological analysis in various problems of computational geometry and topology. Algorithms for Morse-Smale complexes have been presented in case of piecewise linear manifolds [3]. However, previous research in this field is incomplete in the case of smooth functions. In the current paper we use interval arithmetic to compute topologically correct Morse-Smale complex of smooth functions of two variables. The algorithm can also compute geometrically accurate Morse-Smale complex.

1 Introduction

Problem statement. A Morse function $h : \mathbb{R}^2 \to \mathbb{R}$ is a real-valued function with non-degenerate critical points (i.e., critical points with non-singular Hessian matrix). As is well-known, non-degenerate critical points are either maxima, or minima, or saddle points. We are interested in the configuration of integral curves of the *gradient vector field* ∇h of h. A *stable (unstable) separatrix* of a saddle point is the set of all regular points whose forward (backward) integral curve flows into the saddle point. (This notion will be made more precise in Section 2.) A *Morse-Smale function* is a Morse function whose stable and unstable separatrices are disjoint. In particular, the unstable separatrices flow into a sink (maximum), and the stable separatrices flow into a source (minimum). The corresponding gradient vector field will be called a *Morse-Smale system* (MS-system). A *Morse-Smale complex* (MS-complex for short) consists of all separatrices corresponding to a *MS-system*. The MS-complex describes the global structure of a Morse-Smale function. We consider the problem of computing a *certified* approximation of the MS-complex of a Morse-Smale function, i.e., a configuration of curves that is isotopic to the MS-complex. Our algorithm is based on interval arithmetic.

Our Contribution. We present an algorithm computing such a certified approximation of the MS-complex of a given smooth Morse-Smale function on the plane. In particular, the algorithm determines

- isolated certified boxes for saddles, sources and sinks.
- certified initial and terminal intervals for saddle-source or saddle-sink connectors (separatrices).
- disjoint strips around each separatrix, which can be as close to the separatrix as desired.

Related Work. Computing Morse-Smale complexes has been widely studied for piecewise-linear functions [3]. Computing MS-complexes is strongly related to vector field visualization [4]. In a similar context, designing of vector field on surfaces has been studied for many graphics applications [6]. The survey paper [2], focussing on geometrical-topological properties of real functions, gives an overview of recent work on MS-complexes.

2 Preliminaries

Morse function. A function $h : \mathcal{D} \subset \mathbb{R}^2 \to \mathbb{R}$ is called a *Morse function* if all its critical points are non-degenerate. The *Morse lemma* [3] states that near a non-degenerate critical point a it is possible to choose local co-ordinates x, y in which h is expressed as $h(x, y) = h(a) \pm x^2 \pm y^2$. The number of minus signs is called the index $i_h(a)$ of h at a. Thus a two variable Morse function has three types of non-degenerate critical points: minima (index 0), saddles (index 1) and maxima (index 2).

Integral line. An integral line $x : I \subset \mathbb{R} \to \mathcal{D}$ passing through a point p_0 on $\mathcal{D}$ is the unique maximal curve satisfying: $\dot{x}(t) = \nabla h(x(t))$, $x(0) = p_0$, for all $t \in I$. Integral lines corresponding to the gradient vector field of a smooth function $h : \mathcal{D} \to \mathbb{R}$ have many interesting properties, such as: (1) any two integral lines are either disjoint or coincide; (2) an integral line $x : I \to \mathcal{D}$ through a point p of h is injective and if $\lim_{t \to \pm\infty} x(t)$ exists, it is a critical point of h; (3) the function h is strictly increasing along the integral line of a regular point of h and integral; (4) regular integral lines are perpendicular to regular level sets of h.

Stable and unstable manifolds. Consider the integral line $x(t)$ passing through a point p. If the limit $\lim_{t \to \infty} x(t)$ exists, it is called the ω-limit of p and is denoted by $\omega(p)$. Similarly, $\lim_{t \to -\infty} x(t)$ is called the α-limit of p and is denoted by $\alpha(p)$ – again provided this limit exists. The stable manifold of a singular point p is the set $W^s(p) = \{q \in \mathcal{D} \mid \omega(q) = p\}$. Similarly, the unstable manifold of a singular point p is

*Corresponding author. Department of Mathematics and Computing Science, University of Groningen, A.Chattopadhyay@rug.nl

†Department of Mathematics and Computing Science Science, University of Groningen, s.j.holtman@rug.nl

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

the set $W^u(p) = \{q \in \mathcal{D} \mid \alpha(q) = p\}$ The stable (unstable) manifolds of a saddle point (not including the saddle point itself) are called the stable (unstable) separatrices of the saddle point. Each saddle has two stable and two unstable separatrices.

The Morse-Smale complex. A Morse function on $\mathcal{D}$ is called a Morse-Smale (MS) function if its stable and unstable separatrices are disjoint. The MS-complex associated with a MS-function h on $\mathcal{D}$ is the subdivision of $\mathcal{D}$ formed by the connected components of the intersections $W^s(p) \cap W^u(q)$, where p, q range over all singular points of h. According to quadrangle lemma [3], each region of the MS-complex is a quadrangle with vertices of index $0, 1, 2, 1$, in this order around the region.

Poincaré-Hopf Index Theory. Suppose we have a vector field over some simply connected domain $\mathcal{D}$ in the two-dimensional plane. Let Γ be any closed loop in $\mathcal{D}$ which does not pass through any fixed point of the vector field. Now, as we move around Γ in the counter-clockwise sense (which is taken as the positive direction), the vectors on Γ rotate, and when we get back to the starting-point, they will have rotated through an angle $2\pi i_\Gamma$, where i_Γ is an integer, called the *Poincaré-Hopf index* [5] (or, *index* for short) of Γ. For the gradient vector field $\nabla h \equiv (h_x, h_y)$ the index i_Γ of a closed curve Γ, is found by:

$$i_\Gamma = \frac{1}{2\pi} \oint_\Gamma d\phi = \frac{1}{2\pi} \oint_\Gamma d(\tan^{-1} \frac{h_y}{h_x}). \qquad (1)$$

The index of a critical point, say p, of a vector field, say X, is denoted by $i_X(p)$ and is defined to be the index i_Γ of a closed curve Γ which contains only the critical point p, and where no other critical points are on the closed curve. The following result is well-known in Index Theory.

Theorem 1 *(i) The index of a sink and a source is $+1$.*
(ii) The index of a saddle point is -1.
(iii) The index of a closed curve not containing any critical point is 0.
(iv) The index of a closed curve is equal to the sum of the indices of the fixed points within it.

Let p be a critical point of a Morse-function h, say with index $i_h(p)$. Then p is also critical point of the gradient vector field of h, say with Poincaré-Hopf index $i_{\nabla h}(p)$. Then $i_{\nabla h}(p) = (-1)^{i_h(p)}$.

Interval Arithmetic (IA). Interval arithmetic is used to prevent rounding errors in finite precision computations. A *range function* $\Box F$ for a function $F : \mathbb{R}^m \to \mathbb{R}^n$ computes for each m-dimensional interval I (i.e., an m-box) an n-dimensional interval $\Box F(I)$, such that $F(I) \subset \Box F(I)$. A range function is said to be *convergent* if the diameter of the output interval converges to 0 when the diameter of the input interval shrinks to 0. Convergent range functions exist for the basic operators and functions, so all range functions are assumed to be convergent.

3 Methods and Results

Computing the MS-complex of a Morse-Smale function $h : \mathcal{D} \mapsto \mathbb{R}$ reduces to computing separatrices of the corresponding gradient system. More precisely, for computing a certified MS-complex of a Morse function h over $\mathcal{D}$ we proceed as follows:

1. Compute certified intervals of the critical points and to detect their types corresponding to the MS-function.
2. Compute guaranteed one-dimensional intervals corresponding to initial points of each of the separatrices.
3. Compute certified bounds of the separatrices starting from one of these one-dimensional intervals to the correct source or sink.

3.1 Local Analysis: Isolating Critical Points.

The following subdivision algorithm isolates the critical points of a MS-function function h over a bounding box B ($\subseteq \mathcal{D}$). Moreover, the type of each critical point in the corresponding interval is also determined by index and orientation test. We consider the following assumption.

Assumption A: Given a function h we can find a positive number ϵ_c such that in any interval I (from the domain of h) of diameter less than ϵ_c, h can have at most one critical point inside I.

Algorithm. SEARCHCRITICAL(h, B)

1. Initialize a quadtree $\mathcal{T}$ to the bounding square B.
2. Subdivide $\mathcal{T}$ until for all the leaves I we have:
$$\underbrace{0 \notin \Box h_x(I)}_{(i)} \vee \underbrace{0 \notin \Box h_y(I)}_{(ii)} \vee \underbrace{diam(I) < \epsilon_c}_{(iii)}.$$
3. For each leaf I
4. Do if $\neg(i)$, $\neg(ii)$ and (iii) hold then
5. Compute $i_\Gamma :=$ index of boundary Γ of I
6. If $i_\Gamma = 0$
7. h has no critical point inside I
8. If $i_\Gamma = 1$
9. h has a source/sink inside I
10. If $i_\Gamma = -1$
11. h has a saddle inside I

Computing the index of a contour. We assume the curve Γ contains at most one critical point strictly inside it. Now, using the formula in (1) the computation of the index over the rectangular contour Γ boils down to finding all the "jump"-discontinuities of the function $\tan^{-1} \frac{h_y}{h_x}$ over Γ and adding them up. In other words, this reduces to finding isolated 1D-intervals, say J_{c_i} (Figure1), corresponding to zeros c_i of h_x parameterized over Γ such that on these intervals the sign of h_y does not change. Now depending on the change of sign of $\frac{h_y}{h_x}$ over J_{c_i}, while traversing Γ anti-clockwise sense, the contribution in the integration is $-\pi$ (when the change of sign is from negative to positive) or $+\pi$ (when the change of sign is from positive to negative).

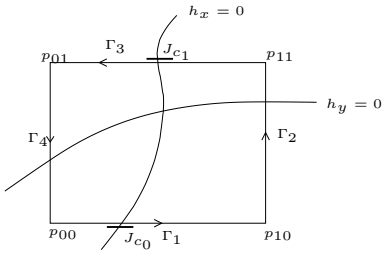


Figure 1: Index of the rectangle $\Gamma := \Gamma_1 \cup \Gamma_2 \cup \Gamma_3 \cup \Gamma_4$.

Certified Intervals for Sinks and Sources. Note that in the algorithm SEARCHCRITICAL(h, B) we use index test to distinguish a saddle from a source or a sink. However, index test cannot distinguish a source from a sink. One method to do so is by orientation test (see in Figure 3). The orientation of the gradient vector field on the boundary of the interval containing the isolated sink is inwards, whereas for a source it is outwards. Again we note that an isolated interval for a source or a sink, obtained using algorithm SEARCHCRITICAL may not be certified. Here by certified interval we mean any integral curve entering into the interval eventually meet the sink or source in forward or backward time (without leaving the box). However, the previous algorithm does not exclude the possibility of having intervals as in Figure 2. One

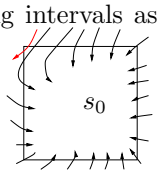


Figure 2: An isolated interval for sink s_0. The interval is not certified.

approach to find a certified interval is by subdividing the interval I recursively, until we find a subinterval such that the orientation of the gradient field on its boundary is either completely inward or completely outward (Figure 3).

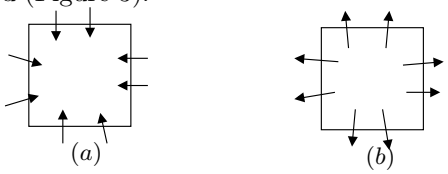


Figure 3: Orientation of the vector field on the boundary of an interval for a (a) sink, (b) source.

Local analysis of saddle intervals. Algorithm SEARCHCRITICAL(h, B) computes isolated 2D-intervals corresponding to each saddle point of h. Now we find four disjoint one-dimensional intervals on the boundary of the 2D-interval (figure 4) such that each of the four separatrices passes through one of these 1D-intervals. The method consists of following three steps.

1. First, we refine the box containing the saddle, recursively, until the function restricted over the boundary of the box has exactly four extrema (two maxima and two minima), Figure 4-(a).
2. Next we find four disjoint 1D-intervals containing four extrema, respectively Figure 4-(b).
3. The final step is to refine these four 1D-intervals, recursively, until they are guaranteed to contain the corresponding separatrix. This requires to satisfy an orientation property between two consecutive 1D-intervals, Figure 4-(c).

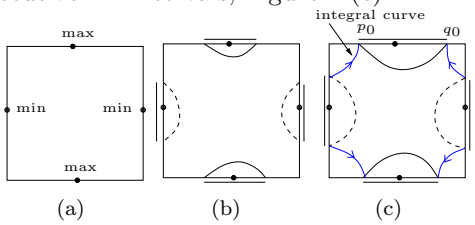


Figure 4: Saddle box: (a) refining until four extrema,(b) four initial 1D-intervals, (c) refining 1D-intervals by orientation test.

3.2 Global Algorithm: Certified Separatrices

Finally, corresponding to each separatrix we compute a certified region bounded by two piecewise linear boundaries. The piecewise linear boundaries start from the end points of the 1D-intervals near each saddle. We need the following assumption for the convergence of the algorithm.

Assumption B: ψ-normal variation. We assume that the function h satisfies *ψ-normal variation* condition, which is: for $x_1, x_2 \in \mathcal{D} \setminus \mathcal{C}$, $\|x_1 - x_2\| \leq \delta$ and $\delta > 0$,

$$\frac{\langle \nabla h(x_1), \nabla h(x_2) \rangle}{\|\nabla h(x_1)\| \|\nabla h(x_2)\|} > \cos \psi.$$

Here, $\mathcal{C}$ denotes the union of all certified intervals of the critical points of h in $\mathcal{D}$.

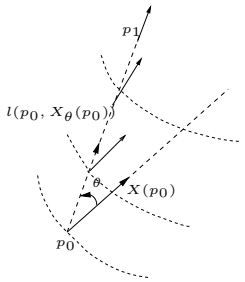


Figure 5: Line-segment $\overline{p_0 p_1}$ on which orientation and monotonicity property hold.

Lemma 2 *Let $X := \nabla h$ satisfies ψ-normal variation in $\mathcal{D} \setminus \mathcal{C}$. Again let, X_θ denote the corresponding vector field rotated anti-clockwise by an angle θ ($\psi \leq \theta < \frac{\pi}{2}$). Then for a directed line segment $l(p_0, X_\theta(p_0))$ along the direction $X_\theta(p_0)$ with starting point p_0 and length δ (segment $\overline{p_0 p_1}$, in the figure 5), the following properties hold:*

(i) for each point $q \in l(p_0, X_\theta(p_0))$, $\Delta(X(q), X_\theta(p_0)) > 0$, (here $\Delta(v_1, v_2) := \begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix}$ for $v_i \equiv (a_i, b_i)$)

(ii) h is monotonically increasing along this line segment $l(p_0, X_\theta(p_0))$.

(iii) The minimum change of the height function h on this line segment is $\delta \cos\psi$.

Algorithm: ComputeSeparatrixBounds(h, B_0)
Input: Vector-field X, starting 1D-interval of the separatrix $\overline{p_0q_0}$, bounding box B_0.
Output: A certified stripe containing a separatrix.

1. Initialize: $\theta \leftarrow \theta_0 (\theta_0 < \frac{\pi}{2})$
2. Rotate(X, θ): Rotate vector-field $X(p)$ anticlockwise by an angle θ.
3. ComputeLeftBoundary(p_0, X_θ): Compute polygonal line-segment l_θ starting from the point p_0 such that the separatrix (passing through the line segment $\overline{p_0q_0}$) remains on the right side of l_θ.
4. Rotate$(X, -\theta)$: Rotate vector-field $X(p)$ clockwise by an angle θ.
5. ComputeRightBoundary$(q_0, X_{-\theta})$: Compute polygonal line-segment $l_{-\theta}$ starting from the point q_0 such that the separatrix (passing through the line segment $\overline{p_0q_0}$) remains on the left side of $l_{-\theta}$.
6. **if** both l_θ and $l_{-\theta}$ meet the same "sink" (or "source") and the region between l_θ and $l_{-\theta}$ does not contain any critical point, then the separatrix converges to that sink; **return**.
7. **else-if** both l_θ and $l_{-\theta}$ meet the boundary of the box B_0 and the region between l_θ and $l_{-\theta}$ does not contain any critical point, then the separatrix converges to the boundary; **return**.
8. **else** $\theta \leftarrow \frac{\theta}{2}$ and **goto** step 2.

Convergence. To prove that the algorithm converges in finite number of steps, we prove the following. First, using lemma (2) we find an upper bound of the number of segment in the line l_θ.

Theorem 3 *Let the change of the height function h along a separatrix, outside the critical region, be H (computed as: $h(final) - h(starting)$). Then an upper bound of the number of segments in l_θ is given by: $\left\lceil \frac{H}{\delta \cos\psi} \right\rceil$.*

This proves that the sub-procedures: ComputeLeftBoundary and ComputeRightBoundary converge in a finite number of steps. Moreover, using the assumption B, we can prove that the global algorithm converges in a finite number of steps.

4 Implementation Results

In this section we illustrate a few implementation outputs with timing results of our algorithm. In figures 6-8 we compute the certified MS-complex of different functions for distinct values of the parameters ϵ_c (described in the algorithm SearchCritical) and angle θ of rotation of the vector field. We use the `Boost` library [1] for IA. All experiments have been performed on a 3GHz Intel Pentium 4 machine under Linux with 1 GB RAM using the `g++` compiler, version 3.3.5.

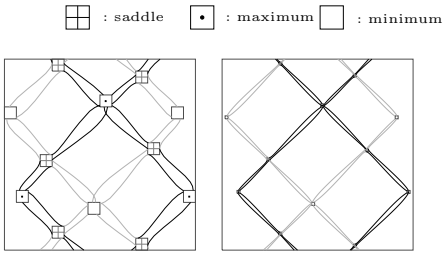


Figure 6: MS-Complex of the function: $\cos x \ \sin y + 0.2\,(x+y)$ for (i) $\epsilon_c = 0.5$, $\theta = \frac{\pi}{10}$, CPU-time = 8 sec., (ii) $\epsilon_c = 0.2$, $\theta = \frac{\pi}{30}$, CPU-time = 20 sec., inside box $[-3.5, 3.5] \times [-3.5, 3.5]$.

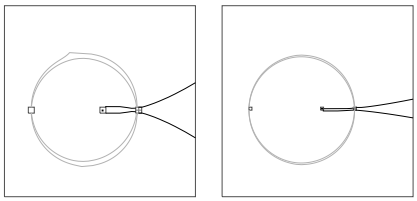


Figure 7: MS-Complex of the function: $10x - \frac{13}{2}(x^2+y^2) + \frac{1}{3}(x^2+y^2)^2$ for (i) $\epsilon_c = 0.5$, $\theta = \frac{\pi}{10}$, CPU-time = 0.16 sec., (ii) $\epsilon_c = 0.2$, $\theta = \frac{\pi}{30}$, CPU-time = 0.5 sec., inside box $[-5, 6] \times [-5, 6]$.

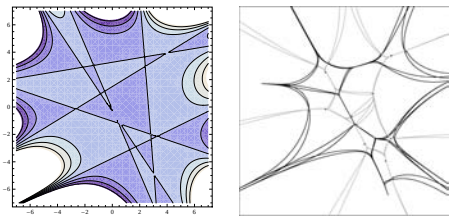


Figure 8: MS-Complex of the function constructed by multiplying 7 linear functions: (i) contour plot using Mathematica, (ii) MS-complex with $\epsilon_c = 0.17$, $\theta = \frac{\pi}{30}$, CPU-time = 15 min., inside box $[-7, 7] \times [-7, 7]$.

Conclusion. The outcome of our research is twofold. Firstly, we compute the topologically correct MS-complex of a Morse-Smale system. The saddle-sink or saddle-source connectivity can also be represented as a graph. On the other hand, depending on a user-specified parameter we can compute the geometrically accurate MS-complex.

References

[1] `Boost` Interval Arithmetic Library. `http://www.boost.org`.

[2] S. Biasotti, L. D. Floriani, B. Falcidieno, P.Frosini, D.Giorgi, C. Landi, L. Papaleo, and M. Spagnuolo. Describing Shapes by Geometrical-Topological Properties of Real Functions. *ACM Computing Surveys*, 40(4):12.1–12:87, 2008.

[3] H. Edelsbrunner, J. Harer, and A. Zomorodian. Hierarchical Morse-Smale Complexes for Piecewise Linear 2-Manifolds. *Discrete Comput. Geom*, 30:87–107, 2003.

[4] J. L. Helman and L. Hesselink. Visualizing Vector Field Topology in Fluid Flows. *IEEE Computer Graphics and Applications*, 11(3):36–46, 1991.

[5] S. Wiggins. *Introduction to applied Nonlinear Dynamical Systems and Chaos*. Springer, USA, 2003.

[6] E. Zhang, K. Mischaikow, and G. Turk. Vector Field Design on Surfaces. *ACM Transactions on Graphics*, 25(4):1294–1326, 2006.

MEMORYLESS ROUTING IN CONVEX SUBDIVISIONS: RANDOM WALKS ARE OPTIMAL

Dan Chen, Luc Devroye, Vida Dujmović, and Pat Morin

Abstract

A *memoryless* routing algorithm is one in which the decision about the next edge on the route to a vertex t for a packet currently located at vertex v is made based only on the coordinates of v, t, and the neighbourhood, $N(v)$, of v. The current paper shows that, for any (randomized) memoryless routing algorithm $\mathcal{A}$, there exists a convex subdivision on which $\mathcal{A}$ takes $\Omega(n^2)$ expected time to route a message to t. This lower bound is matched by a random walk. The current paper also shows the existence of triangulations for which the RANDOM-COMPASS algorithm proposed by Bose *et al* (2002,2004) requires $2^{\Omega(n)}$ time to route between some pair of vertices.

1 Introduction

In recent years, motivated primarily by the proliferation of wireless networks and GPS devices, much research has been done on routing algorithms for geometric networks [4]. In this research a network is modelled as a geometric graph $G = (V, E)$ whose vertex set V is a set of points in $\mathbb{R}^2$. We say that a routing algorithm $\mathcal{A}$ *works* for G if, for any pair of vertices $s, t \in V$, the algorithm always find a path from s to t in a finite number of steps.

The research on geometric routing algorithms largely focuses on utilizing geometric properties of a class of geometric graphs to reduce the complexity of, and information required by, routing algorithms. A particularly interesting and restricted class of routing algorithms are so-called memoryless routing algorithms. A *memoryless* routing algorithm is one in which the decision about the next edge on the route to t for a packet currently located at node v is based only on the coordinates of v, t, and the neighbourhood, $N(v)$, of v. More precisely, a deterministic memoryless routing algorithm is a function $f : \mathbb{R}^2 \times \mathbb{R}^2 \times (\mathbb{R}^2)^+ \to \mathbb{R}^2$ that satisfies $f(v, t, N(v)) \in N(v)$ and $f(t, t, N(t)) = t$ for all inputs. Note that a memoryless routing algorithm makes each routing step without using information obtained in previous routing steps and without any global information about G.

Unfortunately, deterministic memoryless routing algorithms have severe limitations. These stem from the fact that these algorithms can not visit the same vertex more than once without looping forever. Bose *et al* [2, Theorem 2] show that there exists 17 convex subdivisions[1], $G_1, \ldots, G_{17}$, each with 17 vertices such that any deterministic memoryless routing algorithm does not work for at least one of these subdivisions. Thus, convex subdivisions form a class of geometric graphs that are too rich for deterministic memoryless routing algorithms [1].

The same authors [1, 2] observe that randomization can be used to overcome this limitation. A *randomized memoryless* routing algorithm is one in which the decision about the next edge on the route to t for a packet currently located at node v is based only on v, t, the neighbourhood, $N(v)$, of v, and a sequence B of fresh random bits. More precisely, a randomized memoryless routing algorithm is defined by a function $f : \mathbb{R}^2 \times \mathbb{R}^2 \times (\mathbb{R}^2)^+ \times \{0,1\}^\infty \to \mathbb{R}^2$ that satisfies $f(v, t, N(v), B) \in N(v)$ and $f(t, t, N(v), B) = t$ for all inputs. The final argument B is a sequence of random bits that are chosen fresh for each step taken by the routing algorithm. Bose *et al* describe a randomized memoryless algorithm, named RANDOM-COMPASS, that uses one random bit per step works for any convex subdivision. They do not analyze the efficiency of RANDOM-COMPASS except to note that, for some convex subdivisions G, and some pairs $s, t \in V$, the expected number of steps taken by RANDOM-COMPASS when routing from s to t is $\Omega(|V|^2)$.

Observe that, by the theory of random walks (c.f. [6, Theorem 6.6]), the expected time required for a random walk on G to travel from a particular vertex s to a particular vertex t is $O(n^2)$. Therefore, a random walk is at least as efficient, in the worst case, as the RANDOM-COMPASS algorithm. Nevertheless, one might expect that RANDOM-COMPASS is more likely to find short routes, since it uses geometry to find a route that is specifically directed towards the target vertex t. Thus, we might intuit that RANDOM-COMPASS is a heuristic that is usually better than a random walk and never much worse.

In the current paper, we show that this intuition about RANDOM-COMPASS could not be further from the truth. Indeed, for any $n > 0$, there exists a convex subdivision (in fact, a triangulation) G with n vertices and having two vertices s and t such that the expected

[1] A *convex subdivision* is a geometric graph all of whose faces, except the outer face, are convex polygons, and whose outer face is the complement of a convex polygon.

number of steps taken by RANDOM-COMPASS when routing from s to t is $2^{\Omega(n)}$. This triangulation has diameter 3.

Next we study whether *any* randomized memoryless routing algorithm for convex subdivisions can outperform a random walk. We show that, for any randomized memoryless routing algorithm $\mathcal{A}$ and any n, there exists a convex subdivision $G = G(\mathcal{A}) = (V, E)$ of size n and a pair of vertices $s, t \in V$ such that the expected number of steps taken by $\mathcal{A}$ when routing from s to t is $\Omega(n^2)$. Therefore, at least in the worst-case, no algorithm significantly outperforms a random walk.

2 A Bad Example for Random-Compass

The RANDOM-COMPASS algorithm works by using a coin toss to select among the (at most two) neighbours $\mathrm{ccw}_t(v)$ and $\mathrm{cw}_t(v)$ of the current node v that make the minimum and maximum angle, respectively, with the segment vt (see Figure 1.a). When applied on a convex subdivision $G = (V, E)$, Bose *et al* show that, in the directed graph G' that contains the edges $(v, \mathrm{cw}_t(v))$ and $(v, \mathrm{ccw}_t(v))$ for all $v \in V$, there exists at least one directed path $P(v, t)$ from every vertex v to t (see Figure 1.b). This, and Wald's Equation, immediately imply that the expected time to reach t from any vertex is at most 2^n; from any vertex v, RANDOM-COMPASS has probability at least $1/2^{|P(v,t)|} \geq 1/2^{n-1}$ of reaching t by following $P(v, t)$, and the expected number of steps it takes on $P(v, t)$ before falling off $P(v, t)$ is at most 2.

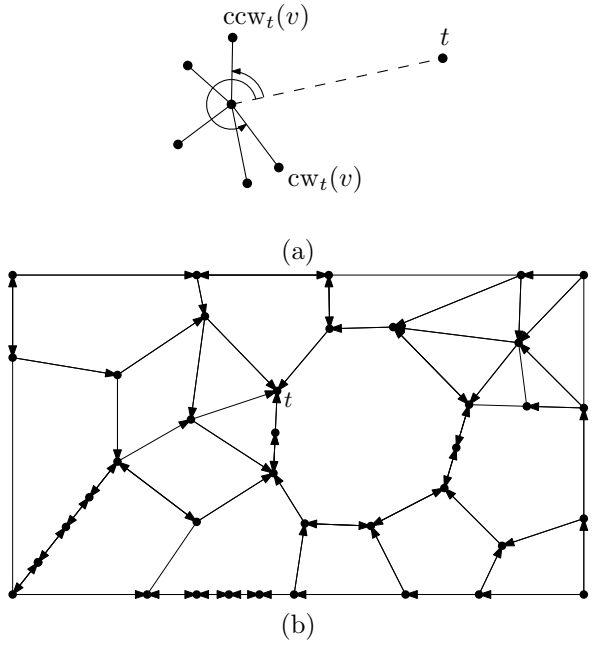


Figure 1: The RANDOM-COMPASS algorithm chooses the next vertex at random among $\mathrm{ccw}_t(v)$ and $\mathrm{cw}_t(v)$.

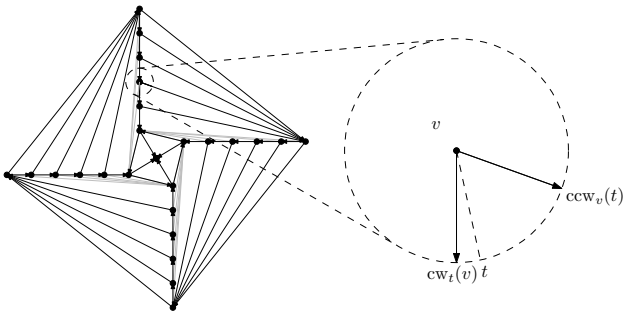


Figure 2: A graph in which RANDOM-COMPASS has expected running time $\Omega(2^{n/4})$.

The example in Figure 2 shows that the above analysis of RANDOM-COMPASS, although very coarse, is about the best one can do. It shows a geometric graph G whose vertex set has size $n = 4k + 1$ and whose vertices are organized as a central vertex t and four paths leading from the outer face to t. The space between these paths is triangulated so that, at any point, RANDOM-COMPASS chooses between an edge that leads one step closer to t or that returns to the outer face.

If we consider the directed graph G' defined above, then we see that, at any point the packet is at some distance i from t and that, it can, with equal probability, move to a vertex of distance $i - 1$ or move to a vertex (on the outer face) of distance k. If we denote by T_i the expected number of steps required by RANDOM-COMPASS to reach t given that it is currently at distance i from t, we see that

$$T_i = \begin{cases} 0 & \text{for } i = 0 \\ (1/2)T_{i-1} + (1/2)T_k & \text{for } i \in \{1, \ldots, k\} \end{cases}$$

Solving this recursion gives $T_k = 2^k(2 - 1/2^k) = \Omega(2^{n/4})$. This proves:

Theorem 1 *For any $n > 1$, there exists a triangulation G having two vertices s and t such that the expected number of steps taken by* RANDOM-COMPASS *when routing from s to t is $2^{\Omega(n)}$.*

Note that the base in the exponent can be improved by using a construction with 3 paths instead of 4. In this case, the lower bound becomes $\Omega(2^{n/3})$. Furthermore, up to a factor of 2, the lower bound on Theorem 1 holds for all choices of the source vertex s since, for any vertex $s \neq t$, the expected time to route from s to t is at least $(1/2)T_k$.

3 A Lower Bound for Any Algorithm

In this section we develop an $\Omega(n^2)$ lower bound for routing on convex subdivisions using any randomized memoryless routing algorithm $\mathcal{A}$. The outline of the lower bound is as follows: We start with a lemma about Markov chains whose transition graphs are paths. We show that, when starting at the midpoint of the path, there is at most one endpoint of the path that can be reached in subquadratic expected time. This lemma is relevant since, if $\mathcal{A}$ finds itself in the interior of a path of degree 2 vertices in G, it will behave like such a Markov chain until it reaches one of the endpoints of this path.

Next, we observe how $\mathcal{A}$ behaves on certain paths of degree 2 vertices and show that, because $\mathcal{A}$ can only reach one endpoint of any path in subquadratic time, that we can always find a subset of these paths that can be pieced together to form a convex subdivision in which $\mathcal{A}$ takes at least quadratic expected time to route from some vertex s to some vertex t.

3.1 Markov Chains

Consider a Markov chain on $\{1, \ldots, n\}$, $n > 1$, where transitions only take place between neighbors. If $p_{i,j}$ is the probability of a transition from i to j, then we have

$$p_{1,2} = p_{n,n-1} = 1,$$
$$p_{i,i+1} = 1 - p_{i,i-1} = \pi_i, 2 \leq i \leq n-1,$$

where $\pi_2, \ldots, \pi_{n-1}$ are fixed probabilities. The vector of these probabilities is denoted by π. We will set $\pi_1 = 1$, $\pi_n = 0$, to be consistent, as the extreme states are reflecting. When $\pi_i = 1/2$ for $2 \leq i \leq n-1$, we obtain a standard random walk on a finite interval with reflecting barriers.

We denote the Markov chain by $X_0, X_1, \ldots, X_t, \ldots$, and denote the hitting times by $T_{i,j}$:

$$T_{i,j} = \min\{t > 0 : X_t = j | X_0 = i\}.$$

For a standard random walk, it is known that

$$\mathrm{E}\{T_{i,j}\} = (j-i)^2, j \neq i, 1 \leq j, i \leq n$$

[5]. The standard random walk is in fact the best possible chain in the following sense:

Lemma 2 *For any vector of probabilities π, and any $n > 1$,*

$$\mathrm{E}\{T_{1,n} + T_{n,1}\} \geq 2(n-1)^2.$$

Next we present a simple corollary of Lemma 2 that is used in our lower bound.

Corollary 3 *Consider a random walk with reflecting barriers on $\{-n, \ldots, n\}$, $n > 0$. In this chain,*

$$\max\left(\mathrm{E}\{T_{0,n}\}, \mathrm{E}\{T_{0,-n}\}\right) \geq \frac{2}{3}n^2.$$

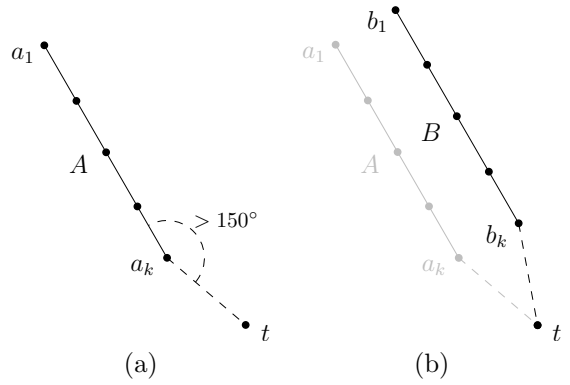


Figure 3: The chains A and B.

The proofs of Lemma 2 and Corollary 3 are omitted due to space constraints and can be found in the full version of the paper [3].

3.2 The Lower Bound

Let $\mathcal{A}$ be a randomized memoryless routing algorithm. Let k be an even integer, let t be the origin, and let $A = a_1, \ldots a_k$ be a path of k collinear vertices such that a_k is closer to t than any of $a_1, \ldots, a_{k-1}$ and the three points a_1, a_k, t make a left turn with $\angle a_1 a_k t$ greater than 150° degrees but less than 180° (see Figure 3.a). Let $B = b_1, \ldots, b_k$ be the reflection of A through the line parallel to A that contains t (see Figure 3.b). Let $A(\alpha)$, respectively, $B(\alpha)$, denote the path A, respectively, B, rotated by an angle of α about the origin, t.

Define the *color* of a path $A(\alpha) = a'_1, \ldots, a'_k$ as follows: Imagine running $\mathcal{A}$ on the graph consisting of A' and the isolated vertex t, starting at $a'_{k/2}$. If $\mathcal{A}$ takes $\Omega(k^2)$ expected time to reach a'_k then color $A(\alpha)$ *blue*, otherwise color $A(\alpha)$ *red*. Note that Corollary 3 implies that, if $A(\alpha)$ is red, then $\mathcal{A}$ takes $\Omega(k^2)$ expected time to reach a_1 starting at $a_{k/2}$.

Intuitively, a path is red (getting hotter — closer to t) if $\mathcal{A}$ could move quickly from $a_{k/2}$ to a_k. A path is blue (getting cooler — further from t) if $\mathcal{A}$ could move quickly to a_1. Define the color (red or blue) of a path $B(\alpha)$ in the same way.

Lemma 4 *If there exists α such that $A(\alpha)$ and $B(\alpha)$ are both blue, then there exists a convex subdivision $G = (V, E)$ with $|V| = 2k + 1$ with vertices $s, t \in V$ such that $\mathcal{A}$ takes $\Omega(k^2)$ steps when routing from s to t.*

Proof. Let $A' = A(\alpha) = a'_1, \ldots, a'_k$ and $B' = B(\alpha) = b'_1, \ldots, b'_k$. The convex subdivision G consists of A' and B' as well as the edges $a'_1 b'_1$, $a'_k t$ and $b'_k t$ (see Figure 4). Since $A(\alpha)$ and $B(\alpha)$ both blue, applying $\mathcal{A}$ to route from $a'_{k/2}$ to t will require $\Omega(k^2)$ expected steps. □

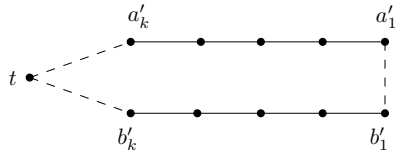


Figure 4: Two blue chains $A(\alpha)$ and $B(\alpha)$.

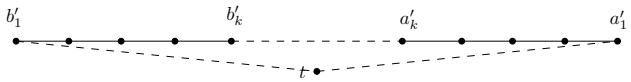


Figure 5: Two red chains $A(\alpha)$ and $B(180+\alpha)$.

Lemma 5 *If there exists α such that $A(\alpha)$ and $B(180+\alpha)$ are both red, then there exists a convex subdivision $G=(V,E)$ with $|V|=2k+1$ with vertices $s,t\in V$ such that $\mathcal{A}$ takes $\Omega(k^2)$ steps when routing from s to t.*

Proof. Let $A' = A(\alpha) = a'_1,\ldots,a'_k$ and $B' = B(180+\alpha) = b'_1,\ldots,b'_k$. The convex subdivision G consists of A' and B' as well as the edges $a'_kb'_k$, a'_1t and b'_1t (see Figure 5). Since $A(\alpha)$ and $B(180+\alpha)$ are red, applying $\mathcal{A}$ to route from $a'_{k/2}$ to t will require $\Omega(k^2)$ expected steps. □

Theorem 6 *For any integer $k>0$ and any memoryless routing algorithm $\mathcal{A}$, there exists a convex subdivision $G=(V,E)$ with $|V|=\Theta(k)$ having vertices $s,t\in V$ such that $\mathcal{A}$ takes $\Omega(k^2)$ steps when routing from s to t.*

Proof. If either of Lemma 4 or Lemma 5 apply to $\mathcal{A}$ then the proof is complete. Otherwise, observe that the exclusion of these two lemmata implies that, for any α, at least one of $A(\alpha)$ and $B(\alpha)$ is blue. To see this, note that if $A(\alpha)$ is red, then (the exclusion of) Lemma 5 implies that $B(\alpha+180)$ is blue, so (the exclusion of) Lemma 4 implies that $A(\alpha+180)$ is red, so (the exclusion of) Lemma 5 implies that $B(\alpha)$ is blue.

Therefore, there exists 3 blue chains $X = x_1,\ldots,x_k$, $Y = y_1,\ldots,y_k$, and $Z = z_1,\ldots,z_k$ where $X\in\{A(0),B(0)\}$, $Y\in\{A(120),B(120)\}$ and $Z\in\{A(240),B(240)\}$. We can then take G to be the graph containing X, Y, and Z, as well as the edges x_1y_1, y_1z_1, z_1x_1, x_ky_k, y_kz_k, z_kx_k, x_kt, y_kt, z_kt (see Figure 6). Because X, Y, and Z are all blue, the expected number of steps required to route from $x_{k/2}$ to t using $\mathcal{A}$ is $\Omega(k^2)$.

All that remains is to verify that G is indeed a convex subdivision. This is readily established using the fact that the angles $\angle x_1x_kt$, $\angle y_1y_kt$, and $\angle z_1z_kt$, are all between 150 and 180 degrees. □

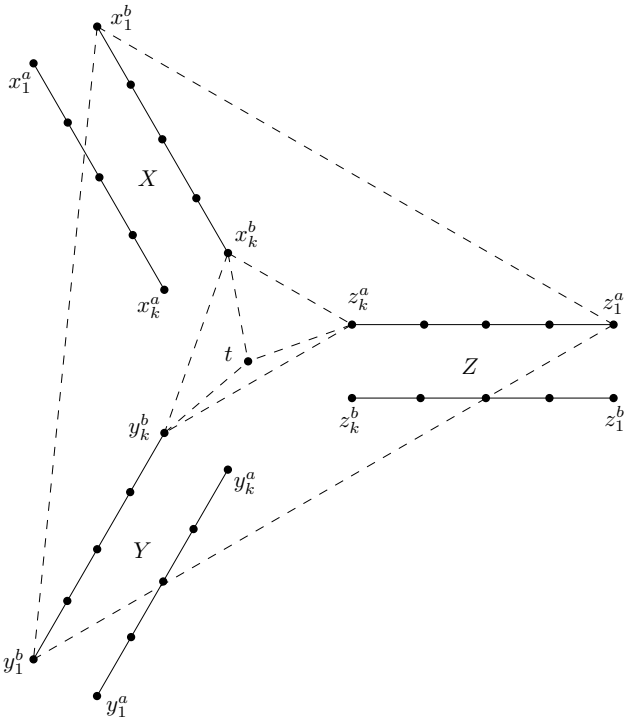


Figure 6: Three blue chains

References

[1] P. Bose, A. Brodnik, S. Carlsson, E. D. Demaine, R. Fleischer, A. López-Ortiz, P. Morin, and J. I. Munro. Online routing in convex subdivisions. *International Journal of Computational Geometry and Applications*, 12(4):283–296, 2002.

[2] P. Bose and P. Morin. Online routing in triangulations. *SIAM Journal on Computing*, 33(4):937–951, 2004.

[3] D. Chen, L. Devroye, V. Dujmovic, and P. Morin. Memoryless routing in convex subdivisions: Random walks are optimal. *arXiv*, 0911.2484v1, 2009.

[4] S. Giordano and I. Stojmenović. Position based routing algorithms for ad hoc networks: A taxonomy. In X. H. X. Cheng and D.-Z. Du, editors, *Ad Hoc Wireless Networking*, pages 103–136. Kluwer, 2003.

[5] J. W. Moon. Random walks on random trees. *Journal of the Australian Mathematical Society*, 15:42–53, 1973.

[6] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

Planar Hop Spanners for Unit Disk Graphs *

Nicolas Catusse† Victor Chepoi† Yann Vaxès†

Abstract

In this paper, we present an algorithm that, given a set of n terminals in the plane, constructs a planar hop spanner with constant stretch for the Unit Disk Graph defined by this set of terminals. This algorithm improves on previous constructions in the sense that (i) our construction ensures the planarity for the whole graph while previous algorithms ensure only the planarity of a backbone subgraph; (ii) the hop stretch factor provided by our algorithm is significantly smaller.

1 Introduction

Given a connected graph $G = (V, E)$ with n vertices embedded in the Euclidian plane, let $d_G(u, v)$ be the length of a shortest path connecting u and v in G. A subgraph H of G is a *spanner* of G if there is a positive real constant t such that for any two vertices, $d_H(u, v) \leq td_G(u, v)$. The constant t is called the *length stretch factor* if the length of an edge is the Euclidian distance between its endpoints and the *hop stretch factor* if each edge has length 1.

The problem of constructing sparse spanners of geometrics graphs has received considerable attention from researchers in computational geometry and ad-hoc wireless networks; we refer the reader to the book by Narasimhan and Smid [7]. The simplest model of a wireless network graph is the *Unit Disk Graph* (UDG): an edge between two terminals u, v exists in this graph if the Euclidian distance between u and v is at most one. Notice also that some routing algorithms such as *Greedy Perimeter Stateless Routing* require a *planar* subgraph to route the messages through the network.

In this paper, we design an algorithm that, given a set of n points on the plane, constructs a planar spanner with constant hop stretch factor for the unit disk graph defined by these points. Contrary to the problem of constructing planar *Euclidean length* spanners, for which several algorithms provide small stretch factors (see [6], for instance), the problem of constructing planar *hop* spanners with constant stretch factor remained open. Some partial solutions ensuring the planarity of a certain backbone subgraph were proposed in [1, 5]. Our algorithm improves on the results of [1, 5] in the sense that (i) our construction ensures the planarity for the whole graph; (ii) the hop stretch factor provided by our algorithm is significantly smaller.

The rest of the paper is organized as follows. In Section 2, we briefly review the literature related to geometric spanners and Unit Disk Graphs. Section 3 presents a very simple construction that provides a sparse spanner for UDG with low hop stretch factor. Unfortunately, this construction does not ensure the planarity of the spanner. Section 4 describes an algorithm that updates the spanner defined in Section 3 in order to obtain a planar spanner while preserving a small hop stretch factor.

2 Previous Works

Spanner properties of some geometric graphs have been surveyed by Eppstein in [3]. Bose, Devroye, Evans and Kirkpatrick [2] proved that the *Gabriel Graph* is an $\Omega(\sqrt{n})$ hop spanner and a $\Theta(\sqrt{n})$ Euclidean spanner for UDG, and that the *Relative Neighborhood Graph* is a $\Theta(n)$ and a $\Theta(n)$ Euclidean spanner for UDG. Gao, Guibas, Hershberger, Zhang and Zhu [5] proposed an algorithm to construct a hop spanner for UDG. This algorithm creates several clusters connected by a *Restricted Delaunay graph* which is planar. This construction cannot be distributed and its hop stretch factor is not given. Alzoubi, Li, Wang, Wan and Frieder [1] proposed for the same problem a distributed algorithm that uses the *Local Delaunay Triangulation* defined by Li, Călinescu and Wan in [6]. However, the hop stretch factor is huge (around 15000) and the intra-cluster edges may cross the edges of the triangulation (and therefore does not provide a full planar hope spanner). Chen, Jiang, Kanj, Xia and Zhang [4] presented the construction of Euclidean spanners for *Quasi-UDG* which can be used for routing. Their construction method is similar to our approach in the sense that it also uses a regular squaregrid to partition the set of terminals into clusters.

3 Sparse almost planar spanners

Let X be a set of n points (terminals) in the plane. In this section, we describe a very simple algorithm that constructs a sparse 5-spanner $H' = (X, E')$ of

*This research was partly supported by the ANR grant BLAN06-1-13889 (projet OPTICOMB).

†Laboratoire d'Informatique Fondamentale, Université d'Aix-Marseille, Faculté des Sciences de Luminy, F-13288 Marseille Cedex 9, France{nicolas.catusse, victor.chepoi, yann.vaxes}@lif.univ-mrs.fr

the Unit Disk Graph $G = (X, E)$ defined by X. It uses a regular grid Γ on the plane with squares of side $\frac{\sqrt{2}}{2}$. A square of Γ is said to be *nonempty* if it contains at least one terminal from X. For any point $x \in X$, let $\pi(x)$ denote the square of Γ containing x.

The graph $H' = (X, E')$ has two types of edges : a subset $E'_0 \subseteq E'$ of edges connecting points lying in the same square and a subset E'_1 of edges running between points lying in neighboring squares, let $E' = E'_0 \cup E'_1$. To define E'_0, in each nonempty square π we pick a terminal (the *center* of π) and add to E'_0 an edge between this terminal and every other terminal located in π. Clearly, all of them are edges of G because the side of squares is $\frac{\sqrt{2}}{2}$. In E'_1 we put exactly one edge of G running between two nonempty squares if such an edge exists. In the sequel, with some abuse of notation, we will denote by $\pi\pi'$ the shortest edge of UDG running between two squares π and π'.

Algorithm 1 Construction of sparse spanner H'

1: For each square π, pick a terminal c_π (the *center* of π) and add to E'_0 an edge between c_π and every other terminal located in π.
2: For each pair of squares π and π' connected by an edge of G, add the shortest edge between π and π' to E'_1.

Recall that a subgraph $H' = (V, F')$ of a graph $H = (V, F)$ is called a *t-spanner* for H if $d_{H'}(u, v) \leq t d_H(u, v)$ for any two vertices $u, v \in V$. Notice that it is sufficient to ensure this condition for every pair of vertices u, v adjacent in H.

Proposition 3.1 *The graph $H' = (X, E')$ is a hop 5-spanner for G containing at most $10n$ edges.*

Proof. For the first assertion, it is sufficient to prove that $d_{H'}(u, v) \leq 5 d_G(u, v)$ for any two adjacent vertices u and v in G. If u, v belong to the same square π, then they are neighbors of the center of π, hence they are connected in H' by a path of length 2. Now, suppose that u and v belong to different squares. Since uv is an edge of G, the graph H' must contain an edge $u'v'$ of G with $u' \in \pi(u)$ and $v' \in \pi(v)$. Therefore the terminals u and v are connected in H' by a path of length ≤ 5 consisting of two paths of length 2 passing via the centers of the clusters $\pi(u)$ and $\pi(v)$ and connecting u and v to u' and v', respectively, and the edge $u'v'$ joining these clusters. To prove the second assertion, let n_0 denote the number of non-empty squares. Then obviously $|E'_0| \leq n - n_0$. Since from each nonempty square π we can have edges of E'_1 to at most 20 other such squares, and since each such edge is counted twice, we conclude that $|E'_1| \leq 10 n_0$. Therefore $|E'| \leq (n - n_0) + 10 n_0 \leq 10n$. □

4 Planar spanners

In this section, we describe how to build a planar hop spanner H for a UDG graph G. We first compute a planar set of *inter-cluster edges* E_3 whose end-vertices belong to distinct squares of Γ. Then we compute a second set of *intra-cluster edges* E_0 connecting the vertices that belong to the same square.

4.1 Computing E_3

The edge set E_3 is a planar subset of the edge set E'_1 defined in the previous section. We must remove some edges to obtain a planar graph while preserving a bounded hop stretch factor of the resulting spanner.

First we define the l_1-*distance* between two squares π and π' in Γ as the graph distance between π and π' in the dual grid (squares become vertices and two vertices are adjacent if their squares have a common side). The l_1-*length* of an edge of G is the l_1-distance between the squares that contain their end-vertices. The principle of our algorithm is to minimize the l_1-length of remaining edges: if the end-vertices of an edge uv are joined by a path having the same total l_1-length as uv, then uv is removed. An edge with large l_1-length potentially crosses many squares and, as a consequence, many edges of UDG. Hence, taking such an edge in our planar spanner would exclude many other (potentially, good) edges from the spanner. For each removed edge, there is a path having a constant number of edges between its end-vertices. Therefore, after the removal of these edges, the stretch factor is still bounded by a constant. The minimization of the l_1-length is not sufficient to obtain a planar graph but we show that this operation considerably decreases the number of crossing configurations. The next step of the algorithm consists in repairing the remaining crossings. During this step some edges are removed to ensure the planarity and some other edges are added back to preserve the distance between the end-vertices of removed edges. Using the proof outlined below, we establish that the edge set obtained at the end of this process is planar.

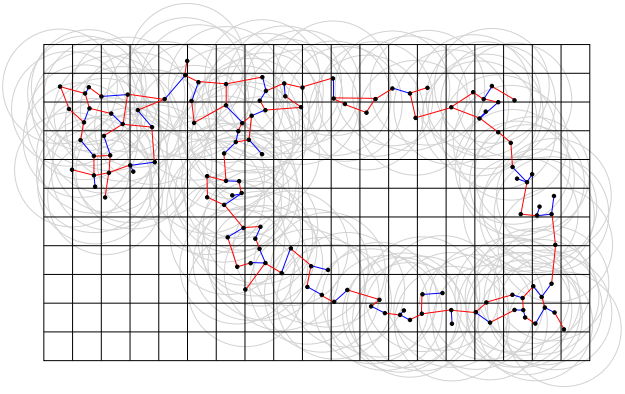


Figure 1: Spanner H

Algorithm 2 Construction of the spanner H

1: Let $H' = (X, E'_1)$ be the inter-cluster graph returned by the algorithm **Construction of Sparse Spanner** on input X.
2: Let $G_1 = (X, E_1)$ be the graph obtained from H' by removing every edge $ab \in E'_1$ whose end-vertices are joined by a *replacement path,* i.e., a path $P \neq ab$ between $\pi(a)$ and $\pi(b)$ such that $l_1(P) = l_1(ab)$.
3: For each pair of crossing edges $xy, x'y'$ of E_1, identify the crossing configuration (according to Fig. 2) and remove the edge $x'y'$. Let $G_2 = (X, E_2)$ be the graph obtained from G_1 by removing these edges.
4: For each edge $x'y'$ removed during Step 3, except if $xy, x'y'$ form a Configuration 0, if there is no replacement path in G_2 for $\pi(y)\pi(y')$, then add the edge $\pi(y)\pi(y')$. Let $G_3 = (X, E_3)$ be the resulting graph.
5: Compute the set of intra-cluster edges E_0 as described in sub-section 4.2
6: Output the graph $H = (X, E_0 \cup E_3)$.

Theorem 1 *The inter-cluster graph $G_3 = (X, E_3)$ is planar.*

To prove this theorem, we proceed in several steps (the proofs of Propositions 4.1-4.5 require several case analysis and are omitted from this abstract). First, we classify the edges of G according to the relative positions of the squares containing their end-vertices. Then, using this classification, we consider all possible crossing configurations between two edges of E_1. Proposition 4.2 analyzes these configurations and shows that in most cases one of the two crossing edges has a replacement path.

Proposition 4.1 *If the crossing configuration between two edges of $xy, x'y' \in E'_1$ does not belong to the list of Fig. 2, then one of these edges, say xy, has a replacement path passing via $\pi(x')$ or $\pi(y')$.*

Since the edges of E_1 do not admit replacement paths, we conclude that only a few crossing configurations may occur between two edges of E_1.

Proposition 4.2 *For two edges of E_1, there exist only six possible crossing configurations listed in Fig. 2.*

Now we consider the edges added at Step 4. To prove that the graph returned by our algorithm is planar, we will show that these new edges do not intersect each other and do not intersect the edges that survive Step 3. Let $e' \in E_3 - E_2$ be an edge added during Step 4, let $e \in E_3 \cap E_2$ be an edge crossing e' that survived Step 3. Let also $xy, x'y' \in E_1$ be

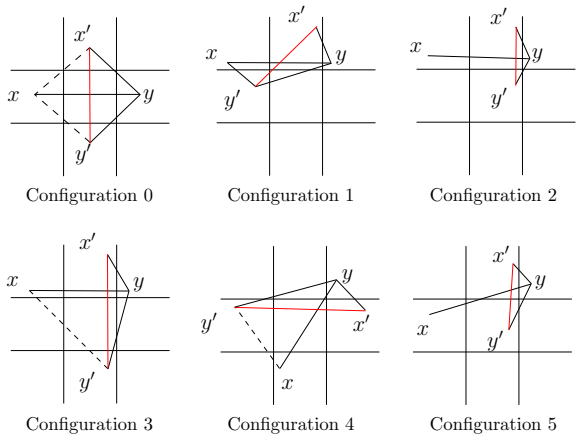


Figure 2: The six remaining configurations after Step 2. The solid lines are edges in E'_1 and the dashed lines are edges that may be in E'_1 or not.

the crossing edges because of which the edge $x'y'$ has been removed and the edge $e' = \pi(y)\pi(y')$ has been added (by examining all crossing configurations listed in Fig. 2 one can check that $\pi(y).\pi(y')$ has been removed during step 2). By a case analysis of all configurations listed in Fig. 2, we verify that in all cases except Configuration 0 (that can be easily treated separately) there exists a replacement path for e' in E_1 passing via $\pi(x')$. Since the edge e' is added at Step 4 only if it does not admit a replacement path in E_2, the following result excludes the existence of a replacement path for e' distinct from the path going through $\pi(x')$.

Proposition 4.3 *If the edge $\pi(y)\pi(y')$ of Fig. 2 has a replacement path in E_1 distinct from the path that goes through $\pi(x')$, then there is a replacement path between $\pi(y)$ and $\pi(y')$ in E_2.*

To deal with the crossing configurations formed by the edges e and e', we distinguish two cases. Either such a configuration belongs to the list from Fig. 2 and a case analysis leads to a contradiction, or, by Proposition 4.1, one of these two edges admits a replacement path that passes through a square containing an end-vertex of the other edge. This edge must be e' because e was not removed during Step 2. However, as noticed above, Proposition 4.3 implies that this path must be the one passing through $\pi(x')$ because otherwise this replacement path would survive Step 3 and e' would not be added. Since the replacement path arising in Proposition 4.1 passes via a square containing an end-vertex of e', the edge e must be incident to $\pi(x')$. Therefore, in the proof of Proposition 4.4 we analyze all possible crossings between e' and an edge incident to $\pi(x')$.

Proposition 4.4 *An edge from $E_3 - E_2$ cannot cross an edge from $E_3 \cap E_2$.*

Finally, we prove that edges added during Step 4 do not cross each other. First, notice that if two edges of $E_3 - E_2$ cross, then one of them crosses an edge from E_2. By Proposition 4.4, this edge of E_2 was removed during Step 3. Hence, we get a crossing between an edge of $E_3 - E_2$ and an edge of $E_2 - E_3$. The list of these configurations can be extracted from the proof of Proposition 4.4. They are analyzed case by case to get the following proposition (and to conclude the outline of the proof of Theorem 1):

Proposition 4.5 *The edges of $E_3 - E_2$ cannot cross.*

4.2 Computing E_0

We will consider several choices for the set of edges E_0 that interconnect the vertices lying in the same square. Let $\alpha(E_0)$ be the maximum distance in the graph $G_0 = (X, E_0)$ between two vertices belonging to the same square. In the next subsection, the hop stretch factor of our spanner is expressed as a function of $\alpha(E_0)$. A possible choices for E_0 is to set a clique or a star on each non-empty square, yielding $\alpha(E_0) = 1$ and $\alpha(E_0) = 2$, respectively. In this case, the diameter of the clusters is small but we do not get a planar spanner. The following proposition shows that an appropriate choice of E_0 ensures both planarity and constant diameter of clusters.

Proposition 4.6 *There exists a set of intra-cluster edges E_0 such that $H = (X, E_3 \cup E_0)$ is planar and $\alpha(E_0) \leq 66$.*

This set E_0 can be obtained as follows. First, we partition X into clusters consisting of all vertices that belong to the same non-empty square of the grid Γ. Then, we consider each non-empty square π crossed by an edge $e \in E_3$ (one can show that there is at most one such edge) so that the two regions R_1 and R_2 of π separated by e are non-empty. We subdivide the subset of vertices that belong to π into two subsets corresponding to the regions R_1 and R_2. We consider each pair of adjacent partitions (i.e. the partitions whose regions intersect) and we add a shortest edge between them if one exists. Then, we compute a minimum spanning tree on the vertices of Y that are connected by an edge to a vertex outside π (i.e. the vertices of $Y \cap V(E_3)$ plus the end-vertices of the edges that connect the adjacent partitions). Now it remains to connect the subsets arising from a square crossed by an edge of E_3. One can prove that this can always be done with a path passing through at most two neighboring squares. As already noticed in Section 3, there are at most 20 edges of E_3 having an end-vertex in a given square. From this fact, we deduce that there is a path of length at most 66 between every pair of vertices that belong to the same square of Γ.

4.3 Hop stretch factor

Now, we analyze the hop stretch factor of H and prove that it is a $10\alpha(E_0)+9$ hop spanner for G. As noticed above, it suffices to prove the spanner property for two adjacent vertices in G. After Step 1, we get a spanner H' whose hop stretch factor is at most $2\alpha(E_0)+1$. Since the maximal l_1-length of an edge from G is 3, if an edge from E_1 is removed during Step 2, then it is replaced by a path containing at most 3 edges from E_2. Taking into account the edges from E_0, we obtain a stretch factor $\leq 4\alpha(E_0)+3$. In Step 3, again some edges removed and are replaced by paths. The following proposition asserts that the length of these paths is bounded. Finally, in Step 4 edges are only added.

Proposition 4.7 *If an edge from E_2 is removed during Step 3, then it is replaced by a path containing at most $3l_1(a) \leq 9$ edges from E_3.*

The Proposition 4.7 shows that after Step 4 an inter-cluster edge of G is replaced by at most 9 inter-cluster edges of E_3. Taking into account the edges of E_0, we get a stretch factor equal to $10\alpha(E_0) + 9$.

Summarizing, here is the main result of this note:

Theorem 2 *Given a set of terminals in the plane, the graph $H = (X, E_0 \cup E_3)$ computed by our algorithm is a planar spanner of G with hop stretch factor at most $10\alpha(E_0) + 9$.*

References

[1] K. Alzoubi, X. Li, Y. Wang, P. Wan, and O. Frieder, *Geometric spanners for wireless ad hoc networks,* IEEE Transactions on Parallel and Distributed Systems, **14(4)** (2003), 408–421.

[2] P. Bose, L. Devroye, W. Evans, and D. Kirkpatrick, *On the spanning ratio of gabriel graphs and β-skeletons,* Proceedings of the 5th Latin American Symposium on Theoretical Informatics, (2002), 476–493.

[3] D. Eppstein, *Spanning trees and spanners,* In J.-R. Sack and J. Urrutia, editors, Handbook of Computational Geometry, Elsevier Science Publishers B.V., North-Holland, Amsterdam (2000), 425–461.

[4] J. Chen, A. Jiang, I. A. Kanj, G. Xia, and F. Zhang, *Separability and topology control of quasi unit disk graphs,* In INFOCOM 2007, 26th IEEE International Conference on Computer Communications IEEE, (2007), 2225–2233.

[5] J. Gao, L. Guibas, J. Hershberger, L. Zhang, and A. Zhu, *Geometric spanner for routing in mobile networks,* Proc. ACM MobiHoc01, (2001), 45–55.

[6] X. Li, G. Călinescu, and P. Wan, *Distributed construction of planar spanner and routing for ad hoc wireless networks,* IEEE INFOCOM, (2002).

[7] G. Narasimhan and M. Smid, *Geometric Spanner Networks.* Cambridge University Press, 2007.

Embedding into the rectilinear plane in optimal $O(n^2)$ time*

Nicolas Catusse† Victor Chepoi† Yann Vaxès†

Abstract

In this paper, we present an optimal $O(n^2)$ time algorithm for deciding if a metric space (X, d) on n points can be isometrically embedded into the plane endowed with the l_1-metric. It improves the $O(n^2 \log^2 n)$ time algorithm of J. Edmonds (2008). Together with some ingredients introduced by Edmonds, our algorithm uses the concept of tight span and the injectivity of the l_1-plane. A different $O(n^2)$ time algorithm was recently proposed by D. Eppstein (2009).

1 Introduction

Deciding if a finite metric space (X, d) admits an isometric embedding or an embedding with a small distortion into a given geometric space (usually $\mathbb{R}^k$ endowed with some norm-metric) is a classical question in distance geometry which has some applications in theoretical computer science, visualization, and data analysis. The first question can be answered in polynomial time if $\mathbb{R}^k$ is endowed with the Euclidean metric due to classical results of Menger and Schönberg [5]. On the other hand, by a result of Frechet [5], any metric space can be isometrically embedded into some $\mathbb{R}^k$ with the l_∞-metric. However, it is NP-hard to decide if a metric space isometrically embeds into some $\mathbb{R}^k$ endowed with the l_1 (alias rectilinear or Manhattan) metric [5]. More recently, Edmonds [8] established that it is even NP-hard to decide if a metric space embeds into $\mathbb{R}^3$ with l_∞-metric (a similar question for $\mathbb{R}^3$ with l_1-metric is still open). In case of $\mathbb{R}^2$, l_1- and l_∞-metrics are equivalent. The embedding problem for the rectilinear plane was investigated in the papers [2, 11], which ultimately show that a metric space (X, d) embeds into the l_1-plane if and only if any subspace with at most six points does [2] (a similar result for embedding into the l_1-grid was obtained in [3]). As a consequence, it is possible to decide in polynomial time if a finite metric space embeds into the l_1-plane. Edmonds [8] presented an $O(n^2 \log^2 n)$ time algorithm for this problem and very recently we learned that Eppstein [9] described an optimal $O(n^2)$ time algorithm. In this note, independently of [9], we describe a simple and optimal algorithm for this problem, which is different from that of [9].

In the sequel, we denote by d_1 or by $||\cdot||_1$ the l_1-metric and by d_∞ the l_∞-metric. A metric space (X, d) is *isometrically embeddable* into a host metric space (Y, d') if there exists a map $\varphi : X \to Y$ such that $d'(\varphi(x), \varphi(y)) = d(x, y)$ for all $x, y \in X$. In this case we say that X is a subspace of Y. The *(closed) ball* and the *sphere* of center x and radius r are the sets $B(x, r) = \{p \in X : d(x, p) \leq r\}$ and $S(x, r) = \{p \in X : d(x, p) = r\}$, respectively. The *interval* between two points x, y of X is the set $I(x, y) = \{z \in X : d(x, y) = d(x, z) + d(z, y)\}$. Any ball of $(\mathbb{R}^k, d_\infty)$ is an axis-parallel cube. A subset S of X is *gated* if for every point $x \in X$ there exists a (unique) point $x' \in S$, the *gate* of x in S, such that $x' \in I(x, y)$ for all $y \in S$ [7]. The intersection of gated sets is also gated. For a point p of $\mathbb{R}^2$, denote by $Q_1(p), \ldots, Q_4(p)$ the four quadrants of $\mathbb{R}^2$ defined by the vertical and horizontal lines passing via the point p and labeled counterclockwise. Any interval $I_1(x, y)$ of the rectilinear plane $(\mathbb{R}^2, d_1)$ is an axis-parallel rectangle which can be reduced to a horizontal or vertical segment. Any ball of $(\mathbb{R}^2, d_1)$ is a lozenge obtained from an axis-parallel square by a rotation by 45° degrees. In the rectilinear plane, any halfplane defined by a vertical or a horizontal line is gated. As a consequence, axis-parallel rectangles, quadrants, and strips of $(\mathbb{R}^2, d_1)$ are gated as intersections of such halfplanes.

2 Tight spans

A metric space (X, d) is called *hyperconvex* (or *injective*) [1, 10] if any family of closed balls $B(x_i, r_i)$ with centers x_i and radii r_i, $i \in I$, satisfying $d(x_i, x_j) \leq r_i + r_j$ for all $i, j \in I$ has a nonempty intersection, that is, (X, d) is a geodesic space such that the closed balls have the Helly property. As shown by Isbell [10] and Dress [6], for every metric space (X, d) there exists the smallest injective space $T(X)$ extending (X, d), referred to as the *injective hull* [10], or *tight span* [6] of (X, d). In general, tight spans are hard to visualize. Nevertheless, if $|X| \leq 5$, Dress [6] completely described $T(X)$ via the interpoint-distances of X. For example, if $|X| = 3$, then $T(X)$ consists of three line segments joined at a (Steiner) point, with the points of X at the ends of the arms. If $|X| = 4$, then the generic form of $T(X)$ is a rectangle $R(X)$ endowed with the l_1-metric, together with a line segment attached by one end to each corner of this rectangle.

*This research was partly supported by the ANR grant BLAN06-1-13889 (projet OPTICOMB).

†Laboratoire d'Informatique Fondamentale, Université d'Aix-Marseille, Faculté des Sciences de Luminy, F-13288 Marseille Cedex 9, France{nicolas.catusse, victor.chepoi, yann.vaxes}@lif.univ-mrs.fr

Finally, there are three canonical types of tight spans of 5-point metric spaces precisely described in [6]. The generic forms of $T(X)$ for $|X| \leq 5$ are illustrated in Fig. 1-2 of [4]. From the construction of tight spans of 3- and 4-point metric spaces immediately follows that any metric space (X, d) with at most 4 points and its tight span can be isometrically embedded into the l_1-plane. Note also that from the combinatorial characterization of finite metric subspaces of the l_1-plane presented in [2] follows that a tree-metric (X, d) is isometrically embeddable into the l_1-plane if and only if the tree-network $T(X)$ has at most four leaves. Finally since $(\mathbb{R}^2, d_1)$ is injective, by minimality property of tight spans, $T(X)$ is an isometric subspace of the l_1-plane for any finite subspace X of $\mathbb{R}^2$.

3 Algorithm and its correctness

Let (X, d) be a metric space with n points, called *terminals*. Set $X = \{x_1, \ldots, x_n\}$. Our algorithm first finds in $O(n^2)$ time a quadruplet P° of X whose tight span contains a nondegenerated rectangle $R(P^\circ)$. If such a quadruplet does not exists, then (X, d) is a tree-metric and $T(X)$ is a tree-network. If this tree-network contains more than four leaves, then (X, d) cannot be isometrically embedded into the l_1-plane, otherwise such an embedding can be easily derived. Given the required quadruplet P°, we consider any isometric embedding of P° and of its tight span into the l_1-plane as illustrated in Fig. 4 of [4] and partition the remaining points of X into groups depending on their location in the regions of the plane defined by $R(P^\circ)$ and the segments of $T(P^\circ)$. The exact location of points of X in these regions is uniquely determined except for the four quadrants defined by $R(P^\circ)$. At the second stage, we replace the quadruplet P° by another quadruplet P by picking one furthest from $R(P^\circ)$ point of X in each of these quadrants. We show that $R(P)$ is also nondegenerated, moreover, for any isometric embedding φ_0 of P and $T(P)$ into the l_1-plane, the quadrants defined by two opposite corners do not contains other terminals of X. Again the location of the points of X in all regions of the plane except the two opposite quadrants is uniquely determined. To compute the location of the remaining terminals in these two quadrants we adapt the second part of the algorithm of Edmonds [8]: we construct on these terminals a graph as in [8], partition it into connected components, separately determine the location of the points of each component, and then combine them into a single chain of components to obtain a global isometric embedding φ of (X, d) extending φ_0 or to decide that it does not exist.

3.1 Computing the quadruplet P°

For each $i = 1, \ldots, n$, set $X_i := \{x_1, \ldots, x_i\}$. We start by computing the tight span of the first four points of X. If this tight span is not degenerated then we return the quadruplet X_4 as P°. Now suppose that the tight span of the first $i-1$ points of X is a tree-network A_{i-1} with at most four leaves. This means that A_{i-1} contains one or two ramification points (which are not necessarily points of X) having degree at most 4, all remaining terminals of X_{i-1} are either leaves or vertices of degree two of A_{i-1}. We say that two terminals of X_{i-1} are consecutive in A_{i-1} if the segment connecting them in A_{i-1} does not contain other points of X_{i-1}. Note that A_{i-1} contains at most $n+4$ of consecutive pairs. For each pair x_j, x_k of consecutive terminals of X_{i-1} we compute the Gromov product $\alpha_{x_i} := (x_j, x_k)_{x_i} = 1/2(d(x_i, x_j) + d(x_i, x_k) - d(x_j, x_k))$ of x_i with $\{x_j, x_k\}$. Let $\{a, b\}$ be the pair of consecutive points of X_{i-1} minimizing the Gromov product $\alpha_{x_i} = (a, b)_{x_i}$. Let c be the point of the segment $[a, b]$ of A_{i-1} located at distance $\alpha_a := (b, x_i)_a$ from a and at distance $\alpha_b := (a, x_i)_b$ from b.

Denote by A_i the tree-network obtained from A_{i-1} by adding the segment $[x_i, c]$ of length α_{x_i}. By running Breadth-First-Search on A_i rooted at x_i, we check if $d_{A_i}(x_i, x_j) = d(x_i, x_j)$ for any terminal x_j of X_i. If this holds for all $x_j \in X_i$, then the tight span of X_i is the tree-network A_i. If A_i contains more than 4 leaves, then we return the answer "not" and the algorithm halts. Otherwise, if $i = n$, then we return the answer "yes" and an isometric embedding of X and its tight span A_n in the l_1-plane, else, if $i < n$, we consider the next point x_{i+1}. Finally, if x_j is the first point of X_i such that $d_{A_i}(x_i, x_j) \neq d(x_i, x_j)$, then *the tight span of the quadruplet* $\{a, b, x_i, x_j\}$ *is non-degenerated* (see [4] for a proof) *and we return it as* P°.

3.2 Classification of the points of X with respect to the rectangle of $T(P^\circ)$

Let $P^\circ = \{p_1^\circ, p_2^\circ, p_3^\circ, p_4^\circ\}$ be the quadruplet whose tight span $T(P^\circ)$ is non-degenerated. Let R° be one of the two possible isometric embeddings of the rectangle $R(P^\circ)$ of $T(P^\circ)$ and consider a complete or a partial isometric embedding of $T(P^\circ)$ such that $R(P^\circ)$ is embedded as R°. Denote by $Q_1^\circ, Q_2^\circ, Q_3^\circ, Q_4^\circ$ the four (closed) quadrants defined by the four consecutive corners $q_1^\circ, q_2^\circ, q_3^\circ, q_4^\circ$ of R° labeled such that the point p_i° must be located in $Q_i^\circ, i = 1, \ldots, 4$. Let also $S_1^\circ, S_2^\circ, S_3^\circ$, and S_4° be the remaining half-infinite strips. Since we know how to construct in constant time the tight span of a 5-point metric space, we can compute the distances from all terminals p of X to the corners of $R(P^\circ)$ or R° in total $O(n)$ time. Since R° is gated, from the distances of p to the corners of R° we can compute the gate of p in R°. Consequently, for each point $p \in X \setminus P^\circ$ we can decide in which of the nine regions of the plane belongs its location $\varphi(p)$ under any isometric embedding φ of (X, d) subject to the assumption that $R(P^\circ)$ is embedded as R°. If $\varphi(p)$ belongs to one of the four half-strips or to R°, then we can also find the exact location itself by using the gate

of p in R°. So, it remains to decide the locations of points assigned to $Q_1^\circ, Q_2^\circ, Q_3^\circ$, and Q_4°. For any point $p \in X$ which must be located in Q_i°, the set of possible locations of p is either empty (and no isometric embedding exists) or a segment s_p of Q_i° consisting of all points $z \in Q_i^\circ$ such that $\|z - q_i^\circ\|_1 = d(p, q_i^\circ)$.

Notice that for any quadruplet $P' = \{p'_1, p'_2, p'_3, p'_4\}$ of terminals such that p'_i is assigned to Q_i°, $i \in \{1, 2, 3, 4\}$, *the tight span $T(P')$ is also nondegenerated* (see [4] for a proof).

3.3 The quadruplet P and its properties

Let $P = \{p_1, p_2, p_3, p_4\}$ be the quadruplet of X, where p_i is a point of X which must be located in Q_i° and is maximally distant from the corner q_i° of R°. As we established above, the tight span of P is nondegenerated. As we also noticed, there exists a constant number of ways in which we can isometrically embed $T(P)$ into the l_1-plane. Further we proceed in the following way: we pick an arbitrary isometric embedding φ_0 of $T(P)$ and try to extend it to an isometric embedding φ of the whole metric space (X, d) in the l_1-plane. If this is possible for some embedding of $T(P)$, then the algorithm returns the answer "yes" and an isometric embedding of X, otherwise the algorithm returns the answer "not". Let R denote the image of $R(P)$ under φ_0.

We call a terminal p_i of P *fixed* by φ_0 if either $\varphi_0(p_i)$ is a corner of R or the segment of $T(P)$ incident to p_i is embedded by φ_0 as a horizontal or a vertical segment; else we call p_i *free*. The embedding of a free terminal p_i is not exactly determined but is restricted to a segment s_{p_i} consisting of the points of the quadrant defined by q_i having the same l_1-distance to q_i. We call the terminals $p_i, p_{i+1(\mathrm{mod}4)}$ *incident* and the terminals $p_i, p_{i+2(\mathrm{mod}4)}$ *opposite*. From the isometric embedding of $T(P)$ we conclude that at most one of two incident terminals can be free. Moreover, if a terminal p_i of P is fixed but is not a corner of R, then at least one of the two terminals incident to p_i is also fixed. Depending on the number of non-degenerate tips, we obtain a list of possible configurations that are listed in Fig. 4 of [4].

Denote by Π the smallest axis-parallel rectangle containing R and the fixed terminals of P; Fig. 5 of [4] illustrates Π for two cases from Fig. 4 of [4]. Let q_1, q_2, q_3, q_4 be the corners of Π labeled such that q_i is the corner of R corresponding to the point p_i and to the corner q_i° of R°. Denote by $Q_1, \ldots, Q_4$ the quadrants of $\mathbb{R}^2$ defined by the corners of Π and by $S_1, \ldots, S_4$ the remaining half-infinite strips. Again, as in the case of the quadruplet P°, by building the tight spans of $P \cup \{p\}$ for all terminals $p \in X \setminus P$, we can compute in total linear time the distances from all such points p to the corners of R (and to the corners of Π). From these four distances and the distances of p to the terminals of the quadruplet P we can determine in which of the nine regions $Q_1, Q_2, Q_3, Q_4, S_1, S_2, S_3, S_4, \Pi$ of the plane must be located p. Moreover, if p is assigned to Π or to one of the four half-strips S_1, S_2, S_3, S_4, then, in the region in which p assigned, the intersection of the four spheres centered at the terminals of P and having the distances from respective points to p as radii is either empty or a single point.

In [4], we prove the following property of the quadruplet P : *among the four quadrants Q_1, Q_2, Q_3, and Q_4 defined by P, two opposite quadrants, say Q_1 and Q_3, do not contain terminals of $X \setminus P$.*

3.4 Locating in the quadrants Q_1 and Q_3

We show now how to find the exact location of the set X_1 of terminals assigned to Q_1 (the set X_3 of terminals located in Q_3 is treated analogously). Independently of how the extension φ of φ_0 is chosen, for each terminal $u \in X_1$, the l_1-distance $\|\varphi(u) - q_1\|_1$ from the location of u to the corner q_1 of Π is one and the same, which we denote by Δ_u. Since q_1 lies between $\varphi(u)$ and $\varphi(p_i)$ for any $p_i \in P$, the value of Δ_u can be computed by setting $\Delta_u := d(u, p_1) - \|\varphi_0(p_1) - q_1\|_1$. Then the set of all possible locations $\varphi(u)$ of $u \in X_1$ is the *level segment* s_u which is the intersection of Q_1 with the sphere $S(q_1, \Delta_u)$ of radius Δ_u centered at q_1.

To compute the locations of the terminals of X_1 in the quadrant Q_1, we adapt to the l_1-plane the definition of a graph (which we denote by $G_1 = (X_1, E_1)$) defined by Edmonds [8] in the l_∞-plane. Two terminals $u, v \in X_1$ are adjacent in G_1 if and only if $d(u, v) > |\Delta_u - \Delta_v|$. Denote by $C_1, \ldots, C_k$ the connected components of the graph G_1. The following properties are established in Lemmata 3-5 of [8]:

(1) Each component C_i is *rigid,* i.e., once the location of any point $u \in C_i$ is fixed, the locations of the remaining points of C_i are also fixed (up to symmetry).

(2) The components $C_1, \ldots, C_k$ of the graph G_1 can be numbered so that the points of each C_i appear consecutively in the list of points $u \in X_1$ sorted in increasing order of their distances Δ_u to q_1;

(3) For a component C_i of G_1, let B_i be the smallest axis-parallel rectangle containing $\{\varphi_i(u) : u \in C_i\}$ for an isometric embedding φ_i of (C_i, d) in the l_1-plane. Let b_i be the upper right corner of B_i. Then the embedding of $C_1, \ldots, C_k$ preserves the distances between all pairs of points lying in different components if and only if for every pair of consecutive components C_i and C_{i+1}, the box B_{i+1} lies entirely in $Q_1(b_i)$.

The location in the quadrant Q_1 of some terminals of X_1 can be fixed by terminals already located in the two half-strips incident to Q_1. We say that a terminal $u \in X_1$ is *fixed by a terminal p* already located in $S_1 \cup S_4$ if the intersection of the segment s_u with the sphere $S(\varphi(p), d(p, u))$ is a single point. If $u \in X_1$ is fixed by a terminal located in S_1, then

u is also fixed by the upmost terminal p^* located in this half-strip. Analogously, if $u \in X_1$ is fixed by a terminal of S_4, then u is also fixed by the rightmost terminal p_* located in S_4. By considering the intersections of the segments $s_u, u \in X_1$, with the spheres $S(\varphi(p^*), d(p^*, u))$ and $S(\varphi(p_*), d(p_*, u))$ we can decide in linear time which terminals of X_1 are fixed by p^* and p_* and find their location in Q_1 (for an illustration, see Fig. 7 of [4]). According to property (1), if a terminal of a connected component of G_1 is fixed, then the location of the whole component is also fixed (up to symmetry). Let C_j be the connected component of G_1 containing the furthest from q_1 terminal $u \in X_1$ fixed by p^* or p_*, say by p^*. In [4], we prove that *all terminals of* $C_1, \ldots, C_{j-1}$ *are also fixed by* p^*.

It remains to locate in Q_1 the terminals of the components $C_{j+1}, C_{j+2}, \ldots, C_k$. We compute separately an isometric embedding of each component C_i for $i = j+1, \ldots, k$. For this, we fix arbitrarily the location of the first two points u, v of C_i in the segments s_u and s_v so that to preserve the distance $d(u, v)$ (the terminals of C_i are ordered by their distances to q_1). By property (1) of [8], the location of the remaining points of C_i is uniquely determined and each point w of C_i will be located in its level segment s_w. Let φ_i be the resulting embedding of C_i. Denote by B_i the smallest axis-parallel rectangle (alias *box*) containing the image $\varphi_i(C_i)$ of C_i. Let a_i and b_i denote the lower left and the upper right corners of B_i. Note that a_i belongs to the l_1-interval between q_1 and the image $\varphi_i(u)$ of any terminal u of C_i, while the l_1-interval between q_1 and b_i contains the images of all terminals of C_i. Therefore if we set $\Delta_{a_i} := \Delta_u - \|a_i - \varphi_i(u)\|_1$ and $\Delta_{b_i} := \Delta_u + \|\varphi(u) - b_i\|_1$, where u is any terminal of C_i, then in all isometric embeddings of (C_i, d) in which all terminals $u \in C_i$ are located on s_u, the points a_i and b_i must be located on the level segments s_{a_i} and s_{b_i}, defined as the intersections of the quadrant Q_1 with the spheres $S(q_1, \Delta_{a_i})$ and $S(q_1, \Delta_{b_i})$.

By properties (2) and (3) of [8], to define a single isometric embedding of the components $C_{j+1}, \ldots, C_k$ we need to assemble the boxes $B_{j+1}, \ldots, B_k$ such that *for two consecutive components* C_i *and* C_{i+1}, *the box* B_{i+1} *lies entirely in the quadrant* $Q_1(b_i)$. In [4], we prove that this is possible if and only if *for each pair of consecutive boxes* B_i, B_{i+1}, $i = j, j+1, \ldots, k-1$, *the inequality* $\Delta_{b_i} \leq \Delta_{a_{i+1}}$ *holds*. This local condition depends only on the values of $\Delta_{a_i}, \Delta_{b_i}$ and is independent of the actual location of the boxes $B_i, i = 1, \ldots, k$. As a result, the algorithm that embeds the boxes $B_{j+1}, \ldots, B_k$ is very simple. For each $i = j, \ldots, k-1$, we compute the box B_{i+1} and the values of $\Delta_{a_{i+1}}$ and $\Delta_{b_{i+1}}$. If $\Delta_{a_{i+1}} < \Delta_{b_i}$ for some i, then return the answer "there is no isometric embedding of (X, d) extending φ_0 of $T(P)$". Otherwise, having already located the box B_i, by what has been shown above, the intersection of $Q_1(b_i)$ with the level segment $s_{a_{i+1}}$ is non-empty. Therefore we can translate B_{j+1} such that its lower left corner a_{i+1} becomes a point of this intersection.

In this way, we obtain an embedding of $C_{j+1}, \ldots, C_k$ and $B_{j+1}, \ldots, B_k$ satisfying the conditions (1)-(3), thus an isometric embedding of the metric space $(\bigcup_{i=j+1}^{k} C_i, d)$ in Q_1. Analogously, by constructing the graph $G_3 = (X_3, E_3)$ and its components, either we obtain a negative answer or we return an isometric embedding of the metric space defined by the non-fixed components of G_3 in Q_3. Denote by φ the extension of φ_0 obtained by the method described above. In $O(n^2)$ we test if φ is an isometric embedding of (X, d) into the l_1-plane. If the answer is negative, then we return "there is no isometric embedding of (X, d) extending φ_0", otherwise we return φ as an isometric embedding. The algorithm returns the global answer "not" if for all possible embeddings φ_0 of $T(P)$ it returns the negative answer. From what we established follows that in this case (X, d) is not isometrically embeddable into the l_1-plane.

Summarizing, we obtain our main result (see [4] for complete proof and complexity analysis):

Theorem 1 *For a metric space* (X, d) *on* n *points, it is possible to decide in optimal* $O(n^2)$ *time if* (X, d) *is isometrically embeddable into the* l_1*-plane and to find such an embedding if it exists.*

References

[1] N. Aronszajn and P. Panitchpakdi, Extensions of uniformly continuous transformations and hyperconvex metric spaces, *Pacific J. Math.* **6** (1956), 405–439.

[2] H.–J. Bandelt and V. Chepoi, Embedding metric spaces in the rectilinear plane: a six–point criterion, *Discr. Comput. Geom.* **15** (1996) 107–117.

[3] H.-J. Bandelt, V. Chepoi, Embedding into the rectilinear grid, *Networks* **32** (1998), 127–132.

[4] N. Catusse, V. Chepoi and Y. Vaxès, Embedding into the rectilinear plane in optimal $O(n^2)$ time, arXiv:0910.1059 (2009).

[5] M. Deza and M. Laurent, *Geometry of Cuts and Metrics,* Springer-Verlag, Berlin, 1997.

[6] A.W.M. Dress, Trees, tight extensions of metric spaces, and the cohomological dimension of certain groups, *Adv. Math.* **53** (1984), 321–402.

[7] A. Dress and R. Scharlau, Gated sets in metric spaces, *Aequat. Math.* **34** (1987), 112–120.

[8] J. Edmonds. Embedding into l_∞^2 is easy, embedding into l_∞^3 is NP-complete, *Discr. Comput. Geom.* **39** (2008), 747–765.

[9] D. Eppstein, Optimally fast incremental Manhattan plane embedding and planar tight span construction, arXiv:0909.1866v1, (2009).

[10] J. Isbell, Six theorems about metric spaces, *Comment. Math. Helv.* **39** (1964), 65–74.

[11] S.M. Malitz and J.I. Malitz, A bounded compactness theorem for L^1–embeddability of metric spaces in the plane, *Discr. Comput. Geom.* **8** (1992) 373–385.

Hide-and-Seek: A Linear Time Algorithm for Polygon Walk Problems

Atlas F. Cook IV* Chenglin Fan† Jun Luo†

Abstract

Jack and Jill have decided to play hide-and-seek along the boundary of a simple polygon. To start the game, Jack and Jill each choose an arbitrary path on this boundary. After fixing these paths, our goal is to determine whether Jack can control his speed such that he walks along his path from beginning to end without being seen by Jill. We solve this problem with a linear-sized *skeleton visibility diagram* that implicitly represents visibility between pairs of points on the boundary of the simple polygon. Note that this data structure has applications for any *polygon walk problem* where one entity wishes to remain hidden throughout a traversal of some path.

1 Introduction

Let P be a simple polygon with n vertices, and let ∂P be the boundary of P. Jack and Jill want to play hide-and-seek along ∂P. To start the game, Jack and Jill each choose fixed paths on ∂P. Both children have excellent eyesight and can see infinitely far inside P but cannot see past the opaque boundary of ∂P. Jack wants to control his speed on ∂P so that he travels on a *safe path* that keeps him hidden from Jill at all times. We show how to compute such a safe path in linear time using a *skeleton visibility diagram* that implicitly represents visibility between pairs of points on ∂P.

We refer to any problem where one or more entities walk along ∂P while maintaining certain properties as a *polygon walk problem*. Related work by Icking and Klein [4] uses two guards to patrol the boundary of a simple polygon from a single entrance point until they reach a single exit point. The two guards walk in opposite directions on this boundary and always maintain mutual visibility for security purposes. Given a fixed entrance point, their $O(n \log n)$ time approach determines whether an intruder can always be detected by the two guards. Here, n is the number of vertices of the simple polygon. Subsequently, Heffernan [3] improved this algorithm to optimal $O(n)$ time. Additional work by Zhang and Kameda [6] has developed an $O(n)$ time algorithm to detect all possible entrance points on the boundary that permit the two guards to detect an intruder.

Polygon walk problems with varying numbers of guards have also been studied. LaValle et al. [5] show how to search a polygonal region with a single guard in $O(n^2)$ time. Efrat et al. [1] sweep a polygonal chain of guards along the boundary of a simple polygon in $O(n^3)$ time. Each consecutive pair of guards in the chain are always mutually visible, and the goal is to detect any intruder using the minimum number of guards.

Note that previous research typically requires two or more guards to walk along the boundary of a simple polygon while maintaining mutual visibility inside a simple polygon. Our problem is different because we have two entities walk along the boundary while maintaining mutual *invisibility*.

2 Preliminaries

Given a simple polygon P, we say that two points p and q are *mutually visible* if the line segment $\overline{pq} \in P \cup \partial P$; otherwise, p and q are *mutually invisible*. We define a *configuration* $\langle p, q \rangle \in \partial P \times \partial P$ by a pair of points on the boundary of P [5]. We call such a configuration *safe* when p and q are mutually invisible and *unsafe* otherwise. We pick an arbitrary point on the boundary ∂P as the origin, and we measure all distances along ∂P in a clockwise fashion from this origin. Let $|\partial P|$ denote the total length of ∂P. Let x_1, x_2, y_1, y_2 be points on ∂P such that Jack travels from x_1 to x_2 and Jill travels from y_1 to y_2 (see Figure 1a). Define a two-dimensional configuration space such that the x-axis represents the position of Jack on ∂P and the y-axis represents the position of Jill on ∂P (see Figure 1b). Observe that if Jack and Jill are located at the same point on ∂P, then they must be mutually visible. Consequently, every point on the lines $y = x$ and $y = x - |\partial P|$ in this configuration space is always unsafe. This permits us to prune our search space to the infinite area between and including the lines $y = x$ and $y = x - |\partial P|$. This infinite area is referred to as the *visibility space*. We shade all safe configurations in the visibility space a gray color and call this refined space a *visibility diagram*. See Figure 1b.

We now describe all safe configurations in the visibility diagram as a discrete set of shapes. A *reflex vertex* is a vertex $r \in P$ whose interior angle inside P is more than π. For example, vertices 2, 5, 8, and 11 in Figure 1a are reflex vertices. Each reflex vertex of P defines one gray shape in the visibility diagram. We call each gray shape a *cell*.

Notice that the boundary of each cell in Figure 1b contains one horizontal line segment and one vertical line segment such that these line segments intersect on either

*Department of Mathematics and Computing Science, TU Eindhoven, Netherlands, a.f.cook@tue.nl

†Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China, {cl.fan,jun.luo}@sub.siat.ac.cn

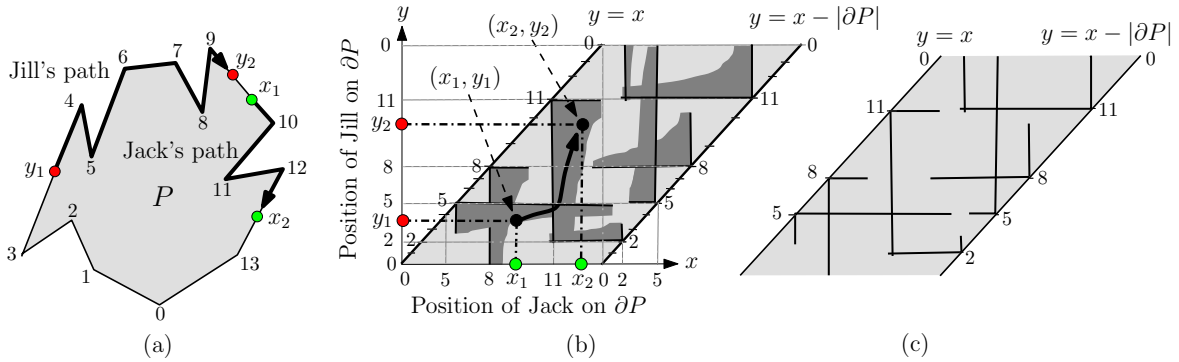


Figure 1: (a) Simple polygon P, (b) Visibility diagram of P, (c) Skeleton visibility diagram of P

$y = x$ or $y = x - |\partial P|$. We define the *skeleton* of each cell as the union of these two line segments. Due to visibility constraints in a simple polygon, the corner point of every skeleton corresponds to a reflex vertex of P. We define the *skeleton visibility diagram* as the union of all skeletons in the visibility diagram (see Figure 1c).

3 The Boundary of a Cell

The following notation will be useful for describing a cell in the visibility diagram. The boundary of P is a clockwise sequence of vertices. The vertices immediately preceding and succeeding a vertex r on ∂P are denoted by $Pred(r)$ and $Succ(r)$, respectively. For any two points $a, b \in \partial P$, the open and closed portions of ∂P from a to b in clockwise order are denoted by $\partial P(a, b)$ and $\partial P[a, b]$.

Figure 2 shows the cell for a reflex vertex $r \in P$. Let $B(r) \in \partial P$ be the *backward extension point* where the extension of the edge from $Succ(r)$ to r leaves P for the first time. Similarly, let $F(r)$ be the *forward extension point* where the extension of the edge from $Pred(r)$ to r leaves P for the first time. The cell for r is composed of two

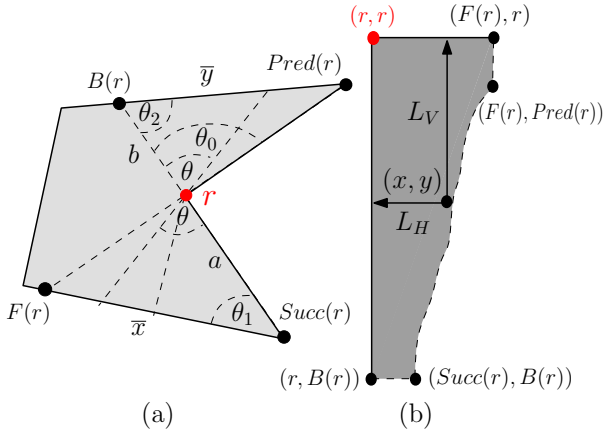


Figure 2: (a) A reflex vertex $r \in P$ defines (b) a single cell in the visibility diagram. The functions $\overline{x}$ and $\overline{y}$ define the clockwise portion of the cell boundary from $(F(r), Pred(r))$ to $(Succ(r), B(r))$.

horizontal line segments, two vertical line segments, and a monotone set of curves. We describe the set of curves by two piecewise monotone functions $\overline{x} = \frac{a \sin\theta}{\sin(\theta+\theta_1)}$ and $\overline{y} = \frac{b \sin\theta}{\sin(\theta+\theta_2)}$. As illustrated in Figure 2, a is the length of the edge from r to $Succ(r)$, and b is the length of the edge from r to $B(r)$. Define θ_0 as the angle formed by $Pred(r)$, r, and $B(r)$. Let θ_1 be the angle formed by r, $Succ(r)$, and $Succ(Succ(r))$. Define θ_2 as the angle formed by r, $B(r)$, and $Pred(r)$. Finally, let θ be a parameter in $[0, \theta_0]$.

In order to determine the slope of these curves, we compute the derivative $\frac{d\overline{y}}{d\overline{x}} = \frac{b \csc(\theta+\theta_2)(\cot\theta - \cot(\theta+\theta_2))}{a \csc(\theta+\theta_1)(\cot\theta - \cot(\theta+\theta_1))} \geq 0$. Here, $0 \leq \theta \leq \pi$, $0 \leq (\theta+\theta_1) \leq \pi$, and $0 \leq (\theta+\theta_2) \leq \pi$. Since this derivative is always positive, the curve is monotone. This means that the horizontal line segment L_H from any point (x, y) in a cell to the vertical skeleton of the cell is completely contained inside the cell. Similarly, the vertical line segment L_V from any point (x, y) in a cell to the horizontal skeleton of the cell is contained inside the cell.

4 Properties of the Skeleton Visibility Diagram

Figure 3 illustrates the four types of intersections between a pair of cells.

Lemma 1 *Two cells intersect if and only if their skeletons intersect.*

Proof. If two skeletons intersect, then their cells must also intersect because each skeleton is a subset of its associated cell. We now prove that if two cells intersect, then their skeletons must also intersect. We prove this for the two types of intersections depicted in Figures 3a and 3c. The proof for the other two types is similar.

Consider the type of intersection in Figure 3a. Figure 4a illustrates two possible arrangements such that a pair of cells intersect but their skeletons do not intersect. However, we know that the top left corners of the two cells must lie on the line $y = x$ as shown in Figure 4b. Therefore, the two arrangements in Figure 4a cannot occur.

Now consider the type of intersection in Figure 3c. Figure 4c illustrates two possible arrangements such that a pair of cells intersect but their skeletons do not intersect.

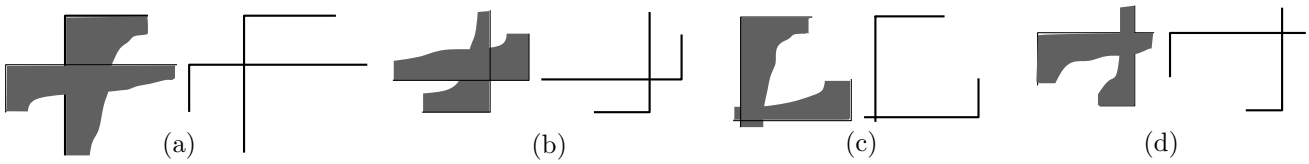


Figure 3: There are four types of intersections between a pair of cells. The corresponding skeletons are also shown.

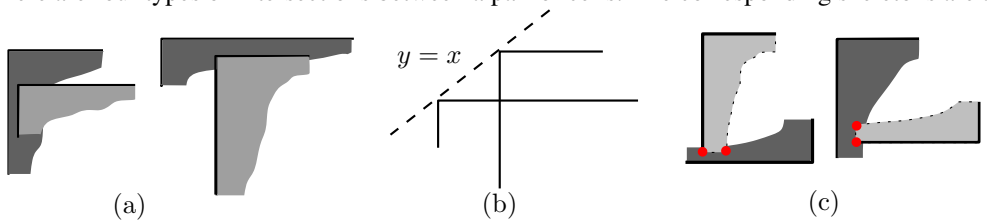


Figure 4: (a,b) Two arbitrary cells could intersect without having intersecting skeletons; however, if the top-left corners of both cells lie on $y = x$, then the cells only intersect when their skeletons intersect. (c) If one cell touches $y = x$ and one cell touches $y = x - |\partial P|$, then the only way for the two cells to intersect without having intersecting skeletons would be for a vertex that bounds a horizontal or vertical line segment to lie in the interior of a cell. This cannot occur.

Neither of these scenarios can occur because it is not possible for a vertex that bounds a horizontal or vertical line segment of a cell to be contained inside a second cell. □

We call each connected set of gray points in the visibility diagram a *unit*. Thus, each unit is the union of one or more intersecting cells. Recall from Figure 1 that the start and end points of Jack are $x_1, x_2 \in \partial P$, and the start and end points of Jill are $y_1, y_2 \in \partial P$. Let $s = (x_1, y_1)$ and $t = (x_2, y_2)$ be points in the visibility diagram. We define a *safe path* $\pi_{\text{safe}}(s, t)$ as a path that connects the points s and t in the visibility diagram such that the entire path is in the gray area (see Figure 1b). Note that Jack and Jill can traverse their entire paths without seeing each other if and only if a safe path $\pi_{\text{safe}}(s, t)$ exists.

5 Computing a Safe Path for Jack and Jill

Recall that a unit is a connected set of safe points. Consequently, a safe path $\pi_{\text{safe}}(s, t)$ exists if and only if s and t lie in the same unit. To decide whether s and t lie in the same unit, we determine a cell that contains s and a cell that contains t. We then decide whether these two cells are in the same unit.

5.1 Determining Cells for s and t

Recall that a pair of points $x, y \in \partial P$ define a single point (x, y) in the visibility diagram. Our task is to determine one cell in the visibility diagram that contains (x, y). To simplify this process, we define the visibility polygon $Vis(P, x)$ as the set of all points in P that are directly visible from x (see Figure 5).

Lemma 2 *Given any fixed point (x, y) in the visibility diagram, a cell that contains (x, y) can be determined in $O(n)$ time.*

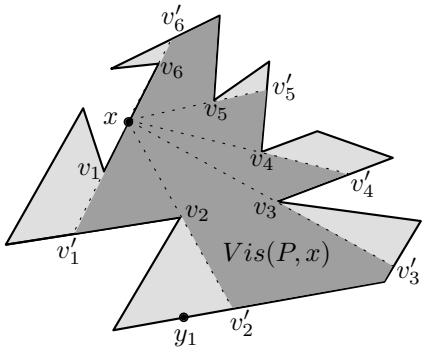


Figure 5: A visibility polygon $Vis(P, x)$ with reflex vertices $v_1, ..., v_6$. Each v'_i is the first point where the ray from x through v_i leaves P.

Proof. Compute the visibility polygon $Vis(P, x)$ in $O(n)$ time [2] (see Figure 5). Traverse the boundary of P and determine the maximally connected interval of ∂P that is *not* directly visible from x and also contains y. If no such interval exists, then no cell contains (x, y). Otherwise, one endpoint of this interval must be a reflex vertex r, and the cell associated with r must contain (x, y). □

Lemma 2 implies that we can calculate a cell that contains $s = (x_1, y_1)$ and a cell that contains $t = (x_2, y_2)$ in $O(n)$ time.

5.2 Determining if Two Cells Are in One Unit

To determine whether two cells are in the same unit, we associate each cell's *skeleton* with the unit that contains it. Figure 6 illustrates the process of constructing units of skeletons. For the moment, we ignore the vertical line segments of all skeletons that touch $y = x - |\partial P|$ (see Figure 6). We scan all skeletons in the visibility diagram in decreasing order according to the y coordinates of their

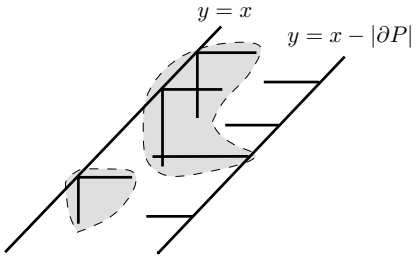


Figure 6: Units of skeletons are partially computed by only considering the horizontal line segments of skeletons that touch $y = x - |\partial P|$.

corner vertices. During this scan, we maintain two sets. Set U is the desired list of units that are being constructed. Each unit in U is a list of skeletons that are ordered by the y coordinates of their corner vertices. Set W is a "wavefront" of units that only includes skeletons that could possibly intersect the not-yet-processed skeletons in the visibility diagram. The scan proceeds as follows.

If the current skeleton $\mathcal{S}$ has its corner vertex on the line $y = x$, then we scan the skeletons in the current wavefront W from bottom to top. If there is no skeleton in the wavefront W that intersects $\mathcal{S}$, then we create a new unit for the current skeleton $\mathcal{S}$ in both W and U. Otherwise, $\mathcal{S}$ intersects a sequence of consecutive units in W. We merge all (linked-list) units in W that intersect $\mathcal{S}$ and add $\mathcal{S}$ to the front of this merged unit. We also update U in an identical manner. Each skeleton in W whose vertical line segment does *not* intersect the horizontal line segment of $\mathcal{S}$ can be deleted from the wavefront W because its vertical line segment will never intersect the horizontal line segment of any skeleton below $\mathcal{S}$. The scan of the wavefront W stops when the vertical line segment of the current skeleton in W lies entirely to the right of $\mathcal{S}$.

If the current skeleton $\mathcal{S}$ has its corner vertex on the line $y = x - |\partial P|$, then we scan the units in the current wavefront W from top to bottom. The rest of the algorithm is analogous to the previous paragraph.

Lemma 3 *Assuming that we ignore the vertical line segments of all skeletons that touch $y = x - |\partial P|$, the units of all skeletons in the visibility diagram can be calculated in $O(n)$ time.*

Proof. Each skeleton is inserted into the sets U and W exactly once and is deleted from the wavefront W at most once. Although the wavefront W is scanned $O(n)$ times, each skeleton in W will be examined at most twice during normal operation (once when it is at the head of its unit and once when it is at the tail of its unit). Additionally, the algorithm will examine a total of $O(n)$ skeletons in order to stop the scans of the wavefront. □

We also perform a symmetric operation where units of skeletons are calculating without considering the vertical line segments of the skeletons that touch the line $y = x$. The true units of skeletons are then obtained by merging the sets U from these two operations.

Theorem 4 *Suppose Jack and Jill walk along fixed paths on the boundary of a simple polygon. We can decide whether the two children can possibly traverse their paths without ever seeing each other in linear time. An existing traversal can then be returned in output-sensitive time.*

Proof. Compute the visibility diagram for the simple polygon, and compute the units for the skeletons. Associate the start points of these paths with a unit in the visibility diagram. Associate the end points of these paths with a second unit in the visibility diagram. If these two units are the same, then a rectilinear traversal can be returned through the skeletons in this unit. □

6 Conclusion

Suppose Jack and Jill walk along arbitrary paths on the boundary of a simple polygon. We can determine whether these two children can possibly traverse their paths without ever seeing each other in linear time. We use linear-sized structures to implicitly represent visibility between pairs of points on the boundary of a simple polygon. These structures have applications for any *polygon walk problem* where one entity wishes to remain hidden at all times during a traversal of some path.

References

[1] A. Efrat, L. J. Guibas, S. Har-Peled, D. C. Lin, J. S. B. Mitchell, and T. M. Murali. Sweeping simple polygons with a chain of guards. *11th Symposium on Discrete Algorithms (SODA)*, pages 927–936, 2000.

[2] L. J. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. E. Tarjan. Linear time algorithms for visibility and shortest path problems inside simple polygons. *2nd Symposium on Computational Geometry (SoCG)*, pages 1–13, 1986.

[3] P. J. Heffernan. An optimal algorithm for the two-guard problem. *9th Symposium on Computaitonal Geometry (SoCG)*, pages 348–358, 1993.

[4] C. Icking and R. Klein. The two guards problem. *7th Symposium on Computational Geometry (SoCG)*, pages 166–175, 1991.

[5] S. M. LaValle, B. H. Simov, and G. Slutzki. An algorithm for searching a polygonal region with a flashlight. *16th Symposium on Computational Geometry (SoCG)*, pages 260–269, 2000.

[6] J. Z. Zhang and T. Kameda. A linear-time algorithm for finding all door locations that make a room searchable. *Theory and Applications of Models of Computation (TAMC)*, pages 502–513, 2008.

A Reactive-Agent Based Approach for a Facility Location Problem Using Dynamic Additively Weighted Voronoi Diagram

Arman Didandeh, Mehdi Khosravian, Bahram Sadeghi Bigham *

Abstract

We consider solving a newly mentioned hybrid class of Facility Location problems concerning about locating a set of attentive and reactive facilities on the problem plane with respect to a set of dynamic demands. The policy of assigning this set of facilities should satisfy the objective that our system minimizes on a total loss function after the system goes into a steady state. We propose an approach called RA-DAWV which is based on Additively Weighted Voronoi diagram in a continuous space and models the problem in discrete-time, while we try to assign the attentive-reactive facilities in each time cycle. We express our solution on a real world problem of fire in a location of points with different importance levels and we show the results of this expression as a case study. At the end of the paper, we show that the time complexity of the proposed algorithm, considering n, c and p to be the number of demand points, the number of available facilities and the number of time cycles, respectively, would be $O(c(n^2+p)logn)$. This includes the AWVD depiction and also the Point Location. We also provide a novel NP solution for the decision making process of plane Voronoi sectioning.

1 Introduction

Facility location is the problem of positioning some facilities on a problem plane in order to optimize one or several objectives or in general a demand satisfaction. In this section, we describe a bit the approaches on Facility Location problem and we also introduce Voronoi diagrams and reactive agents, as we use them in the next sections to solve the main problem.

1.1 Facility Location Problem

Facility Location problems are known to be NP-hard problems among computer scientists [1]. While no generic solution is available for this problem, Facility Location problems have been categorized into several known and specific classes and solutions are made specifically to cover the needs defined for these categories. The main and most important categories are: Set Covering Facility Location Problem (SCFLP), Maximum Covering Facility Location Problem (MCFLP), p-Center and p-Median. [1](modified). There are also several other categories like dynamic location problems, stochastic location problems and also multi-objective location problems [1]. Of all the methods represented for all these Facility Location problem categories, two general views are of most interest to the authors of this paper, which are the Voronoi diagram [5, 6, 7] view and the reactive agents view. The basic tool of all in the former are Voronoi diagrams and are described in detail in the coming subsections 1.2 and 1.3. Of the latter is [1] that looks at the facilities as attentive objects, known as reactive agents. This assumption have advantages like being well suited for the dynamic problems and good for distributed problems with several evolving/moving entities, cooperating to perform collective and local goals. The key idea here was agents are attracted to demands.

1.2 Voronoi Diagrams

One of the most well studied structures in computational geometry (CG) is the Voronoi diagram for a set of sites [6]. There have been various generalizations of the standard Euclidean Voronoi diagram, including generalizations to metrics, convex distance functions, the power distance, which yields the Power diagram, Polar diagram [5] and others. The sites considered include points, convex polygons, line segments, circles and more general smooth convex objects.
While Voronoi diagrams are useful and powerful tools to prepare rational solutions in several problems, attempts have been made to converse them thoroughly. Discussions about Voronoi diagrams usually split them into static and dynamic diagrams. Static Voronoi diagrams mostly gather the information of the problem plane, i.e. the location of the sites, and divide the plane into cells. On the other hand, dynamic Voronoi diagrams also model alterations in the problem plane, usually concerning changes in the size or weight of the sites, or addition/deletion of them. One of the most valuable attempts made to discuss the Dynamic Voronoi diagrams is [7].

*Department of Computer Science and IT, Institute for Advanced Studies in Basic science, Zanjan, Iran {a_didandeh, m_khosravian, b_sadeghi_b}@iasbs.ac.ir

1.3 AWVD

One of the generalizations is the Additively Weighted Voronoi diagram or, in short, AWVD. If the weights are positive, AWVD can be viewed geometrically as the Voronoi diagram for a set of circles, the centers of which are the points and the radii of which are the corresponding weights [2]. [9] presents a sweepline algorithm for a set of circular sites that compute their Voronoi diagrams in a 2D Euclidean space. [9] considers the radius for each site to be non-negative and the radii should not be identical. Also in [2], Karavelas et al. have proposed an algorithm for the construction of the Dynamic Additively Weighted Voronoi diagram of a set of weighted points on the plane. The novelty in their approach is that they use the dual of the additively weighted Voronoi diagram to represent it. The time complexity of AWVD will be addressed in Section 5.

1.4 Reactive Agents

A good definition for a "reactive agent" is: "a simple agent which reacts to stimuli and provides a simple service to a user". As it comes in [4], a simple reactive agent (also known as a reflex agent) is a production system where inputs from the environment are compared with rules to determine which actions to carry out. We take the advantages of this kind of agents to model the facilities of our system to have a simple architecture and an implemented intelligence.

The rest of this paper is structured as follows: Section 2 gives an insight of the problem and its statements here. Then Section 3 talks about the RA-DAWV approach. We continue our ideas with a case study on forest fire in Section 4. Section 5 concludes this paper by discussing our approach and some future work.

2 Problem Statement

In this section we take a look at what exactly is the problem we are trying to solve. Given a collection of facilities represented by a set P and sitting on the problem plane, we rule them to move towards the location of the demands of the system as they call them. Active demands in the problem plane generate a certain radial value of loss on the space. The problem we define here is discrete-time continuous-space. We call every time slice a cycle here and at the beginning of each cycle j , we assign a set $Q_{i,j} \subseteq P$ to the demand S_i in order to minimize the loss function of the whole system. Better to say, the total loss function is memory-measured cycle by cycle. This value is of our concern when we study the demands and try to assign attentive facilities to them. The problem is an extended version of the p-Median class with respect to some differences. We focus on the best policy to assign facilities to demands, in order to have less loss in our system. We assume that the problem plane is a real world 2D space. We imagine a convex hull containing our demands in it and also a bounding box holding this convex hull inside. This assumption makes us be able to use RA-DAWV approach to come to a policy of assignments.

3 RA-DAWV Approach

After getting the insight on the problem, here we mention our ideas on the solution. To have a good real-time policy based on online data from the environment, we have chosen a process of solution based on Dynamic AWVDs. Hence we try to have an AWVD inside the problem bounding box, and update this diagram according to the status of the system, regarding to the facility assignments and Voronoi sites' topology. We consider our attentive facilities to be reactive agents. This assumption is simply modeled with the help of a gravity function for any demand, as will be described later. Any reactive agent is a member of the pre-said P set and is represented by p_i. In any cycle, p_i is assigned to one and only one demand.
We also assume each demand to be a weighted Voronoi site, a circle with a certain center $C_i(x_i, y_i)$ on continuous-space and a certain radius r_i that shows the activation range of that demand. Every site also has a loss memory, shown by another radius R_i that shows an upper approximation for how much each demand has caused damage. It models the possibility of intensity re-growth for any demand with active facility assignment. For any demand S_i, its R_i property is set due to the maximum value of its r_i property since its birth time till present time. So the formal representation of any demand in our system is as an ordered 3-tuple demonstrated by: $S_i =< C_i(x_i, y_i), r_i, R_i >$.
Without loss of generality, we can approximately model the decision process in discrete-time.
Here we bring this idea into field that whenever a demand is born in the system -which is represented as a circle on the plane with a known geometrical topology-, it generates a gravity function. The value of this gravity depends on the intensity of the demand, the number of active attentive reactive agents that are assigned to this certain demand and also its position on the problem plane. To be more exact on the latter, we define a value for any point on the problem plane and consider the locus of the points that have similar values to be subset sections of the above plane. These sections have the same value for us when deciding about the assignment as we have mentioned above in Section 2. Hence we have broken the problem into cycles and the beginning of each cycle can be thought as the beginning of a new problem -with a total loss

value-, we desire to spread a subset of facilities on the problem plane like iron powder. So the demands that have more gravity values act like magnet and grab more facilities. These grabbed facilities will satisfy the demands and serve them during the cycle time. We also assume agents to have **Instant Transformation Ability** property in order to move instantly to their mission point in the between-cycles time gap. Now and with all these matters of problem, we go in detail of our solution. Our presented approach aims to solve the problem by sectioning the plane into Voronoi cells with Voronoi sites being the active demands in the under-study cycle. This sectioning is done using a decision function, known as the $f_{desicion}$. To see how this function helps out the sectioning, we first need to have a naive estimation of the intensity value of any active demand on the subsequent cycle. For this purpose, we define a **velocity of growth** for any site S_i as: $\alpha_i^j = (r_i^j - r_i^{j-1})/(t^j - t^{j-1})$. Now the estimated intensity value for the subsequent active demands would be as: $r_i^{j+1} = r_i^j + \alpha_i^j(t^{j+1} - t^j)$. This estimated naive intensity is a radius for site S_i in t^{j+1}, though making the "approximated restricted area" of S_i as: $H_i^{j+1} = \Pi(r_i^{j+1})^2$. It is obvious that α_i^j can model all 3 classes of demands: the under-service demands by having a negative value, the newly born demands by being zero, and the less important/very intense demands by taking a positive value.
In the next step, we take advantage of the approximated restricted area of all demands to calculate a set of values known as $f_{pred}^{i,j+1}$. This set is defined for any point A with a weight w_A as:

$$f_{pred}^{i,j+1} = \oint_{{H_i}^{j+1}} w_A \, dx \, dy = \int \int_{r_i^j}^{r_i^{j+1}} w_A \, r dr \, d\theta$$

After having these sets for points of our plane, we turn them into the gravity values known as: $g_{i,j}$. This helps us assign any point A to the definite site that its intensity of demand will reach sooner to A. Several versions of gravity may come in handy, regarding to the application domains we wish to implement them in. Here we mention two of them:

$$g : \mathbb{R} * (Q \subseteq P) \longrightarrow \mathbb{R}$$

$$g_{i,j}^{(1)} = \frac{e^{f_{pred}^{j+1}}}{|Q_{i,j}|} \qquad g_{i,j}^{(2)} = {f_{pred}^{j+1}}^{\frac{|P|}{|Q_{i,j}|}}$$

The fourth step of our solution is called the **radii estimation** process. In the radii-estimation step of the RA-DAWV, we utilize the gravity values for each Voronoi site to estimate new values for the next cycle site radii. These radii are known as the subsequent cycle information, as they are the most influential piece of data needed to make the next cycle's decisions. This is done using the formula: $r(new)_i^{j+1} = 2\pi(g_{i,j}^{5/2})$.

Using the estimated radii, we simply determine a new area of $H_i^{j+1}(new) = \Pi(r(new)_i^{j+1})^2$. This area is more precise on the intensity of the subsequent demands in the coming cycle. Now and according to [9], we have a set of circle Voronoi sites, which can help section the plane into Voronoi cells in a polynomial time.
Another approach, using the estimated radii could be a novel, while NP approach. Here we determine a function $\delta_A^{i,j}$ which helps us decide about the sectioning the plane: $\delta_A^{i,j} = e^{g_{i,j}}/2\Pi(d(A, C_i) - r(new)_i^{j+1})$ The value $d(A, C_k)$ is simply the Euclidean distance of the point A and the point C_k which is known as the center of site S_k.
Now we have the information needed for decision making:

$$f_{decision} : \mathbb{R}^2 * (B \subseteq \mathbb{N}) \longrightarrow \mathbb{N}$$

$$f_{decision}^{j+1} = \begin{cases} k & d(A, C_k) \leq r_k \\ argmax(\delta_A^{i,j}) & otherwise \end{cases}$$

At the end of this section, and now that we have discussed the RA-DAWV approach, we would like to address another concept of our system, known as the "total loss function". The total loss function of the system is a function that calculates the whole loss that the un-serviced demands have put to our system. It is computed when the system is at steady state and is defined as:

$$f_{loss} = \sum_i \int \int_{A \in H_i} w_A \, dx \, dy = \sum_i \oint \phi \int_0^{R_i} w_A \, r dr \, d\theta$$

,where: $H_i = \pi R_i^2$ and $R_i = \max_j r_i^j$

4 Forest Fire: A Case Study

In this section, we try to survey the assignment of fireman agents whilst a forest is on fire in some locations. The forest is the location of a military troop and some sections of the forest are more important than the others for us. Consider Figure 1 to be the statement of the beginning of a cycle. The red points show the centers of the fires, and the orange radii around the fire centers show how drastic they are. There is also a gray circle that models how much each fire has caused damage to the military zone since the beginning of the forest fire. The dark gray sections of the forest are very important to the troop. Also there is a dotted sectioning that models the Voronoi diagram of the cycle over the fires. We consider a team of $p = 8$ firemen that can act as we define them their zone of activities. Surely there is not always a need to assign all the firemen, and sometimes we can have less than enough firemen and so, we should decide on which fire to be extinguished depending on the importance of the points near the growing fires. Calling RA-DAWV results in a need of $q = 5$ firemen to the fire centers, according to Figure 2. As they act on the fires, they growth of the fires

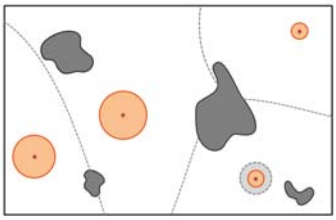


Figure 1: Forest fire, beginning of a cycle

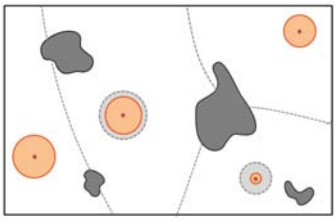


Figure 2: Forest fire, end of the same cycle

become less, but some other fire centers are created by the fire generator engine. So at the end of this cycle, we run the RA-DAWV algorithm once more to come to a newly-made decision. As you can see, the execution of the RA-DAWV algorithm is both easy and helpful. These actions are done repetitively until the last cycle. In the next section we discuss a bit about the RA-DAWV algorithm and its complexity. We also point to some lacuna and weak points of this algorithm that are good to see in the future work of authors.

5 Conclusion and Future Works

In this paper we addressed a certain class of Facility Location problems, hybridized with Voronoi diagrams in which ordinary points of the problem space (and not only service points) have weights of importance. Also sites are considered to be circular and we use AWVD to come to a solution. Our problem is considered in discrete-time and with repetitive cycles, so we believe that we are solving the Dynamic AWVD problem. An example of application is that we have n points of the problem space to go on fire and the fire is expanding in a circular manner. Also the neighbor points of the fire-centers are of different importance to us, so some fires should be controlled and extinguished as soon as possible not to let the more important neighbor points take any damage. If p and c are the number of fireman agents and the number of cycles of a complete run, then drawing an Additively Weighted Voronoi diagram for the set of n circular sites is of $O(n^2 log\ n)$ time complexity. We also feel a need for each agent to be rendered under a Point Location algorithm to recognize its corresponding site. This algorithm will take $O(plog\ n)$ time. Whilst we have at most c cycles, we can claim that the time complexity for a whole parade process of the RA-DAWV algorithm is $O(c(n^2 + p)log\ n)$.
In this paper we have simplified the problem by considering the sites to be circular. Also several parameters could be added to the problem statement to make it more applicable in the real world, like the wind effect on fire. The problem is still open for closed curves. Also it may be possible that the problem could be solved in less time or be proved that our algorithm's time complexity is tight.

References

[1] Sana Moujahed, Olivier Simonin, Abderrafia Koukam, Khaled Ghdira. *A Reactive Agent Based Approach to Facility Location: Application to Transport.* AAMAS'06 May 8-12 2006, Hakodate, Hokkaido, Japan

[2] M. I. Karavelas, M. Yvinec. *Dynamic Additively Weighted Voronoi Diagrams in 2D.* Unit de recherch INRIA Sophia Antipolis, 2004, route des Lucioles, BP93, 06902 Sophia Antipolis Cedex (France).

[3] B. Coppin. *Artificial Intelligence Illuminated, 1st edition.* chapter19, page 54; Jones and Bartlett Publishers, 2004.

[4] Bahram Sadeghi Bigham, Ali Mohades, Lidia M. Ortega. *Dynamic Polar Diagram.* Inf. Process. Lett. 109(2): 142-146 (2008).

[5] Franz Aurenhammer. *Voronoi diagrams-a survey of a fundamental geometric data structure* . ACM Computing Surveys (CSUR), Volume 23 , Issue 3 (September 1991), Pages: 345 - 405.

[6] Daniel P. Huttenlocher, Klara Kedem, Jon M. Kleinberg. *On dynamic Voronoi diagrams and the minimum Hausdorff distance for point sets under Euclidean motion in the plane* . Annual Symposium on Computational Geometr, Proceedings of the eighth annual symposium on Computational geometry, Berlin, Germany, Pages: 110 - 119, 1992.

[7] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness.* W.H Freeman and Co, 1979.

[8] Jean-Daniel Boissonnat, Christophe Delage. *Convex Hull and Voronoi Diagram of Additively Weighted Points.* Lecture Notes in Computer Science, Springer Berlin / Heidelberg, Volume 3669/2005, 367-378.

[9] 10. Li Jin, Donguk Kim, Lisen Mu, Deok-Soo Kim, Shi-Min Hu. *A sweepline algorithm for Euclidean Voronoi diagrams of circles.* Elsevier, Computer-Aided Design 38 (2006) 260-272.

Approximating the Fréchet Distance for Realistic Curves in Near Linear Time

Anne Driemel* Sariel Har-Peled † Carola Wenk‡

Abstract

We introduce a new realistic family of curves, which we call c-packed curves. This family is closed under simplification, a property that makes it especially useful. We can $(1+\varepsilon)$-approximate the Fréchet distance of two polygonal c-packed curves in $\mathbb{R}^d$ in time $O(n/\varepsilon + cn\log n)$. Our algorithm also works in near linear time for low-density curves in the plane.

1 Introduction

The Fréchet distance and its variants have been widely used to compare geometric curves, in applications such as dynamic time-warping, signature verification [8], and others. Unlike the Hausdorff distance, which is solely based on nearest neighbor distances between points on the curves, the Fréchet distance requires continuous and order-preserving assignments of points and hence is better suited for comparing the intrinsic structure of the curves.

It has been an open problem to find a subquadratic algorithm for computing the Fréchet distance for two curves in the plane. In fact, recently Alt [1] conjectured that the decision version of the problem may be 3SUM-hard. The only subquadratic algorithms known are for quite restricted classes of curves such as for closed convex curves and for κ-*bounded* curves [3]. For two polygonal curves of total complexity n in the plane, their Fréchet distance can be computed in $O(n^2\log n)$ time [2]. For closed convex curves the Fréchet distance equals the Hausdorff distance and for κ-bounded curves the Fréchet distance is at most $(1+\kappa)$ times the Hausdorff distance, and hence the $O(n\log n)$ algorithm for the Hausdorff distance (see [1]) can be applied.

Aronov *et al.* [4] provided a near linear time $(1+\varepsilon)$-approximation algorithm for the *discrete* Fréchet distance, which only considers distances between vertices of the curves. Their algorithm works for *backbone curves*, which are required to have (roughly) unit edge length and a minimal distance between any pair of vertices, and which are used to model protein backbones in molecular biology.

The Input Model. Realistic input models, such as *fatness* and *low density*, were introduced for the analysis of problems where the worst case complexity is dominated by degenerate and highly unlikely configurations. We introduce a new class of curves, called c-packed curves, for which we can approximate the Fréchet distance quickly if the constant c is small.

A curve π is ***c*-packed** if the total length of π inside any ball is bounded by c times the radius of the ball. A κ-bounded curve might have infinite length and as such may not be c-packed. But unlike κ-bounded curves, the Fréchet distance between two c-packed curves might be arbitrarily larger than their Hausdorff distance. Indeed, c-packed curves are a considerably more general and a more natural family of curves. For example, a c-packed curve might self cross and revisit the same location several times, and c-packed curves are closed under constant concatenation. Intuitively, c-packed curves behave reasonably in any resolution, unlike backbone curves. See the figure for a few examples of c-packed curves.

The boundary of convex polygons, algebraic curves of bounded maximum degree, the boundary of (α,β)-covered shapes [7], and the boundary of γ-fat shapes [5] are all c-packed. Some fractal curves, like the Koch's snowflake, have infinite length, but a finite diameter. This can also happen to γ-fat curves if one allows unbounded description complexity. However, one can show that (α,β)-covered polygons are c-packed even in this case, see [6].

2 Preliminaries

Notations and Definitions. Let $\pi : [0,1] \to \mathbb{R}^d$ be a curve. In the following, we will identify π with its range $\pi[0,1] \subseteq \mathbb{R}^d$. We use $\|.\|$ to denote both the Euclidean norm of a point as well as the length of a curve. For a polygonal curve π, let $V(\pi)$ denote the set of vertices of π. We denote with $\mathbf{b}(\mathsf{p},r)$ the

*Utrecht University; The Netherlands; anne@cs.uu.nl. This work has been supported by the NWO under RIMGA.

†University of Illinois; http://www.uiuc.edu/~sariel/. This work was supported by a NSF AF award CCF-0915984.

‡University of Texas at San Antonio; carola@cs.utsa.edu. Supported by the NSF CAREER grant CCF-0643597.

ball of radius r centered at p. Let $\binom{\mathsf{P}}{2}$ be the set of all pairwise distances of a point set P. Given a set of numbers $U \subseteq \mathbb{R}$, an ***atomic interval*** of U is a maximal interval that does not contain any point of U in its interior.

Fréchet Distance. A ***reparameterization*** is a one-to-one and continuous function $f : [0,1] \to [0,1]$. It is ***orientation-preserving*** if it maps $f(0) = 0$ and $f(1) = 1$. Given two orientation-preserving reparameterizations f and g for two curves π and σ, respectively, define their ***width*** as $\mathrm{width}_{f,g}(\pi,\sigma) = \max_{s,t\in[0,1]} \|\pi(f(s)) - \sigma(g(t))\|$. The ***Fréchet distance*** between π and σ is defined as

$$\mathsf{d}_{\mathcal{F}}(\pi,\sigma) = \min_{f,g:[0,1]\to[0,1]} \mathrm{width}_{f,g}(\pi,\sigma).$$

Informally, the Fréchet distance can be interpreted as a shortest possible leash one needs to walk a dog, where the dog walks monotonically along π according to f, while the handler walks monotonically along σ according to g. The Fréchet distance complies with the triangle inequality; that is, for any three curves π, σ and τ we have that $\mathsf{d}_{\mathcal{F}}(\pi,\tau) \leq \mathsf{d}_{\mathcal{F}}(\pi,\sigma) + \mathsf{d}_{\mathcal{F}}(\sigma,\tau)$. Let π, σ be curves and $\delta > 0$ a parameter, the ***free space*** of π and σ is defined as

$$\mathcal{D}_{\leq\delta}(\pi,\sigma) = \left\{ (s,t) \in [0,1]^2 \;\middle|\; \|\pi(s) - \sigma(s)\| \leq \delta \right\}.$$

Since we are interested only in polygonal curves, the square $[0,1]^2$ can be broken into a (not necessarily uniform) grid called the ***free space diagram***, where a vertical line corresponds to a vertex of π and a horizontal line to a vertex of σ. Every two segments of π and σ define a ***free space cell*** in this grid. The Fréchet distance between π and σ is at most δ if and only if there is an (x,y)-monotone path in the free space diagram between $(0,0)$ and $(1,1)$ that is fully contained in $\mathcal{D}_{\leq\delta}(\pi,\sigma)$.

3 On c-packed Curves and their Free Space

Let $\pi' = \mathrm{simpl}(\pi,\mu)$ denote a ***μ-simplification*** of π, such that $V(\pi') \subseteq V(\pi)$, and all edges of π' have length at least μ; and let the simplification algorithm preserve a Fréchet distance of at most μ between any edge u of π' and the part that got simplified into u. One can compute such a simplification in time $O(|V(\pi)|)$ using a straight forward greedy algorithm.

Lemma 1 *Let π be a curve in $\mathbb{R}^d$, and let $\pi' = \mathrm{simpl}(\pi,\mu)$ be the simplified curve for a $\mu > 0$. Then $\|\pi \cap \mathbf{b}(\mathsf{p},\mathsf{r}+\mu)\| \geq \|\pi' \cap \mathbf{b}(\mathsf{p},\mathsf{r})\|$ for any ball $\mathbf{b}(\mathsf{p},\mathsf{r})$.*

Definition 2 A curve π in $\mathbb{R}^d$ is ***c-packed*** if for any point p in $\mathbb{R}^d$ and any radius $r > 0$, we have that the total length of π inside the ball $\mathbf{b}(\mathsf{p},r)$ is at most cr.

Lemma 3 *Let π be a c-packed curve in $\mathbb{R}^d$, $\mu > 0$ be a parameter, and let $\pi' = \mathrm{simpl}(\pi,\mu)$ be the simplified curve. Then, π' is a $6c$-packed curve.*

Proof. Assume, for the sake of contradiction, that $\|\pi' \cap \mathbf{b}(\mathsf{p},\mathsf{r})\| > 6c\mathsf{r}$ for some $\mathbf{b}(\mathsf{p},\mathsf{r})$ in $\mathbb{R}^d$.

If $\mathsf{r} \geq \mu$, then set $\mathsf{r}' = 2\mathsf{r}$ and Lemma 1 implies that $\|\pi \cap \mathbf{b}(\mathsf{p},\mathsf{r}')\| \geq \|\pi \cap \mathbf{b}(\mathsf{p},\mathsf{r}+\mu)\| \geq \|\pi' \cap \mathbf{b}(\mathsf{p},\mathsf{r})\| > 6c\mathsf{r} = 3c\mathsf{r}'$, which contradicts the fact that π is c-packed.

If $\mathsf{r} < \mu$, however, then let U denote the segments of π' intersecting $\mathbf{b}(\mathsf{p},\mathsf{r})$ and let $k = |U|$. Observe that $k > 6c\mathsf{r}/2\mathsf{r} = 3c$, as any segment can contribute at most $2\mathsf{r}$ to the length of π' inside $\mathbf{b}(\mathsf{p},\mathsf{r})$. As such, we have that $\|\pi' \cap \mathbf{b}(\mathsf{p},2\mu)\| \geq \|\pi' \cap \mathbf{b}(\mathsf{p},\mathsf{r}+\mu)\| \geq \|U \cap \mathbf{b}(\mathsf{p},\mathsf{r}+\mu)\| \geq k\mu$, since every segment of the simplified curve π' has a minimal length of μ. By Lemma 1, this implies that $\|\pi \cap \mathbf{b}(\mathsf{p},3\mu)\| \geq \|\pi' \cap \mathbf{b}(\mathsf{p},2\mu)\| \geq k\mu > 3c\mu$, which is a contradiction. □

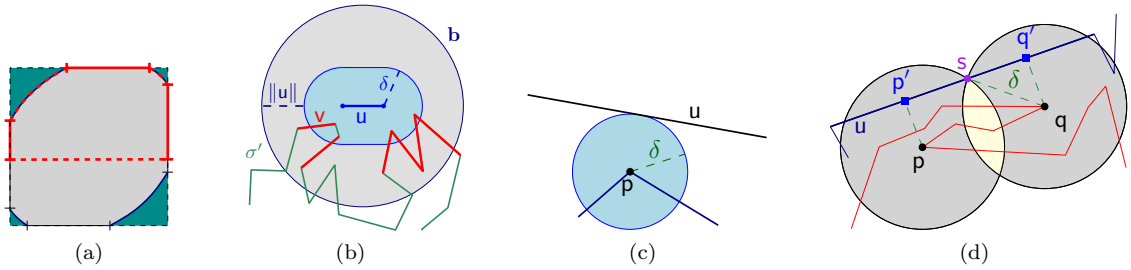
Free Space Computation. For two curves π and σ, their ***relevant free space***, denoted by $\mathcal{R} = \mathcal{R}_{\leq\delta}(\pi,\sigma)$, is the set of all the points of $\mathcal{D}_{\leq\delta}(\pi,\sigma)$ that are reachable from $(0,0)$ by an (x,y)-***monotone*** path. The ***complexity*** of the relevant free space, for distance δ, denoted by $\mathcal{N}_{\leq\delta}(\pi,\sigma)$, is the total number of grid cells with non-empty intersection with $\mathcal{R}$. One can compute this set of cells and extract an existing monotone path in $O(\mathcal{N}_{\leq\delta}(\pi,\sigma))$ time, by performing a BFS of the grid cells that visits only the relevant cells. Note that, to fully describe $\mathcal{R}$, it is sufficient to specify reachability intervals for each relevant cell C, which describe $\mathcal{R} \cap \partial C$, since $\mathcal{R} \cap C$ is the clipping of an affine transformation of a disk to C, see Figure 1a.

Lemma 4 *Let π and σ be two c-packed polygonal curves in $\mathbb{R}^d$ of total complexity n, and let $\mu \leq \delta$ be parameters. Let $\pi' = \mathrm{simpl}(\pi,\mu)$ and $\sigma' = \mathrm{simpl}(\sigma,\mu)$. The complexity of $\mathcal{D}_{\leq\delta}(\pi',\sigma')$ is $O((c + \delta/\mu)n)$.*

Proof. A free space cell of $\mathcal{D}_{\leq\delta}(\pi',\sigma')$ corresponds to two segments $\mathsf{u} \in \pi'$ and $\mathsf{v} \in \sigma'$. The free space in this cell is non-empty if and only if there are two points $\mathsf{p} \in \mathsf{u}$ and $\mathsf{q} \in \mathsf{v}$ such that $\|\mathsf{p} - \mathsf{q}\| \leq \delta$. We charge this pair of points to the shorter of the two segments. Consider a segment $\mathsf{u} \in \pi'$, and consider the ball $\mathbf{b}$ of radius $r = (3/2)\|\mathsf{u}\| + \delta$ centered at the midpoint of u, see Figure 1b. Every segment $\mathsf{v} \in \sigma'$ that gets charged to u, is of length at least $\|\mathsf{u}\|$, and the length of $\mathsf{v} \cap \mathbf{b}$ is at least $\|\mathsf{u}\|$. Since σ' is $6c$-packed, by Lemma 3, we have that the number of such charges is at most

$$c' = \frac{\|\sigma' \cap \mathbf{b}\|}{\|\mathsf{u}\|} \leq \frac{6cr}{\|\mathsf{u}\|} \leq \frac{6c((3/2)\|\mathsf{u}\| + \delta)}{\|\mathsf{u}\|} = 9c + \frac{6\delta}{\mu},$$

since $\|\mathsf{u}\| \geq \mu$. There are at most $c'n$ free space cells that contain a point of $\mathcal{D}_{\leq\delta}$, implying the claim. □



(a) (b) (c) (d)

Figure 1: (a) Example free space cell with example reachability intervals. (b) Illustration of the packing argument for Lemma 4. (c) Vertex-edge event. (d) Monotonicity event and illustration to Lemma 5.

Free Space Events. Consider the structural changes that happen to the free space as one changes δ. We are interested in the ***radii*** (i.e., the value of δ) of these events, since they are candidate values for the Fréchet distance. One can identify three types of events [2]: vertex-edge events, monotonicity events, and the events that two vertices are in distance δ. A ***vertex-edge event*** happens when a vertex is in distance δ of an edge (Figure 1c). The value of a ***monotonicity event*** is the radius δ of the spheres around two vertices, such that an edge passes through the intersection of those spheres, as shown in Figure 1d.

Lemma 5 *Let x be the radius of a monotonicity event involving vertices p, q and a segment u. Then there exists a number y such that $y/2 \leq x \leq 3y$, and y is either in $\mathcal{W} = \binom{V(\pi)\cup V(\sigma)}{2}$ or y is the radius of a vertex-edge event.*

Proof. Let s be the point of intersection of the two spheres of radius x around p and q, which lies on u, see Figure 1d. Let p' (resp. q') be the closest point on u to p (resp. q). Clearly $\|\mathsf{p}' - \mathsf{q}'\| \leq \|\mathsf{p} - \mathsf{q}\|$, and the point s lies on the segment $\mathsf{p}'\mathsf{q}'$. This implies that $x = \|\mathsf{p} - \mathsf{s}\| \leq \|\mathsf{p} - \mathsf{p}'\| + \|\mathsf{p}' - \mathsf{s}\| \leq \|\mathsf{p} - \mathsf{p}'\| + \|\mathsf{p}' - \mathsf{q}'\| \leq \|\mathsf{p} - \mathsf{p}'\| + \|\mathsf{p} - \mathsf{q}\|$, by the triangle inequality. A similar argument implies that $x \geq \|\mathsf{p} - \mathsf{p}'\| - \|\mathsf{p} - \mathsf{q}\|$.

If $\|\mathsf{p} - \mathsf{p}'\| \geq 2\|\mathsf{p} - \mathsf{q}\|$ then the above implies that $x \in [1/2, 3/2]\|\mathsf{p} - \mathsf{p}'\|$, which is the radius of a vertex-edge event. If $\|\mathsf{p} - \mathsf{p}'\| \leq 2\|\mathsf{p} - \mathsf{q}\|$ then $x \leq 3\|\mathsf{p} - \mathsf{q}\|$, and of course $\|\mathsf{p} - \mathsf{q}\| \in \mathcal{W}$. And since the two balls of radius x centered at p and q, respectively, cover the segment pq, we have that $\|\mathsf{p} - \mathsf{q}\|/2 \leq x$, which implies the claim. □

4 The Approximation Algorithm

The Fréchet distance should be sufficiently close to a free space event in one of the "approximate" diagrams, i.e., the free space diagram of the curves simplified with a parameter μ. Thus, we can identify two kinds of events to search over: (i) the event that the simplification of an input curve changes, and (ii) a free space event in the appropriate diagram.

Let $\mathcal{W}$ denote the set of all pairwise distances of $V(\pi) \cup V(\sigma)$. Observe that the simplifications of π (resp. σ) are the same for any two simplification parameters contained in the same atomic interval of $\mathcal{W}$. Thus, we can approximate the simplification events using approximate distance selection, as suggested by Aronov *et al.* [4]. Let **aprxDistances**(P) denote the algorithm that computes a set Z of $O(n)$ numbers, such that for any $y \in \binom{\mathsf{P}}{2}$, there exists an $x \in Z$ that 2-approximates y. Let **searchEvents**$(\pi, \sigma, Z, \varepsilon)$ denote the algorithm that performs a binary search over the values of Z, to compute the atomic interval of Z that contains the Fréchet distance between π and σ. Let **searchInterval**$(\pi, \sigma, [\alpha, \beta], \varepsilon)$ denote the algorithm that does the same on the set Z' that partitions $[\alpha, \beta]$ into $O(\beta/(\varepsilon\alpha))$ subintervals of length $\varepsilon\alpha$. Note that **searchInterval** finds a $(1+\varepsilon)$-approximation of $\mathsf{d}_{\mathcal{F}}(\pi, \sigma)$ if it lies inside $[\alpha, \beta]$.

These two algorithms will need a procedure **decider**(δ, ε) to perform the decisions during the search. Lemma 6 provides a "fuzzy" decider that can be called multiple times with parameter $\varepsilon' < \varepsilon$ to implement the exact decider. To prove the lemma one can choose $\mu = \varepsilon\delta/4$ as the simplification parameter and decide whether the Fréchet distance of the simplified curves is at most $\delta' = \delta + 2\mu$, see [6].

Lemma 6 *Let π and σ be c-packed polygonal curves in $\mathbb{R}^d$ with total complexity n, and let $\varepsilon > 0$ and $\delta > 0$ be parameters. Then, there is an algorithm that, in $O((c + 1/\varepsilon)n)$ time, outputs either of the following:*

(i) If $\mathsf{d}_{\mathcal{F}}(\pi, \sigma) \leq \delta$, then it outputs reparameterizations of width $\leq (1+\varepsilon)\delta$.
(ii) If $\mathsf{d}_{\mathcal{F}}(\pi, \sigma) > (1+\varepsilon)\delta$, then "$\mathsf{d}_{\mathcal{F}}(\pi, \sigma) > \delta$".
(iii) If $\mathsf{d}_{\mathcal{F}}(\pi, \sigma) \in (\delta, (1+\varepsilon)\delta]$, either of the above.

Now, assume we know simplifications τ and η, such that their Fréchet distance represents the desired approximation of $\mathsf{d}_{\mathcal{F}}(\pi, \sigma)$, and that we have narrowed the search range to an interval $[h^-, h^+]$ that does not contain any distance between points in $V(\tau) \cup V(\eta)$. Then, let **aprxFréchetNoSimp** denote the algorithm that computes a $(1+\varepsilon)$-approximation of $\mathsf{d}_{\mathcal{F}}(\tau, \eta)$ in the following fashion. By Lemma 5, any remaining relevant event is 3-approximated by a vertex-edge event with radius $\leq h^+$. We can compute $\mathcal{R}_{\leq h^+}(\tau, \eta)$ and extract these events from it, perform binary search on them, and search the margins

of the acquired interval using a simpler version of **decider** that does not perform simplification. This takes $O((n+N)\log(N/\varepsilon))$ time, for $N = \mathcal{N}_{\leq h^+}(\tau, \eta)$.

The resulting main algorithm is laid out below.

aprxFréchet(π, σ, ε)

(A) $Z \leftarrow$ **aprxDistances**$(V(\pi) \cup V(\sigma))$

(B) $[\alpha, \beta] \leftarrow$ **searchEvents**$(\pi, \sigma, Z, \varepsilon)$

(C) **searchInterval**$(\pi, \sigma, [\alpha, 4\alpha'], \varepsilon)$, for $\alpha' = \frac{30}{\varepsilon}\alpha$

(D) **searchInterval**$(\pi, \sigma, [\beta'/4, \beta], \varepsilon)$, for $\beta' = \beta/3$

(E) Let $\pi' = \text{simpl}(\pi, \mu)$ $\sigma' = \text{simpl}(\sigma, \mu)$, for $\mu = 3\alpha$

(F) $\delta \leftarrow$ **aprxFréchetNoSimp**$(\pi', \sigma', [\alpha', \beta'], \varepsilon/4)$

(G) Return the reparameterizations and the resulting width as the approximation.

Lemma 7 *The algorithm* **aprxFréchet** *computes a* $(1+\varepsilon)$*-approximation to* $\mathsf{d}_{\mathcal{F}}(\pi, \sigma)$.

Proof. If $\mathsf{d}_{\mathcal{F}}(\pi, \sigma) \in [0, 4\alpha'] \cup [\beta'/4, \infty)$ then, by step (B), we find the solution either in step (C) or (D). In particular, this must be the case if $4\alpha' > \beta'/4$.

Otherwise, since $\mu = 3\alpha \leq \beta'/4$, it follows by the triangle inequality that $\mathsf{d}_{\mathcal{F}}(\pi', \sigma') \leq \mathsf{d}_{\mathcal{F}}(\pi', \pi) + \mathsf{d}_{\mathcal{F}}(\pi, \sigma) + \mathsf{d}_{\mathcal{F}}(\sigma, \sigma') \leq 2\mu + \beta'/4 < \beta'$. A similar argument shows $\mathsf{d}_{\mathcal{F}}(\pi', \sigma') > \alpha'$. And since no distance between two vertices is contained in $[\alpha', \beta']$, we can apply **aprxFréchetNoSimp** and retrieve a value $\delta \in [\mathsf{d}_{\mathcal{F}}(\pi', \sigma'), (1+\varepsilon/4)\mathsf{d}_{\mathcal{F}}(\pi', \sigma')]$. By the triangle inequality the returned Fréchet distance is

$$\begin{aligned}\Delta &\leq \mathsf{d}_{\mathcal{F}}(\pi, \pi') + \delta + \mathsf{d}_{\mathcal{F}}(\sigma, \sigma') \\ &\leq \mathsf{d}_{\mathcal{F}}(\pi, \pi') + (1+\varepsilon/4)\mathsf{d}_{\mathcal{F}}(\pi', \sigma') + \mathsf{d}_{\mathcal{F}}(\sigma', \sigma) \\ &\leq 5\mu + (1+\varepsilon/4)\mathsf{d}_{\mathcal{F}}(\pi, \sigma) \leq (1+\varepsilon)\mathsf{d}_{\mathcal{F}}(\pi, \sigma),\end{aligned}$$

since $5\mu = 15\alpha = (\varepsilon/2)\alpha' \leq (\varepsilon/2)\mathsf{d}_{\mathcal{F}}(\pi, \sigma)$. Note that $\Delta \geq \mathsf{d}_{\mathcal{F}}(\pi, \sigma)$ since it is the width of a specific reparameterization between the two curves. □

Theorem 8 *Given two polygonal c-packed curves* π *and* σ *with a total of* n *vertices in* $\mathbb{R}^d$*, and a parameter* $1 > \varepsilon > 0$*, one can* $(1+\varepsilon)$*-approximate the Fréchet distance between* π *and* σ *in* $O((cn/\varepsilon)\log n)$ *time.*

Proof. We only prove the running time, as correctness is provided above. Computing and sorting Z takes $O(n \log n)$ time. Steps (B) and (C) and (D) perform $O(\log n + \log(1/\varepsilon^2)) = O(\log n)$ calls to **decider**, since the two respective intervals have $O(1/\varepsilon^2)$ (resp. $O(1/\varepsilon)$) subintervals. Each call to **decider** takes $O((c+1/\varepsilon)n)$ time by Lemma 4, so overall this takes $O((c/\varepsilon)n \log n)$ time. (We assume that $\varepsilon = \Omega(1/n)$, otherwise we can use the standard approach [2]).

Step (F) takes time $O((n+N)\log(N/\varepsilon))$, with $N = \mathcal{N}_{\leq \beta'}(\pi', \sigma')$. Now, consider $\mu' = \beta'$ and observe that $\text{simpl}(\pi, \mu') = \pi'$ and $\text{simpl}(\sigma, \mu') = \sigma'$, since μ and μ' are in the same atomic interval of the set of simplification events. By Lemma 4, we have that

$$N = \mathcal{N}_{\leq \beta'}(\pi', \sigma') = O((c + \beta'/\mu')n) = O(cn).$$

Thus, step (F) takes $O(cn \log(cn/\varepsilon)) = O(cn \log n)$ time, since $c \leq n$ and $\varepsilon = \Omega(1/n)$. And, clearly, step (E) and (G) also take time in $O(n)$. □

Remark 9 *The running time can be slightly improved to* $O\big(\frac{n}{\varepsilon} + cn \log n\big)$*, see [6].*

Remark 10 *The algorithm also works in near linear time for low-density curves in the plane; and in higher dimensions in subquadratic time, see [6].*

Acknowledgments. The authors thank Mark de Berg and Marc van Kreveld for insightful discussions on the studied problems. This research was initiated during a workshop supported by the Netherlands Organization for Scientific Research (NWO) under BRICKS/FOCUS grant number 642.065.503.

References

[1] H. Alt. The computational geometry of comparing shapes. In *Efficient Algorithms: Essays Dedicated to Kurt Mehlhorn on the Occasion of His 60th Birthday*, pages 235–248. Springer-Verlag, 2009.

[2] H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *Internat. J. Comput. Geom. Appl.*, 5:75–91, 1995.

[3] H. Alt, C. Knauer, and C. Wenk. Comparison of distance measures for planar curves. *Algorithmica*, 38(1):45–58, 2004.

[4] B. Aronov, S. Har-Peled, C. Knauer, Y. Wang, and C. Wenk. Fréchet distance for curves, Revisited. In *Proc. 14th Annu. European Sympos. Algorithms*, pages 52–63, 2006.

[5] M. de Berg. Improved bounds on the union complexity of fat objects. *Discrete Comput. Geom.*, 40(1):127–140, 2008.

[6] A. Driemel, S. Har-Peled, and C. Wenk. Approximating the Fréchet distance for realistic curves in near linear time (full version). `http://cs.uiuc.edu/~sariel/papers/09/frechet/`, 2009.

[7] A. Efrat. The complexity of the union of (α, β)-covered objects. *SICOMP*, 34(4):775–787, 2005.

[8] M. E. Munich and P. Perona. Continuous dynamic time warping for translation-invariant curve alignment with applications to signature verification. In *Proc. 7th Int. Conf. Comp. Vision*, pages 108–115, 1999.

Guarding 1.5D Terrains with Demands

Khaled Elbassioni* Domagoj Matijević† Domagoj Ševerdija‡

Abstract

We consider the 1.5D terrain guarding problem in which every point on the terrain that is to be covered has an integer demand associated with it. The goal is to find a minimum cardinality set of guards such that each point is guarded by a number of guards satisfying its demand. We present an algorithm that yields a 6.7-approximation in the case where the minimum demand $d_{\min} < 5$, and a 3-approximation otherwise. To the best of our knowledge, this is the first constant factor approximation algorithm for this problem.

As in our previous result [6] we use a fractional solution to the linear programming relaxation of the corresponding covering problem to decide, for each point, the amount of demand that has to be satisfied from the left and right sides of the point.

1 Introduction

In the 1.5D terrain guarding problem we are given a polygonal region in the plane determined by an x-monotone polygonal chain, and the objective is to find the minimum number of guards to place on the chain such that every point in the polygonal region is guarded. This kind of guarding problems and its generalizations to 3-dimensions are motivated by optimal placement of antennas for communication networks (see [3, 1] and the references therein for more details).

The problem considered in this paper is generalization of the 1.5D terrain guarding problem. In the *1.5D terrain guarding problem with demands* we are given an x-monotone polygonal chain T in the plane, a set $G \subset T$ of guards and a set $N \subset T$ of points with the associated demand function $d_p : N \to \mathbb{Z}^+$. The goal is to find a minimum cardinality set of guards such that each point $p \in N$ is guarded by at least d_p different guards form this set.

One motivation for studying this version of the problem is that it allows for more robust guarding. Namely, none of the points should stay unguarded even if some of the guards collapse.

*Max-Planck Institute für Informatics, Saarbrücken, Germany, elbassio AT mpi-inf.mpg.de

†Department of Mathematics, J.J. Strossmayer University, Osijek, Croatia, domagoj AT mathos.hr

‡Department of Mathematics, J.J. Strossmayer University, Osijek, Croatia, dseverdi AT mathos.hr

Previous Work Chen et al. [3] claimed that the 1.5D-terrain guarding problem is NP-hard but they did not give a complete proof of the claim (see [5]). They also gave a linear time algorithm for *the left-guarding* problem, that is, the problem of placing the minimum number of guards on the chain such that each point of the chain is guarded from its left. Based on purely geometric arguments, Ben-Moshe et al. [1] gave the first constant-factor approximation algorithm for the 1.5D-terrain guarding problem, though they did not state the value of the approximation factor explicitly (it was claimed to be at least 6 in [8]). Clarkson and Varadarajan [4] gave constant factor approximation algorithms for a more general class of problems using ϵ-nets and showed that their technique can be used to get a constant approximation for the 1.5D-terrain guarding problem. King gave another geometric algorithm[1] with approximation factor 5.

Elbassioni et al. [6] presented a 4-approximation algorithm for the problem. Unlike most of the previous techniques, their method was based on rounding the linear programming relaxation of the corresponding covering problem. Besides the simplicity of the analysis, which mainly relies on decomposing the constraint matrix of the LP into totally balanced matrices, their algorithm generalizes to the weighted and partial versions of the problem.

Most recently, King and Krohn [11] resolved the question about the hardness of the problem and showed that problem is indeed NP-hard. Gibson et al. [7] obtained a PTAS for the standard 1.5D terrain guarding problem using a local search technique. Their analysis relied on a result by Mustafa and Ray [13].

Clarkson et al. [2] considered a number of geometric set covering problems with demands and gave, among other results, an LP-based algorithm that yields an approximation factor of $O(\log z^*)$ for set systems with bounded VC-dimension, where z^* is the value of the optimal solution of the standard covering LP relaxation. By a recent result of King [10], the set system arising in 1.5D-guarding has VC-dimension 4, implying an $O(\log z^*)$-approximation for the 1.5D-guarding problem with demands.

[1] King claimed that the the problem can be approximated within a factor of 4 in [9]; however, his analysis turned out to have an error that increases the approximation factor to 5.

Our Contribution We present an algorithm that yields a 6.7-approximation in the case where the minimum demand $d_{\min} < 5$, and a 3-approximation otherwise. The main idea of our approach is to compute the fractional solution of the corresponding LP and, based on this solution, to decide, for each point, the portion of demands which has to be met from the left side and that which has to be met from the right side. As a result, we end up with left and right guarding problems with demands, such that satisfying the portions of demands on the left and the right, we meet the original demand for each point.

The system matrix for the left and the right problems was shown in [6] to have a special structure; namely, the matrix is *totally balanced*. For that reason, it can be shown that the corresponding left and right guarding problems have integral optimal solutions as described in the book chapter by Kolen and Tamir [12], even in the case with demands. However, in order to get more insight into the problem we provide an alternative proof by constructing a very simple and easy to understand procedure that finds the optimal solution for the left and the right guarding problem. We show how to combine the left and right solutions to arrive at the claimed approximation guarantee.

2 Preliminaries

A terrain T is an x-monotone polygonal chain. Let V denote set of vertices of T, and $|V|$ is complexity of the terrain T. For two points $p, q \in T$ we say that p sees q and denote $p \sim q$ if the line segment connecting p and q does not go strictly below T. We say that p is seen from $S \subset T$ if there exists some $g \in S$ such that $p \sim g$. Let $N \subset T$, $|N| = n$, be some set of points with the demand function $d_p : N \to \mathbb{Z}^+$ defined. Let $G \subset T$, $|G| = m$, be some set of guards.

In the *1.5D terrain guarding problem with demands* the task is to find minimum set of guards $A \subseteq G$ such that every point p in N is guarded by at least d_p guards from A.

We write $p < q$ if point p is on the strict left of q. All approximation algorithms mentioned in the Previous Work part are based on the following *order claim*:

Lemma 1 *Let $a < b < c < d$ be four points on T. If $a \sim c$ and $b \sim d$, then $a \sim d$.*

For any point $p \in N$ we define $S(p)$ to be the set of guards from G that see p, $S_L(p)$ to be the set of guards from G that see point p strictly from the left and $S_R(p)$ the set of guards from G that see p strictly from the right.

3 Terrain guarding with demands

Consider the following integer LP formulation for the problem:

$$\text{minimize} \sum_{g \in G} x_g \qquad \text{(LP1)}$$

subject to

$$\sum_{g \in S(p)} x_g \geq d_p \qquad \forall p \in N \qquad (1)$$

$$x_g \in \{0, 1\} \qquad \forall g \in G$$

Variable x_g indicates whether $g \in G$ is chosen as a guard and constraint (1) demands that every point $p \in N$ is guarded with at least d_p guards from G.

Let x^* denote the optimal solution to the LP relaxation.

Rounding large values. We fix some parameter $\alpha \in (0, 1/2)$ that will be defined later. We let $G_0 = \{g \in G : x^*_g \geq \alpha\}$, which we take into our final solution. Then we get a reduced problem by redefining $d'_p = d_p - |S(p) \cap G_0|$ for all $p \in N$, $N' = \{p \in N : d'_p \geq 1\}$, and $d_{\min} = \min\{d'_p : p \in N'\}$. Define further $G' = G \setminus G_0$, $S'(p) = S(p) \cap G'$ and similarly $S'_L(p)$ and $S'_R(p)$. Let

$$N_L = \big\{p \in N' \mid \sum_{g \in S'_L(p)} x^*_g \geq \frac{1}{2}\big(\sum_{g \in S'(p)} x^*_g - x^*_p\big)\big\}$$

$$N_R = \big\{p \in N' \mid \sum_{g \in S'_R(p)} x^*_g \geq \frac{1}{2}\big(\sum_{g \in S'(p)} x^*_g - x^*_p\big)\big\},$$

where we assume that $x^*_p = 0$ if $p \notin G'$.

For each $p \in N'$, we define

$$\begin{aligned} d_{p,L} &= \lceil \textstyle\sum_{g \in S'_L(p)} x^*_g \rceil, \forall p \in N_L \\ d_{p,L} &= \lfloor \textstyle\sum_{g \in S'_L(p)} x^*_g + x^*_p \rfloor, \forall p \in N_R \\ d_{p,R} &= \lceil \textstyle\sum_{g \in S'_R(p)} x^*_g \rceil, \forall p \in N_R \\ d_{p,R} &= \lfloor \textstyle\sum_{g \in S'_R(p)} x^*_g + x^*_p \rfloor, \forall p \in N_L \end{aligned}$$

as the demand of point p that has to be satisfied from the guards that are to the left and right of p, respectively. Note that every point $p \in N'$ must be either in N_L or N_R.

Consider the LP formulation for the left-guarding problem:

$$\text{minimize} \sum_{g \in G'} x_g \qquad \text{(LP2)}$$

subject to

$$\sum_{g \in S'_L(p)} x_g \geq d_{p,L} \qquad \forall p \in N'$$

$$0 \leq x_g \leq 1 \qquad \forall g \in G'$$

and the corresponding dual:

$$\begin{aligned} &\text{minimize} && \sum_{p\in N'} d_{p,L} y_p - \sum_{g\in G'} z_g && \text{(LP3)}\\ &\text{subject to}\\ &\sum_{p:\ g\in S'_L(p)} y_p - z_g \ \le 1 && \forall g \in G'\\ &\qquad y_p \ge 0 && \forall p \in N'\\ &\qquad z_g \ge 0 && \forall g \in G' \end{aligned}$$

The right-guarding LP can be formulated symmetrically. Let z^*, z^*_L, z^*_R be the optima for the original, left and the right guarding problem, respectively.

The following important claim was shown by Kolen and Tamir [12] from the property that constraint matrix for the left-guarding problem is totally balanced (see Elbassioni et al. [6]). In contrast to their approach, we give a simple procedure that, in $O(nm)$ time, returns an optimal set of guards for the left-guarding problem.

Lemma 2 *Let G_L and G_R be the optimum sets of guards for the left and right guarding problem, respectively. Then $|G_L| = z^*_L$ and $|G_R| = z^*_R$.*

Our combinatorial proof of Lemma 2. For simplicity of notation, we will just assume for this subsection that $G_0 = \emptyset$, and hence $G' = G$. We first give a combinatorial algorithm for the problem of guarding a set of points N from the left.

Algorithm 1 LEFT-GUARDING(T, N, G)

1. $A(p) \leftarrow \emptyset$, $\forall p \in N$
2. **for** $p \in N$ processed from *left to right* **do**
3. **while** the number of guards in $A = \cup_{p\in N} A(p)$ that see p is less than $d_{p,L}$ **do**
4. $A(p) = A(p) \cup \{L(p)\}$
5. **return** A

In the algorithm, with $A(p)$ we denote the set of guards activated to satisfy the demand of the point p, and with $L(p)$ we denote the leftmost guard in the set $S_L(p)\backslash A$.

For the purpose of the analysis, we distribute the dual updates as follows:

Algorithm 2 DUAL-UPDATES$(\{A(p)\}, T)$

1. $y_p = 0$, $\forall p \in N$, $z_g = 0$, $\forall g \in G$
2. $x_g = \begin{cases} 1, & g \in A = \cup_{p\in N} A(p) \\ 0, & \text{otherwise} \end{cases}$
3. **for** $p \in N$ processed from *right to left* **do**
4. **if** $A(p)$ has a non-marked guard **then**
5. mark all guards in $S_L(p)$
6. $y_p = 1$
7. $z_g = \sum_{p|g\in S_L(p)} y_p - 1$, $\forall g \in A$

We assume that all the guards are initially unmarked. We will first argue that the primal and the dual solutions constructed above are both feasible.

Primal feasibility. Follows from line (3) of the *Algorithm 1.*

Dual feasibility. It is enough to show that $\sum_{p|g\in S_L(p)} y_p \ge 1$, $\forall g \in A$, and $\sum_{p|g\in S_L(p)} y_p \le 1$, $\forall g \in G\setminus A$. For the first claim, suppose that $g \in A(p)$ and $y_p = 0$. Then there should exist some $p' > p$ such that p' marked g and hence $y_{p'} = 1$ and $g \in S_L(p')$. For the second claim, consider some guard $\bar{g} \notin A$ and suppose that there are two points $p < p'$ such that $\bar{g} \in S_L(p) \cap S_L(p')$ and $y_p = y_{p'} = 1$. Since $y_p = 1$ there exists a guard $g < \bar{g}$ such that $g \in A(p)$ and g was unmarked when *Algorithm 2* was processing p. By the order claim it follows that g must also see p' and, therefore, cannot be unmarked.

We have found an integer feasible solution of the primal and an integer feasible solution of the dual problem. All that is left to prove that these solutions are optimal is to show that the complementary-slackness conditions hold.

Primal complementary-slackness. We need to show that

$$\begin{aligned} y_p = 1 \ &\Rightarrow \sum_{g\in S_L(p)} x_g = d_{p,L}\\ z_g > 0 \ &\Rightarrow \ x_g = 1. \end{aligned}$$

If $z_g > 0$ then g is in A and hence $x_g = 1$. Suppose that $y_p = 1$. Then we want to claim that $\sum_{g\in S_L(p)} x_g = d_p$. Since $y_p = 1$, there exists a guard $g \in A(p)$ that was unmarked when p was being processed in the *Algorithm 2.* Suppose that $\sum_{g\in S_L(p)} x_g > d_p$. Then there is a $g' > g$ such that $g' \sim p$ and $g' \in A(p')$ for some $p' > p$. Suppose that guard g' is marked. But then the point that marked g' must also mark g by the order claim. On the other hand, if g' is not marked, then the point p' that is to the right of p (and, therefore, processed before point p), would have marked it since $g' \in A(p')$ and is not marked, together with g.

Dual complementary-slackness We need to show that $x_g = 1 \Rightarrow \sum_{p|g\in S_L(p)} y_p - z_g = 1$. This follows from step 7 of *Algorithm 2.* □

We conclude with the final theorem.

Theorem 3 *There is a 6.7-approximation for the 1.5D guarding problem with demands.*

Proof. Note first that $G_0 \cup G_L \cup G_R$ is a feasible solution, since each point p is seen by at least d_p guards from this set. Indeed, if $p \notin N'$ then p is already covered by G_0. If $p \in N'$, then $\sum_{g\in S'_L(p)} x^*_g + \sum_{g\in S'_R(p)} x^*_g + x^*_p \ge d_p - \sum_{g\in S(p)\cap G_0} x^*_g \ge d_p - |S(p)\cap G_0| = d'_p$, from which follows $d_{p,L} + d_{p,R} + |S(p)\cap G_0| \ge$

d_p, implying feasibility for point p, by the feasibility of G_L and G_R for the left and right subproblems, respectively.

Now we bound the approximation ratio. Note first by the definition of G_0 that $|G_0| \leq \frac{1}{\alpha}\sum_{g\in G_0} x^*_g$. We will show next that the restriction of $(1+\beta)x^*$ on G' is feasible for LP2, for some positive constant β. This will imply that $z^*_L \leq (1+\beta)\sum_{g\in G'} x^*_g$. By a similar argument, we can also show that $z^*_R \leq (1+\beta)\sum_{g\in G'} x^*_g$.

Namely, note that $\forall p \in N_L$ it holds that $\sum_{g\in S'_L(p)} x^*_g \geq \frac{1}{2}(d_{\min}-\alpha)$, and thus

$$\begin{aligned}\sum_{g\in S'_L(p)}(1+\beta)x^*_g &= \sum_{g\in S'_L(p)} x^*_g + \beta\cdot\sum_{g\in S'_L(p)} x^*_g \\ &\geq \sum_{g\in S'_L(p)} x^*_g + \beta\cdot\frac{1}{2}(d_{\min}-\alpha) \\ &\geq \sum_{g\in S'_L(p)} x^*_g + 1,\end{aligned}$$

where the last inequality follows for $\beta \geq 2/(d_{\min}-\alpha)$.

Moreover, $\forall p \in N_R$ by the fact that

$$d_{p,L} = \lfloor \sum_{g\in S'_L(p)} x^*_g + x^*_p \rfloor \leq \sum_{g\in S'_L(p)} x^*_g + \alpha,$$

it is enough to show the following

$$\sum_{g\in S'_L(p)}(1+\beta)x^*_g \geq \sum_{g\in S'_L(p)} x^*_g + \alpha \qquad (2)$$

Using $\sum_{g\in S'_L(p)} x^*_g \geq 1-\alpha$ (since otherwise, $d_{p,L}=0$), inequality (2) is satisfied if $\beta\cdot(1-\alpha)\geq\alpha$.

Finally note that for all $g \in G'$ the inequality $(1+\beta)x^*_g \leq 1$ will be satisfied if $\beta \leq \frac{1}{\alpha}-1$.

Hence, the cost of the returned solution is

$$\begin{aligned}|G_0|+|G_L|+|G_R| &= |G_0| + z^*_L + z^*_R \\ &\leq \frac{1}{\alpha}\sum_{g\in G_0} x^*_g + 2\cdot(1+\beta)\sum_{g\in G'} x^*_g \\ &\leq \max\{\frac{1}{\alpha}, 2\cdot(1+\beta)\}z^* \\ &\leq \max\{\frac{1}{\alpha}, 2\cdot(1+\beta)\}OPT\end{aligned}$$

where OPT denotes the optimal integer solution to the original problem.

The above constraints on β imply that the approximation factor is bounded by

$$\gamma = \min_{\alpha\in(0,\alpha')} \max\{\frac{1}{\alpha}, \frac{4}{d_{min}-\alpha}+2, \frac{2\alpha}{1-\alpha}+2\} \qquad (3)$$

where $\alpha' = \min\{\frac{1}{2}, \frac{3+d_{\min}-\sqrt{(3+d_{\min})^2-4d_{\min}}}{2}\}$. Note that for $d_{\min}\geq 3$, $\alpha'=\frac{1}{2}$.

One can easily verify that for $d_{\min}=1$, the maximum value in (3) will be for $\alpha = 0.149$ that balances the terms $\frac{1}{\alpha}$ and $\frac{4}{1-\alpha}+2$. This leads to $\gamma = 6.7$, which concludes the proof of the theorem. □

Remark. With a more careful analysis of (3), one can express the approximation factor in terms of $d_{\min}$, for $d_{\min} < 5$. Moreover, for $d_{\min}\geq 5$ and $\alpha = 1/3$, the approximation factor will reduce to $\gamma = 3$.

References

[1] B. Ben-Moshe, M. J. Katz, and J. S. B. Mitchell. A constant-factor approximation algorithm for optimal 1.5d terrain guarding. volume 36, pages 1631–1647. SICOMP, 2007.

[2] C. Chekuri, K. L. Clarkson, and S. Har-Peled. On the set multi-cover problem in geometric settings. In *SCG*, pages 341–350, 2009.

[3] V. E.-C. Chen, D. and J. Urrutia. Optimal guarding of polygons and monotone chains. In *CCCG*, pages 133–138, 1995.

[4] K. L. Clarkson and K. R. Varadarajan. Improved approximation algorithms for geometric set cover. In *SCG*, pages 135–141, 2005.

[5] E. D. Demaine and J. O'Rourke. Open problems from cccg 2005. In *CCCG*, 2005.

[6] K. M. Elbassioni, E. Krohn, D. Matijevic, J. Mestre, and D. Severdija. Improved approximations for guarding 1.5-dimensional terrains. *Algorithmica*, to appear:x–x, September 2009.

[7] M. Gibson, G. Kanade, E. Krohn, and K. R. Varadarajan. An approximation scheme for terrain guarding. In *APPROX-RANDOM*, pages 140–148. Springer, 2009.

[8] J. King. Approximation algorithms for guarding 1.5-dimensional terrains (master's thesis). *McGill University*, 2005.

[9] J. King. A 4-approximation algorithm for guarding 1.5-dimensional terrains. In *LATIN*, pages 629–640, 2006.

[10] J. King. VC-dimension of visibility on terrains. In *CCCG*, 2008.

[11] J. King and E. Krohn. The complexity of guarding terrains. In *SODA*, pages 1580–1593. SIAM, 2009.

[12] A. Kolen and A. Tamir. *Covering Problems*. P.B. Mirchandani, R.L. Francis, eds., Discrete Location Theory, Wiley, 1990.

[13] N. H. Mustafa and S. Ray. PTAS for geometric hitting set problems via local search. In *SCG*, pages 17–22, New York, NY, USA, 2009. ACM.

Regular triangulations and resultant polytopes

Ioannis Z. Emiris* Vissarion Fisikopoulos*† Christos Konaxis*

Abstract

We describe properties of the Resultant polytope of a given set of polynomial equations towards an output-sensitive algorithm for enumerating its vertices. In principle, one has to consider all regular fine mixed subdivisions of the Minkowski sum of the Newton polytopes of the given equations. By the Cayley trick, this is equivalent to computing all regular triangulations of another point set in higher dimension. However, the number of all regular triangulations is generally much larger than that of the vertices of the Resultant polytope, as illustrated by our experiments [3]. Thus, we study output-sensitive methods by defining classes of subdivisions, called configurations, which yield the same resultant vertex. Moreover, we offer algorithmic versions of certain results by Sturmfels [11], regarding the edges of the Resultant polytope. Lastly, we settle some easy cases, and discuss harder examples.

1 Introduction

We are interested in algorithms that compute the Newton polytope of the Resultant, called Resultant polytope, of a given set of polynomial equations. Resultants are fundamental objects in polynomial equation solving [12], and in implicitizing parametric (hyper)surfaces [2]. In fact, a projection of the resultant polytope yields the Newton polytope of the (unknown) implicit equation, thus reducing implicitization to a problem in linear algebra. One approach is to compute the regular fine mixed subdivisions of the Minkowski sum of the Newton polytopes of the given equations. Another is based on tropical geometry, e.g. [12, ch.9].

These regular fine mixed subdivisions correspond by Cayley trick to the regular triangulations of a point set A. For each point set A, its Secondary polytope's vertices correspond to the regular triangulations of A and there are output-sensitive that enumerate them [6, 10]. However, the number of vertices of a Secondary polytope can be exponential in $|A|$ and there is a many to one correspondence of Secondary vertices to the vertices of the Resultant polytope, illustrated by our experiments [3]. On the other hand, for the Resultant polytope, we only know a weak exponential upper bound on the number of vertices [11, prop.6.1]. The above results force us to focus on output-sensitive algorithms that enumerate classes of subdivisions which yield the same resultant vertex, without enumerating the entire Secondary polytope. We present our work in progress to this end. We offer algorithmic versions of certain results in [11] regarding the edges of the Resultant polytope. Lastly, we settle some easy cases, and discuss harder examples.

*Department of Informatics & Telecommunications, National and Kapodistrian University of Athens, Greece.

†Contact author (vfisikop@di.uoa.gr)

2 Triangulations, Mixed Subdivisions, and Polynomial Systems

Let A be a point set in $\mathbb{R}^d$. A *polyhedral subdivision* of A is a collection of subsets of A, the cells of the subdivision, such that the union of the cells' convex hulls equals the convex hull of A and every pair of convex hulls of cells intersect at a common face. A polyhedral subdivision is *regular* if it can be obtained as the projection of the lower hull of the lifted point set A, for some lifting to $\mathbb{R}^{d+1}$. A *triangulation* T is a polyhedral subdivision of A, whose cells are all simplices. *Circuits* are the minimum affinely dependent subsets of a point set that have exactly two triangulations. A *bistellar flip* transforms one triangulation to another. Let T be a triangulation of A and $Z_+ \subseteq T$ the triangulation of a circuit $Z \subseteq A$. We say T is *supported* on Z if, by changing the current triangulation Z_+ of Z to the other, denoted Z_-, we obtain another triangulation T'. This is a bistellar flip of T supported on Z. If $|A| = n$, its *Secondary polytope* $\Sigma(A)$ has dimension $n-d-1$, its vertices correspond to the regular triangulations of A, and its edges to bistellar flips [5, 8].

Let $A_0, \ldots, A_k$ be point sets in $\mathbb{R}^d$ and $\overline{A} = A_0 + \cdots + A_k$ their Minkowski sum. A subset of $\overline{A}$ is called *Minkowski cell* if it can be written as $F_0 + \cdots + F_k$ for $F_i \subseteq A_i$. A Minkowski cell is *fine* if all F_i are affinely independent and $\sum_{i=1}^{k} dim(CH(F_i)) = d$. When $k = d$, a Minkowski cell is *i-mixed* if it is a Minkowski sum of k edges and a vertex, i.e., $|F_j| = 2$ for $j \neq i$, $|F_i| = 1$. When $k = d-1$, a Minkowski cell is *mixed* if it is a Minkowski sum of edges. A regular polyhedral subdivision of $\overline{A}$ is a *regular fine mixed subdivision* if all its cells are Minkowski and fine. From now on we consider all mixed subdivisions to be regular and fine, and focus on $k = d$, unless otherwise noted. This is the most important case because it covers system

solving and implicitization; implicitization of surfaces in $\mathbb{R}^3$ corresponds to $k = d = 2$.

Let $f = f_0, \ldots, f_k$ be a polynomial system on k variables. The support $A_i \in \mathbb{N}^k$ of f_i is the set of its exponent vectors corresponding to nonzero coefficients. For any subset $J \subset \{0, \ldots, k\}$, let $r(J)$ denote the rank of the affine lattice generated by $\sum_{j\in J} A_j$. We assume that, for $I = \{0, \ldots, k\}$, $r(I) = |I| - 1$, and $r(J) \geq |J|$ for any proper subset $J \subset I$. The *Newton polytope* $N(f_i)$ of a polynomial f_i is the convex hull of its support. The (sparse) Resultant R of f is a polynomial on the coefficients of f such that $R = 0$ iff f has a solution in $(\mathbb{C}^*)^k$. It generalizes the determinant of an overconstrained linear system and the Sylvester resultant of two univariate polynomials. We call $N(R)$ the *Resultant polytope* and *extreme term* of R a monomial which corresponds to a vertex of $N(R)$.

Proposition 1 [11, thm.2.1] *Following the above notation and assumptions, given a system f and a mixed subdivision of the Minkowski sum of its supports, we get an extreme term of the resultant R equal to*

$$\pm \prod_{i=0}^{k} \prod_{\sigma} c_{iF_i}^{vol(\sigma)},$$

where $\sigma = F_0 + \cdots + F_k$ is an i-mixed cell and $vol(\cdot)$ denotes Euclidean volume.

By the Cayley trick [5], there is a point set $C(A_0, \ldots, A_k) \subset \mathbb{R}^{d+k}$ s.t. all mixed subdivisions of $\overline{A} = A_0 + A_1 + \cdots + A_k$ are in 1-1 correspondence with the regular triangulations of $C(A_0, A_1 \ldots, A_k)$. Hence, one can obtain the $N(R)$ vertices by enumerating all vertices of the corresponding Secondary polytope $\Sigma(A_0, A_1 \ldots, A_k)$. Moreover, $N(R)$ is a Minkowski summand of $\Sigma(A_0, A_1 \ldots, A_k)$ [11]. Methods to enumerate regular triangulations have been proposed in [6, 10], and are experimented with in [3]. But we can do better.

When $k = d - 1$, mixed cell *configurations* are the equivalence classes of mixed subdivisions with the same mixed cells. These are defined, along with a definition of flips between these classes, in [9].

When $k = d$, we focus on the i-mixed cells in order to compute the vertices of $N(R)$. In [7], there is an extension of mixed cells configurations to classes containing the same i-mixed cells for all $i \in \{0, \ldots, k\}$, called i-mixed cell configurations. It turns out that this notion is similar to the I-mixed cell configurations of [1]. We now characterize the flips between i-mixed cell configurations, and generalize the flip defined in [9] between mixed cell configurations.

We shall say that a circuit Z of a triangulation T supported on Z, *involves* an i-mixed cell $F_0 + \cdots + F_k$, if the cell $C(F_0, \ldots, F_k)$ of T does not belong to the triangulation obtained by flipping on Z.

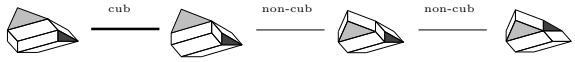


Fig. 1: An example of a cubical and two non cubical flips.

Theorem 2 *([7]) Let $Z = (Z_0, \ldots, Z_k)$ be a circuit and T a triangulation supported on Z. Suppose that Z involves an i-mixed cell $F_0 + \cdots + F_k$. Then, there exists $r \in \{0, \ldots, k\}$, and $c \in A_r$ s.t. for all $i \neq r$, $Z_i = F_i$ or $Z_i = \emptyset$, and $Z_r = F_r \cup \{c\}$ or $Z_r = \{v_r, c\}$, where v_r is a vertex of edge F_r.*

A flip on a circuit as described in this theorem destroys at least one i-mixed cell leading to a new i-mixed cell configuration. Moreover, we can check efficiently if a circuit satisfies the conditions of th. 2 by examining only the cardinalities of the sets Z_i. An algorithm using these flips enumerates only the i-mixed cell configurations, without enumerating all mixed subdivisions, which are more numerous.

The Ξ *polytope* is defined in [1] for $k \leq d-1$. In particular, when $k = d - 1$, Ξ has vertices corresponding to mixed cell configurations, and edges corresponding to flips between them. Based on this, we define Ξ in the case $k = d$ to have vertices corresponding to i-mixed cell configurations. Clearly, Ξ lies, in terms of number of vertices, between the Secondary polytope and $N(R)$. In the sequel, Ξ or $\Xi(A_0, A_1, \ldots, A_k)$ refers to this polytope.

3 R-equivalent Classes

By prop. 1, several mixed subdivisions may produce the same extreme term of the Resultant. We call these subdivisions *R-equivalent*. Similarly, two subdivisions may lead to the same extreme term, even if they belong to the same i-mixed cell configuration. These R-equivalent classes correspond to the vertices of $N(R)$. There are some flips that connect two subdivisions in different R-equivalent classes, hence they correspond to the edges of $N(R)$.

Sturmfels [11, thm.5.2] calls these flips *cubical*. Consider the union of cells affected by one such flip. If the union, lifted generically to $\mathbb{R}^{d+1}$, forms an affine cube, i.e. equals the Minkowski sum of $k + 1$ edges, then the flip is cubical and consists in replacing the "bottom" subdivision by the "top" subdivision, or vice versa (fig. 1, fig. 2). However, this definition of cubical flips is not algorithmically efficient, so we provide a more algorithmic characterization.

Let us start with the generic case, where every two faces of the same dimension in two different $CH(A_i)$ are not parallel.

Lemma 3 *Let S be a mixed subdivision of $A_0 + \cdots + A_k$. Then S has a cubical flip iff there exists a set $\{C_0, \ldots, C_k\}$ of i-mixed cells $C_i = F_0 + \ldots + a_i + \cdots +$*

Fig. 2: A degenerate cubical flip; two (horizontal bold) edges from different A_i's are parallel.

F_k, for $i = \{0, 1, \ldots, k\}$, where $a_i \in F_i \subseteq Ai, |F_i| = 2$, such that, if $C = \bigcup_{i=0}^{k} C_i$, then $C = F_0 + \cdots + F_k$. We say S is supported on C. The cubical flip on S consists of substituting, in every C_i, point a_i with $F_i - \{a_i\}$.

If the generic position assumption does not hold, lem. 3 does not hold, so we generalize this characterization using triangulations. Recall that the set C of i-mixed cells corresponds by Cayley trick to a set Z of simplices and a flip between two mixed subdivisions is a flip between the two corresponding triangulations. Genericallly, C has $k+1$ cells and Z has $k+1$ simplices. The union of these simplices contains $2k+2$ points in a space of dimension $k+d$. If $d = k$, this union of simplices is a circuit. In degenerate cases, there may exist lower dimensional circuits and C may have $< k+1$ cells. As an illustration compare the generic example (fig. 1), where C has 3 cells, with the degenerate example (fig. 2), where C has only 2 cells.

Theorem 4 *Let S be a mixed subdivision of $A_0 + \cdots + A_k$, and T the corresponding triangulation w.r.t. Cayley's trick. Then S has a cubical flip if there exists a set $C = \bigcup_{i=0}^{k} C_i \subseteq S$ of i-mixed cells, as in lem. 3 and, additionally, the corresponding set Z of simplices in T supports a bistellar flip. The cubical flip on S is the bistellar flip of T supported on Z.*

The mapping of cubical flips edges of $N(R)$ is many to one. When a cubical flip is supported on set C, we say that the edge is of type C. Many cubical flips may be supported on the same set C. The types of all Resultant edges can be easily enumerated: they are all possible resultant polytopes of subsets of A_i's with cardinality two. This enumeration also yields the corresponding edge direction, i.e. the difference vector between the two endpoints of $N(R)$. More generally, all faces of $N(R)$ are Minkowski sums of Resultant polytopes corresponding to subsystems of $A_0, \ldots, A_k$. Conversely, every resultant polytope defined on subsets of the A_i's appears as Minkowski summand on some face of $N(R)$ [11].

Example 5 Let $A_0 = \{(0,0), (1,2), (4,1)\}$, $A_1 = \{(0,1), (1,0)\}$, $A_2 = \{(0,0), (0,1), (2,0)\}$, which satisfy the general position assumption. The Secondary polytope of $C(A_1, A_2, A_3)$ is depicted in fig. 3 (left). One can see the R-equivalent classes (dotted) as well as the cubical flips (bold) which connect these classes. All the other flips (non bold) are non-cubical flips. The Resultant polytope can be seen as the polytope with R-equivalent classes as vertices and cubical flips as edges. To each Resultant vertex corresponds one or more mixed subdivisions, and to each edge one or more cubical flips. Here, $\Sigma(C(A_0, A_1, A_2) = \Xi(A_0, A_1, A_2)$ with 36 vertices; $N(R)$ has 6 vertices, and 11 edges corresponding to 9 different cubical flips (fig. 3 right) which are all generic.

4 Secondary, Ξ, and Resultant Polytopes

Let Σ be the Secondary polytope of $C(A_0, A_1, \ldots, A_k)$, Ξ the polytope $\Xi(A_0, A_1, \ldots, A_k)$, and $N(R)$ the Resultant polytope. We offer a case study on these polytopes, and focus on $d = k$.

When $d = k = 1$, every Minkowski cell is an edge, i.e., a sum of an edge and a vertex, thus an i-mixed cell. Then $\Sigma = \Xi$, and they are generally larger than $N(R)$. The number of vertices of $N(R)$ is $\binom{m_0+m_1-2}{m_0-1}$ [4].

For arbitrary d, k, if all $|A_i| \leq 3$, then $\Sigma = \Xi$. To see this, recall that for any fine Minkowski cell $F = \sum_{i=1}^{k} F_i$, it holds that $\sum_{i=1}^{k} dim(F_i) = d$. So, F is not i-mixed iff for some F_i, we have $dim(F_i) > 1$. Since $|A_i| \leq 3$, by the pigeonhole principle, every non i-mixed cell is a sum of $k-2$ edges, two vertices and a triangle. So the union of every pair of non i-mixed cells can be written uniquely.

The smallest case that this does not hold is when there exists i s.t. $|A_i| = 4$, and $|A_j| \leq 3$, $\forall j \neq i$.

Example 6 An instance of the smallest case, for $d = k = 2$, is $A_0 = \{(0,0), (0,1), (2,0), (2,1)\}$, $A_1 = \{(0,0), (1,1), (2,0)\}$, $A_2 = \{(0,0), (0,1)\}$, where Σ, Ξ, and $N(R)$ have, resp., 122, 98, and 8 vertices.

For arbitrary d, k, if for all i, $|A_i| = 2$, then $\Sigma = \Xi = N(R)$. This is the case where all flips are cubical, every Minkowski cell is a sum of edges, called zonotope, and the mixed subdivisions are zonotopal tilings. The above discussion proves the following.

Lemma 7 *If $d = k = 1$, or for all i, $|A_i| \leq 3$, then $\Sigma = \Xi$ and they are at least as large as $N(R)$. If for all i, $|A_i| = 2$, then $\Sigma = \Xi = N(R)$.*

In addition to the case analysis above, we offer relevant experimental results in [3]. In particular, we consider some (highly) nontrivial examples corresponding to the implicitization of a parametric sphere [2].

Example 8 The A_i's are $\{(0,0), (0,2), (2,0), (2,2)\}$, $\{(0,0), (1,0), (0,2), (2,0), (1,2), (2,2)\}$, $\{(0,0), (0,1), (0,2)\}$, and Σ, Ξ and $N(R)$, resp. have 76280, 32076 and 95 vertices.

The A_i's are $\{(0,0), (1,0), (0,2), (2,0), (1,2), (1,2)\}$, $\{(0,0), (0,2), (1,1), (2,0), (2,2)\}$, $\{(0,0),$

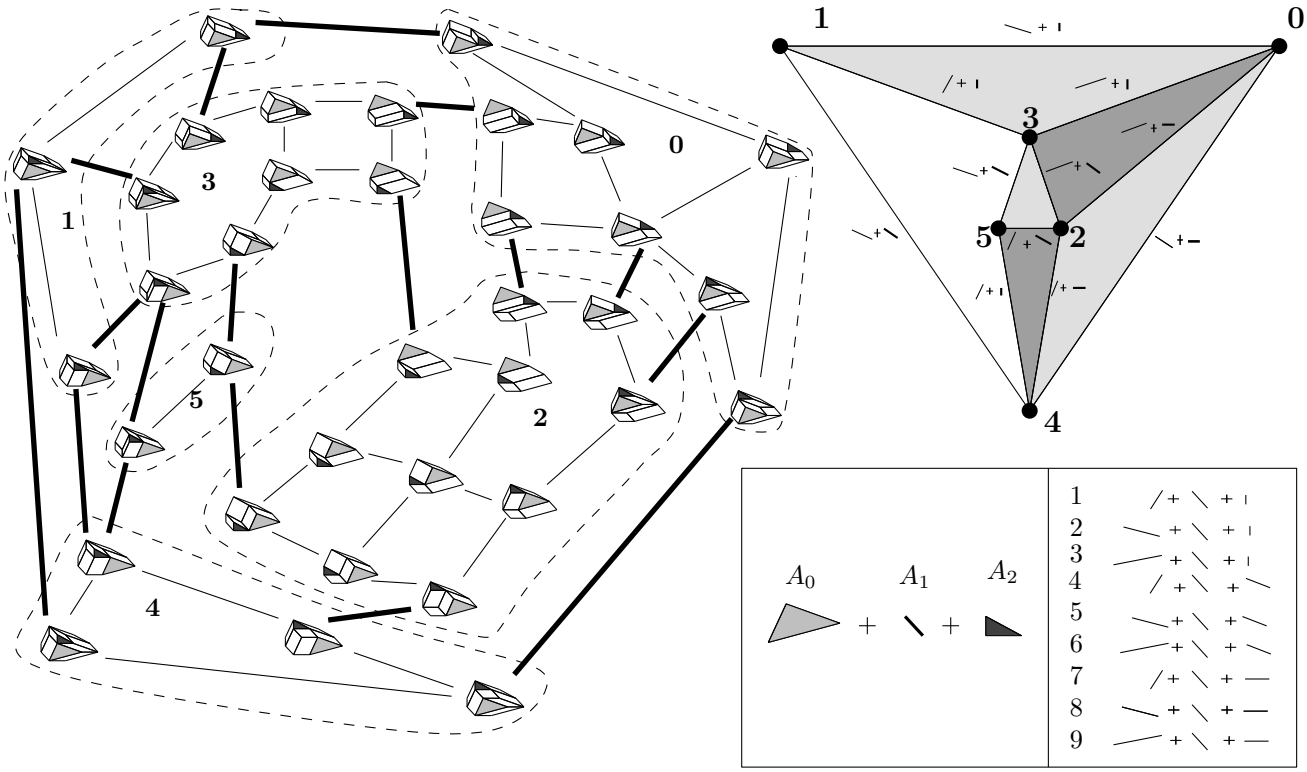


Fig. 3: [left] The Secondary polytope of example 5 with the R-equivalent classes (dotted) and the cubical flips (bold). [right-up] The Resultant polytope of example 5. Each edge is a cubical flip; note that the same cubical flip may appear in two different edges (9 flips in 11 edges). [right-botom] The input point sets and the cubical flips.

$(2,0)\}$ and Σ, Ξ and $N(R)$ have resp. 104148, 43018 and 21 vertices.

Our ultimate goal is an algorithm to enumerate all vertices of $N(R)$ without enumerating the entire Σ or Ξ. To this end we need a unique representation of the resultant vertices and some kind of flip based on the cubical flip. At present, cubical flips do not suffice because there are cases where it is not clear how to obtain one vertex from another just with cubical flips (see fig.3).

Acknowledgments. I.E. and C.K. are supported by the General Secretariat of Research & Technology, Greece, through PENED'03 program, contract 70/03/8473, co-funded by the European social fund (75%) and national resources (25%). V.F. is partially supported by Marie-Curie Network SAGA, contract PITN-GA-2008-214584.

References

[1] R. Cools and T. Michiels. Decomposing the secondary Cayley polytope. *Discrete & Comp. Geometry*, 23(3):367–380, 2000.

[2] I.Z. Emiris and I.S. Kotsireas. Implicitization exploiting sparseness. In R. Janardan, M. Smid, and D. Dutta, editors, *Geometric and Algorithmic Aspects of Computer-Aided Design and Manufacturing*, volume 67 of *DIMACS*, pages 281–298. AMS, 2005.

[3] V. Fisikopoulos. Implicitization experiments on curves and surfaces webpage. http://www.di.uoa.gr/~vfisikop/msc_thesis/exper_implicit.html.

[4] I.M. Gelfand, M.M. Kapranov, and A.V. Zelevinsky. Newton polytopes of the classical resultant and discriminant. *Adv. Math.*, 84(2):237–254, 1990.

[5] I.M. Gelfand, M.M. Kapranov, and A.V. Zelevinsky. *Discriminants, resultants, and multidimensional determinants*. Birkhäuser, Boston, 1994.

[6] H. Imai, K. Imai, and T. Masada. Enumeration of regular triangulations. In *Proc. Symp. Comp. geometry*, pages 224–233. ACM, 1996.

[7] C. Konaxis. Triangulations and resultants. Univ. of Athens, Master's Thesis (in Greek), 2006.

[8] J.A. De Loera, J. Rambau, and F. Santos. *Triangulations: Structures and algorithms*. Manuscript, 2009.

[9] T. Michiels and J. Verschelde. Enumerating regular mixed-cell configurations. *Discrete & Comp. Geometry*, 21(4):569–579, 1999.

[10] J. Pfeifle and J. Rambau. Computing triangulations using oriented matroids. In M. Joswig and N. Takayama, editors, *Algebra, Geometry & Software Systems*, pages 49–75, Berlin, 2003. Springer.

[11] B. Sturmfels. On the Newton polytope of the resultant. *J. Algebraic Comb.*, 3(2):207–236, 1994.

[12] B. Sturmfels. *Solving Systems of Polynomial Equations*. CBMS Regional Conferences Series. AMS, Rhode Island, 2002.

Approximate Nearest Neighbor Queries among Parallel Segments

Ioannis Z. Emiris* Theocharis Malamatos† Elias Tsigaridas‡

Abstract

We develop a data structure for answering efficiently approximate nearest neighbor queries over a set of parallel segments in three dimensions. We connect this problem to approximate nearest neighbor searching under weight constraints and approximate nearest neighbor searching on historical data in any dimension and we give efficient solutions for these as well.

1 Introduction

Nearest neighbor searching is a fundamental geometric problem with applications in many areas. For high dimensions there are no known efficient exact solutions and thus approximate solutions to the problem have been studied. Let $d(p, q)$ denote the euclidean distance between points p, q. Given a set P of points in $\mathbb{R}^d$ and a parameter $\varepsilon > 0$, we say that a point p of P is an *ε-approximate nearest neighbor* (ε-NN) to a point q if $d(p, q) \leq (1+\varepsilon)d(q', q)$ where q' is a nearest point to q in P. Arya et al. [3] have shown how to find efficiently an ε-NN to any given query point in constant dimensions and Indyk and Motwani [6] presented efficient methods for high dimensions. See [5] for more references.

An interesting generalization of the problem arises if we replace the point set P with a set of objects O. For this version there are only few results known. When O is a set of disjoint polyhedra in three dimensions Koltun and Sharir [7] presented a data structure of near quadratic size that can answer an ε-NN query in $O(\log(n/\varepsilon))$ time. In three dimensions again, Wang [9] showed how to answer ε-NN queries when O is a set of triangles, segments, and points in a convex position in $O(\log^2 n/\varepsilon^2)$ query time and using $O(n/\varepsilon^2)$ space. In high dimensions, Magen [8] provided an algorithm for a set of k-flats with query time polynomial in d, $\log n$ and $1/\varepsilon$ but non-polynomial space.

In this paper we consider the case where O is a set of parallel segments. We give two solutions for the problem. The second solution enhances the first using results in Sec. 3. That section is motivated by the *cheap gas station* problem. Given n gas stations (sites) and a car at a position q (query point) we want to find a gas station that is closest or approximately closest to q (since exact distance is not so important) which sells gas for at most w euros. In Sec. 4 we combine the results of the previous sections and present a data structure for answering time-dependent ε-NN queries in any fixed dimension.

2 Methods for parallel segments

Let S be a set of n disjoint parallel segments in $\mathbb{R}^3$. We assume w.l.o.g. that all segments in S are parallel to the x-axis. Let P be the set of the $2n$ endpoints of the segments in S. Let q be a query point in $\mathbb{R}^3$, s be a segment of S nearest to q, and p be the point of s nearest to q. Observe that p is either one of the endpoints of s or that the segment qp is perpendicular to s. Let H_q be the plane passing through q that is parallel to the yz plane. Note that if p is interior to s then p is one of the points in $H_q \cap S$. (See Fig. 1.)

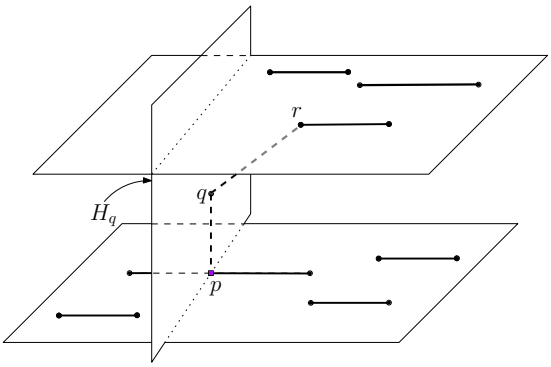


Figure 1: Parallel segments lying in two parallel planes in $\mathbb{R}^3$ and a query point q. The closest endpoint to q is r but the closest point to q is p.

It follows that to find an ε-NN to q in S it suffices to (a) find an ε-NN to q in P, (b) find an ε-NN to q in the set $H_q \cap S$ and then report whichever of the two is nearest to q. For solving (a) we use an (t, ε)-approximate Voronoi diagram (AVD) on set P with $t = O(1/\varepsilon)$ together with the associated data structure [2]. This structure has $O(n)$ space and it returns an ε-NN to any q in $O(\log n + 1/\varepsilon)$ time. For solving (b) we present two methods which are both based on

*Dept. of Informatics and Telecoms, National and Kapodistrian University of Athens, Greece.

†Dept. of Computer Science and Technology, University of Peloponnese, Greece.

‡Dept. of Computer Science, Aarhus University, Denmark and Dept. of Computer Science and Technology, University of Peloponnese, Greece. Partially supported by a research grant from the Danish Council of Independent Research.

a variation of the well-known interval tree [4]. The second method is more complicated but leads to a significant improvement over the first.

2.1 First method

The construction of our interval tree T on S proceeds as follows. Let x_m be the median among all x-coordinates of P. Let H_m be the plane passing through point $(x_m, 0, 0)$ and which is parallel to the yz plane. We store x_m at the root of T. We partition the set of segments S into three sets S_ℓ, S_m, and S_r where each set consists of the segments that lie on the left of H_m, that intersect H_m, and that lie on the right of H_m, respectively. We continue the construction of the tree T recursively on S_ℓ and S_r. (If S_ℓ or S_r is the empty set clearly we get a leaf.) The two roots of the trees built on S_ℓ and on S_r become the left and right child of the root of T, respectively. See Fig. 2 for an example. For the set S_m we build an

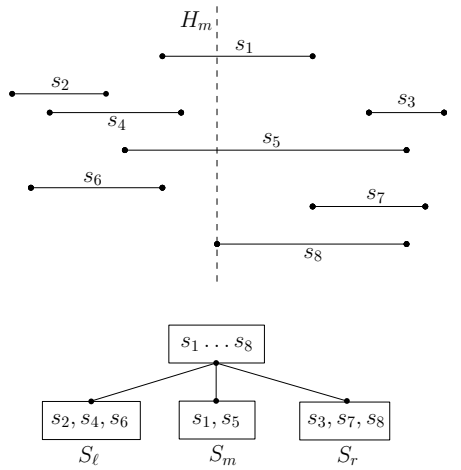


Figure 2: A projection of a set of segments in xz plane. The left endpoint of s_8 is the median x_m. H_m divides the segments to three sets S_ℓ, S_m and S_r. Below the tree corresponding to this partition of segments.

auxiliary data structure T_m which we associate with the root of T. (Structures similar to T_m are built for all internal nodes of T.) T_m is built as follows. H_m cuts naturally each segment in S_m into two pieces. Let C_ℓ be the left pieces of the segments and C_r the right pieces. We build one tree on C_ℓ and one on C_r. We describe the construction of the tree only for C_r since it is symmetric for C_ℓ.

Let x'_m be the median among all x-coordinates of the right endpoints of the segments in C_r. (Note that the x-coordinates of the left endpoints are all equal.) Let H'_m be the plane passing through the point $(x'_m, 0, 0)$ and which is parallel to the yz plane. We store x'_m at the root. The segments of C_r that were not cut by H'_m form the set S'_ℓ. The right pieces of the segments cut by H'_m form the S'_r. The left pieces (which span between planes H_m and H'_m) form the set S'_m. For S'_ℓ, S'_r we continue the construction recursively (unless empty), much like we built our interval tree T above. See Fig. 3 for an example of such a construction. We use the set of segments S'_m to construct a 2D Voronoi diagram for the point set $H'_m \cap S'_m$. Then this is combined with a standard point location algorithm [4] to gives us a data structure T_2 for answering optimally 2D nearest neighbor queries over $H'_m \cap S'_m$. T_2 is associated with the root of the tree for C_r. Structures similar to T_2 are also built for all internal nodes of T_m.

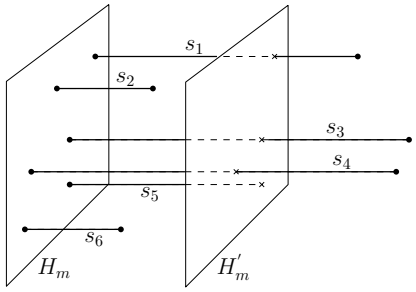


Figure 3: The left endpoint of s_5 is the median x'_m. The parts of segments s_1, s_2 and s_3 that are on the left of H'_m form the set S'_r. The parts between H_m and H'_m, together with s_5 form S'_m. S'_ℓ is formed by s_2 and s_6.

We compute a bound on the size of T_m, that is, for the augmented trees on C_ℓ and on C_r. Let $n' = |S_m|$. Because we split always at the median, the height of T_m is $O(\log n')$. This implies that one segment of S_m may be cut at most $O(\log n')$ times and thus the total size of T_m is $O(n' \log n')$. Using standard results, it is easy to see that the total size of all structures T_2 associated with the nodes of T_m is also $O(n' \log n')$.

We compute a bound on the size of the main data structure T. Since the set of segments S_m used at each node of T for the auxiliary data structure T_m are disjoint and using the space bound on T_m it follows easily that the total size of T is $O(n \log n)$. Due to the balanced splits T has height $O(\log n)$.

We describe now how to solve (b) that is, given a query $q = (q_x, q_y, q_z)$, how to find an ε-NN to q in the set $H_q \cap S$. (In fact this first method finds an exact nearest neighbor to q). We start at the root of T and at each node v we follow the child according to the result of the comparison between q_x and x_m, the value stored at v. At each node v we also visit the auxiliary data structure T_m. We similarly follow the path from the root of T_m to the leaf containing q and at each node we use T_2 to find the nearest neighbor to (q_y, q_z) among $H'_m \cap S'_m$. We report as an answer the nearest point to q over all the points returned from all queries to T_2.

Correctness follows from the fact that we only ex-

clude from consideration segments or fragments of segments that do not intersect plane H_q and thus are not needed for (b).

Since the depth of tree T is $O(\log n)$ and for each node of T we visit an auxiliary data structure T_m with depth also at most $O(\log n)$ and at each node of T_m the data structure T_2 may have been built for at most n sites, it follows that query time is $O(\log^3 n)$. The total query time is the sum of the times used to solve (a) and (b) and thus we get the following result:

Theorem 1 *Given n parallel segments in 3D we can construct a data structure of $O(n \log n)$ space for finding an ε-NN to any given query point q in $O(\log^3 n + 1/\varepsilon)$ time.*

We will discuss the construction times of the data structures in the full version, however they are all in $O(n \operatorname{poly}(\log n, \frac{1}{\varepsilon}))$.

2.2 Improving space and query time

We present next a second method that reduces both the space and query time by a $\log n$ factor. The part of the first method that we change is the auxiliary data structure T_m for the sets C_ℓ and C_r. We again describe just the data structure T_r for segments in C_r and an analogous data structure can be built for C_ℓ.

According to (b), our goal for C_r is to find an ε-NN to q in $H_q \cap C_r$. We solve this by reducing our problem to a weight-constrained approximate nearest neighbor searching problem in two dimensions. Specifically each right endpoint (p_x, p_y, p_z) of a segment in C_r is mapped to the point (p_y, p_z) with weight p_x. We denote with P' the 2D weighted point set obtained (see Fig. 4). Let P'_w be the subset of P' containing only the points with weight at least w. Given a query $q = (q_x, q_y, q_z)$, our goal is to find an ε-NN to point $q_2 = (q_y, q_z)$ in P'_{q_x}. Note that this suffices to achieve our first goal.

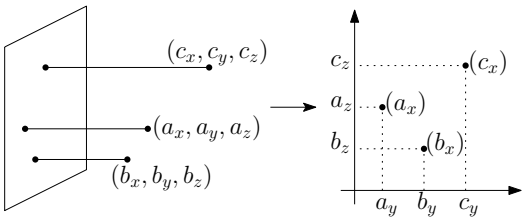


Figure 4: Projection to 2D. The x-coordinate of each point is used as a weight (enclosed in parentheses).

We apply Theorem 4 of the next section (weight-constrained ε-NN queries) on point set P' for $d = 2$, $\gamma = 2$ and $q = q_2$ and easily get this lemma:

Lemma 2 *Given q and C_r we can build a data structure T_r of $O(|C_r| \log(1/\varepsilon))$ size to find an ε-NN to q in $H_q \cap C_r$ in $O(\log |C_r| + 1/\varepsilon^2)$ time.*

Using the above lemma we obtain the following result: (We omit the analysis which is similar to that of the first method.)

Theorem 3 *Given n parallel segments in 3D we can construct a data structure of $O(n \log(1/\varepsilon))$ space for finding an ε-NN to any given query point q in $O(\log^2 n + \log n/\varepsilon^2)$ time.*

3 Weight-constrained ε-NN queries

Given a set of weighted d-dimensional points, we define the *weight-constrained ε-approximate nearest neighbor* problem: given a query q and a weight w, find an ε-NN to q among the points in P that have weight at least w. Here w is a number specified at query time. Note that we allow an approximation error in one parameter (distance) but we require exactness on another (weight). We can also define the symmetric problem where we search for an ε-NN among the points of P with weight at most w.

For the rest of this section we consider only the max version of the problem. We state below our result. This result also provides a space-time tradeoff which is controlled by the parameter γ.

Theorem 4 *Let P be a set of n weighted points in $\mathbb{R}^d$, and let $0 < \varepsilon < 1/2$ and $2 \le \gamma \le 1/\varepsilon$ be two real parameters. We can construct a data structure of $O(n\gamma^d \log(1/\varepsilon))$ space that allows us to answer a weight-constrained ε-NN query in time $O(\log(\gamma n) + 1/(\varepsilon\gamma)^d)$.*

For reasons of space we give here only some of the basic ideas and methods that we have used to prove the theorem. We start with some definitions. Let $b(q, r)$ be a ball of radius r centered at point q. Let $b^+(q, r)$ be a ball of radius $(1+\varepsilon)r$ centered at q. Let $\overline{b}(q, r)$ be the set of points not contained in $b(q, r)$.

Given P, q and r, an *ε-approximate spherical range maximum query* or simply *ε-range maximum query* returns a point in P along with its weight that lies in $b^+(q, r)$ and has weight at least as large as the maximum weight among all points in $b(q, r)$. There are a number of data structures for answering efficiently ε-range maximum queries. Here we will use the data structure in [1]. For $2 \le \gamma \le 1/\varepsilon$ it uses $O(n\gamma^d \log(1/\varepsilon))$ and has query time $O(\log(\gamma n) + 1/(\varepsilon\gamma)^{d-1})$. The dependence on ε in query time can be further improved by using known results on approximate idempotent range searching. This implies a similar improvement on the query time of the theorem however we will not discuss these here.

The key idea is to observe that a weight-constrained ε-NN query can be answered with the help of a carefully chosen series of ε-range maximum queries. Assume that there is a method, e.g. [1], that can answer

an ε-range maximum query in $t(n, \varepsilon)$ time. Given a query point q and a weight w, we search for a weight-constrained ε-NN to q. Suppose that we perform an ε-range maximum query with the range $b(q, r)$ for some r of our choice and that it returns a point at distance r' from q with weight w'. Note that $r' \leq (1+\varepsilon)r$. If $w' \geq w$ this implies we can limit our search for a weight-constrained ε-NN to q among the points in $b(q, r')$. Otherwise if $w' < w$ we are certain that the answer can only be found among the points in $\overline{b}(q, r)$ (since we know from the answer to the ε-NN maximum range query that all points in $b(q, r)$ have weight at most w'). When all points in P have weights less than w there is no weight-constrained ε-NN. To avoid this case we may assume that there is an auxiliary point far enough from P with infinite weight.

We get that after repeating several ε-range maximum queries with the same center q but for different values of the radius r we will have limited our search in an annulus $A = \overline{b}(q, r_1) \cap b(q, r_2)$ for some values r_1, r_2 with $r_1 < r_2$ and we will have found a point p in A satisfying the weight constraint. Observe that if $r_2 \leq (1+\varepsilon)r_1$ clearly p is a valid answer to the query pair q and w. We show next that with a careful selection of the range radii we can easily obtain an efficient solution for our problem when P has small spread. Let the *spread* Δ of a point set P be the ratio between its diameter and the distance of a closest pair of P. Clearly we can perform a binary search for the right radius using at most $O(\log \Delta)$ ε-range maximum queries and thus find a weight-constrained ε-NN in $O(t(n, \varepsilon) \log \Delta)$ time.

The drawback of the approach described above is that it depends on the spread and it pays at least a $O(\log n)$ factor for each ε-range maximum query even though all of these queries share the same center and may also have similar radii. Interestingly though using together the methods of this approach and the methods presented in [1] for answering ε-approximate range queries and particularly for answering ε-approximate kth nearest neighbor queries we can remove this drawback and arrive at the theorem. Roughly the main adaptation is that wherever an approximate range counting query is needed for approximate kth nearest neighbor searching we use instead the corresponding approximate range maximum query and we guide the search according to the point and the weight returned and not the count. A detailed proof will appear in the full version.

4 Querying about the past

We define a *timestamped* operation on a data structure as an operation which carries a label of the time it occurred. An element is *alive* at some moment t if t is between the time of insertion and deletion of the element. We consider the following problem: given a sequence of n timestamped insertions and deletions of d-dimensional points build a data structure which given a query point q and a parameter t finds efficiently an ε-NN of q among the points alive at time t. We call this a *t-moment ε-NN query.*

We tackle the problem using methods from Sec. 2 and 3. Let p be a point that was inserted at time t_s and deleted at time t_f (infinite values are allowed). We use time as an extra dimension and map the point p to the segment with endpoints (t_s, p) and (t_f, p) in $d+1$ dimensions. Note that the points alive at any given moment t correspond to the segments intersecting the hyperplane with equation $x = t$. Hence we can build a similar interval tree as in Sec. 2. To answer queries part (b) is only needed. We extend Lemma 2 in d dimensions and after a simple analysis we get:

Theorem 5 *Let n timestamped insertions and deletions of points in $\mathbb{R}^d$, and let $0 < \varepsilon < 1/2$ be a real parameter. We can construct a data structure of $O(n \log(1/\varepsilon))$ space that allows us to answer any t-moment ε-NN query in time $O(\log^2 n + \log n/\varepsilon^d)$. Here t is given at query time.*

References

[1] S. Arya, T. Malamatos, and D. M. Mount. Space-time tradeoffs for approximate spherical range counting. In *Proc. 16th Annu. ACM-SIAM Sympos. Discrete Algorithms*, pages 535–544, 2005.

[2] S. Arya, T. Malamatos, and D. M. Mount. Space-time tradeoffs for approximate nearest neighbor searching. *J. Assoc. Comput. Mach.*, 57:1–54, 2009.

[3] S. Arya, D. M. Mount, N. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *J. Assoc. Comput. Mach.*, 45:891–923, 1998.

[4] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, Germany, 3rd edition, 2008.

[5] P. Indyk. Nearest neighbors in high-dimensional spaces. In *Handbook of Discrete and Computational Geometry*, pages 877–892. CRC Press, 2004.

[6] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proc. 30th Annu. ACM Sympos. Theory Comput.*, pages 604–613, 1998.

[7] V. Koltun and M. Sharir. Polyhedral Voronoi diagrams of polyhedra in three dimensions. In *Proc. 18th Annu. ACM Sympos. Comput. Geom.*, pages 227–236, 2002.

[8] A. Magen. Dimensionality reductions in ℓ_2 that preserve volumes and distance to affine spaces. *Discrete Comput. Geom.*, 38(1):139–153, 2007.

[9] Y. Wang. Approximating nearest neighbor among triangles in convex position. *Inf. Process. Lett.*, 108(6):379–385, 2008.

Steinitz Theorems for Orthogonal Polyhedra

David Eppstein* Elena Mumford†

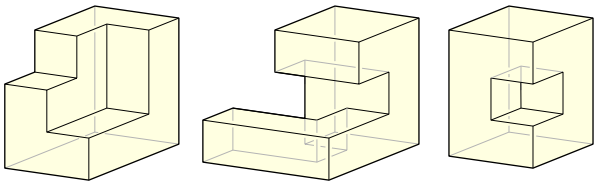


Figure 1: Three types of simple orthogonal polyhedron: Left, a corner polyhedron. Center, an *xyz* polyhedron that is not a corner polyhedron. Right, a simple orthogonal polyhedron that is not an *xyz* polyhedron.

1 Introduction

Steinitz's theorem [8, 14, 17] characterizes the skeletons of three-dimensional convex polyhedra in purely graph-theoretic terms: they are exactly the 3-vertex-connected planar graphs. In one direction, this is straightforward to prove: every convex polyhedron has a skeleton that is 3-connected and planar. The main content of Steinitz's theorem lies in the other direction, the statement that every 3-connected planar graph can be represented as a polyhedron. Steinitz's theorem, together with Balinski's theorem that every d-dimensional polytope has a d-connected skeleton [1], form the foundation stones of polyhedral combinatorics; Grünbaum writes [8] that Steinitz's theorem is "the most important and deepest known result on 3-polytopes."

However, analogous results characterizing the skeletons of other classes of polyhedra or higher dimensional polytopes have been elusive. As Ziegler [17] writes, "No similar theorem is known, and it seems that no similarly effective theorem is possible, in higher dimensions." Even in three dimensions, it remains unknown whether the complete graph K_{12} may be embedded as a genus-six triangulated polyhedral surface, generalizing the toroidal embedding of K_7 as the Császár polyhedron [3].

In this paper, we characterize another class of three-dimensional non-convex polyhedra, which we call *simple orthogonal polyhedra* (Fig. 1): polyhedra with the topology of a sphere, with simply-connected faces, and with exactly three mutually-perpendicular axis-parallel edges meeting at every vertex. We also consider two special cases of simple orthogonal polyhedra, which we call *corner polyhedra* and *xyz polyhedra*. A corner polyhedron (Fig. 1, left) is a simple orthogonal polyhedron in which all but three faces are oriented towards the vector $(1,1,1)$; it can be drawn in the plane by isometric projection with only one of its vertices hidden (the one incident to the three back faces). An *xyz* polyhedron (Fig. 1, center) is a simple orthogonal polyhedron in which each axis-parallel line contains at most two vertices. We show:

- The graphs of corner polyhedra are exactly the cubic bipartite polyhedral graphs such that every separating triangle of the planar dual graph has the same parity. Here cubic means 3-regular, polyhedral means planar 3-connected, and we define the parity of a separating triangle later. The graphs with no separating triangles form the building blocks for all our other characterizations: every cubic bipartite polyhedral graph with a 4-connected planar dual is the graph of a corner polyhedron.

- The graphs of *xyz* polyhedra are exactly the cubic bipartite polyhedral graphs.

- The graphs of simple orthogonal polyhedra are exactly the cubic bipartite planar graphs such that the removal of any two vertices leaves at most two connected components.

Based on our graph-theoretic characterizations of these classes of polyhedron, we find efficient algorithms for finding a polyhedral realization of the graph of any corner polyhedron, *xyz* polyhedron, or simple orthogonal polyhedron. Beyond the obvious applications of our results in graph drawing and architectural design, we believe that these results may have applications in image understanding, where an analysis of the structure of polyhedral and rectilinear objects has been an important subtopic [11, 12, 16].

Due to space considerations we omit the proofs of our results; they can be found in the full version of the paper [7].

2 Corner polyhedra and rooted cycle covers

We define a corner polyhedron to be a simple orthogonal polyhedron with the additional property that three faces (the *back faces*) are oriented towards the vector $(-1,-1,-1)$, and all remaining faces (the *front faces*) are oriented towards the vector $(1,1,1)$. The three back faces necessarily share a vertex, the *hidden vertex*. Parallel projection of a corner polyhedron onto a plane perpendicular to the vector $(1,1,1)$ gives rise to a drawing, the so-called *isometric projection*, in which the axis-parallel edges of the three-dimensional polyhedron are mapped to three sets of parallel lines that form angles of $\pi/3$ with respect to

*Computer Science Department, University of California, Irvine, eppstein@uci.edu

†TU Eindhoven, e.mumford@tue.nl

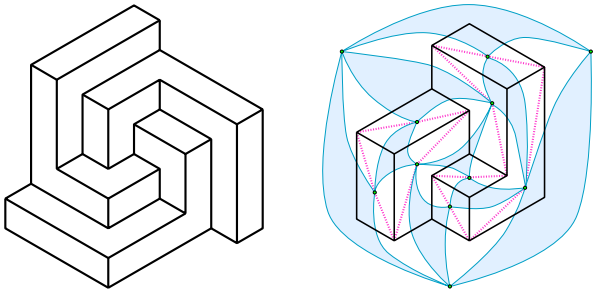


Figure 2: Left: Isometric projection of a corner polyhedron for a truncated rhombic dodecahedron. Right: connecting each sharp corner of a visible face to the dual vertex of the face produces a rooted cycle cover.

each other; see Fig. 2 (left) for an example. If the edges of the corner polyhedron have integer lengths, the resulting isometric projection is a drawing of all of the vertices of the polyhedron, except the hidden vertex, on the hexagonal lattice. It is possible to include the hidden vertex as well by connecting it to its three neighbors by lattice paths, one of which is straight and the other two have one bend each; the resulting drawing of the whole graph has two bends.

In an isometric drawing of a corner polyhedron, none of the faces can have an interior angle of $5\pi/3$, for any simple orthogonal polyhedron with such a projected angle would have more than three back faces. Additionally, each face (being the projection of a planar orthogonal polygon) has edges of only two of the three possible slopes. Therefore, each face of the drawing has the shape of a *double staircase*: there are two vertices at which the interior angle is $\pi/3$, and the two sequences of interior angles on the paths between these vertices alternate between interior angles of $2\pi/3$ and $4\pi/3$. (Conversely, it follows by Thurston's results on height functions [15] that a drawing in the hexagonal lattice for which all faces have this shape comes from a three-dimensional orthogonal surface.) Because a simple orthogonal polyhedron forms a planar graph in which all faces have an even number of edges, it must be bipartite; one of its two color classes consists of the vertices having sharp face angle, and the other color class contains all the other vertices.

The two-to-one correspondence between interior faces and vertices with sharp angles gives rise to an important structure on the graph of the polyhedron, which we find simpler to describe in terms of its dual graph. The dual graph of a corner polyhedron (as with a bipartite cubic polyhedral graph more generally) is an *Eulerian triangulation*, a maximal planar graph in which every vertex has even degree. It has a unique planar embedding, for which all the faces are triangles; the triangles may be two-colored so that the two triangles that share each edge have different colors. Within each interior face of the projected corner polyhedron, we connect the dual vertex to its two sharp corners. The result of forming these connections is a structure that we call a *rooted cycle cover*: a set of vertex-disjoint cycles in the dual Eulerian triangulation, that cover every dual vertex except for the three vertices of the *root triangle* dual to the hidden vertex, and that include exactly one edge from every triangle with the same color as the root triangle—see Fig. 2 for an illustration. Conversely, as we show, every rooted cycle cover of an Eulerian triangulation gives rise to a corner polyhedron representation of its dual graph. This equivalence between a combinatorial structure (a rooted cycle cover) and a geometric structure (a corner polyhedron) is a key component of our characterization of the graphs of corner polyhedra.

Specifically, we prove the following results:

Theorem 1 *A graph $\mathcal{G}$ can be represented as a corner polyhedron, with a specified vertex v as the single hidden vertex, if and only if the dual graph of $\mathcal{G}$ has a cycle cover rooted at the triangle dual to v.*

Theorem 2 *If $\mathcal{G}$ is a cubic bipartite polyhedral graph with a 4-connected dual, then it can be represented as a corner polyhedron.*

If Δ is an Eulerian triangulation (the dual to a cubic bipartite planar graph), with a chosen root triangle δ, then we may uniquely two-color the triangles of Δ so that any two adjacent triangles are adjacent. For any separating triangle γ of Δ, this coloring will assign equal colors to the three triangles that are on the side of γ that does not contain δ and that are incident to one of the edges of γ. We say that γ has *even parity* if these three triangles have the same color as δ, and *odd parity* otherwise.

Theorem 3 *If $\mathcal{G}$ is a cubic bipartite graph with a non-4-connected dual, and v is any vertex of $\mathcal{G}$, then $\mathcal{G}$ has a corner representation for which v is the hidden vertex if and only if all separating triangles have odd parity with respect to the root triangle dual to v.*

3 *xyz* polyhedra

In our previous paper [6] we defined an *xyz graph* to be a cubic graph embedded in three dimensional space, with axis parallel edges, such that the line through each edge passes through no other vertices of the graph. We can extend this definition to an *xyz polyhedron*, a simple orthogonal polyhedron whose skeleton forms an *xyz* graph. Alternatively, we can consider a weaker definition: a *singly-intersecting* simple orthogonal polyhedron is a simple orthogonal polyhedron with the property that, for any two faces with a nonempty intersection, their intersection is a single line segment. Geometrically, the intersection of two faces lies along the line of intersection of their planes, so a singly-intersecting polyhedron must be an *xyz* polyhedron, but not necessarily vice versa. However, the graphs of the two classes of polyhedra are the same: by perturbing the face planes of a singly-intersecting polyhedron, one may obtain an *xyz* polyhedron that represents the same graph.

As we showed in [6], a planar *xyz* graph must be 3-connected and bipartite, and the same results hold for *xyz* polyhedra. Our main result is a converse to this:

Theorem 4 *The following three classes of graphs are equivalent: (1) Cubic 3-connected bipartite planar graphs; (2) Graphs of xyz polyhedra, and (3) Graphs of singly-intersecting simple orthogonal polyhedra.*

4 Simple orthogonal polyhedra

As can be seen in Fig. 1 (right), the graph of an arbitrary simple orthogonal polyhedra may not always be 3-connected, although it is always 2-connected. Pairs of faces of the polyhedron may meet in multiple edges, and the removal of any two of these edges (or the removal of endpoints from any two of these edges) leaves a disconnected graph. Therefore, there exist graphs simple orthogonal polyhedra that are not graphs of *xyz* polyhedron, and we need to use a more general class of graphs to characterize the simple orthogonal polyhedra. Replacing the 3-connectivity condition in the characterization of *xyz* polyhedra by 2-connectivity would be too general, however. Not every 2-connected bipartite 3-regular graph is the graph of a simple orthogonal polyhedron; for instance, the graph depicted in Fig. 2 is not the graph of a simple orthogonal polyhedron, as the results in this section will show.

Instead, our characterization uses the SPQR tree, a standard tool for representing the planar embeddings of a graph in terms of its triconnected components [4, 5, 9, 10, 13].

Theorem 5 *The following three classes of graphs are equivalent: (1) Cubic 2-connected graphs in which every triconnected component is either a bipartite polyhedral graph or an even cycle; (2) Bipartite cubic planar graphs in which the removal of any two vertices leaves at most two connected components (counting an edge between the two vertices as a component, if one exists), and (3) Graphs of simple orthogonal polyhedra.*

Our proof technique leads to a stronger result: if $\mathcal{G}$ is the graph of a simple orthogonal polyhedron, then every planar embedding of $\mathcal{G}$ can be represented as a simple orthogonal polyhedron.

5 Algorithms

Below we outline an algorithm that takes a 2-connected cubic planar graph as an input and embeds it as a simple orthogonal polyhedron, when such a representation exists. The algorithms for taking as input a 3-connected graph and representing it either as an *xyz* polyhedron or as a corner polyhedron, when such a representation exists, are similar but with fewer steps.

1. Decompose the graph into its triconnected components, as represented by an SPQR tree, in $O(n)$ time [9, 10]. Check that the SPQR tree does not contain any P nodes (triconnected components that are multigraphs rather than simple graphs). If it does, report that no orthogonal polyhedral representation exists and abort the algorithm.

2. Transform each atom (triconnected component that is not a cycle) into its dual Eulerian triangulation. If any atom is nonplanar or has a non-Eulerian dual, report that no orthogonal polyhedral representation exists and abort the algorithm.

3. Partition each Eulerian triangulation into 4-connected Eulerian triangulations by splitting it on its separating triangles.

4. Recursively decompose each 4-connected Eulerian triangulation into simpler 4-connected Eulerian triangulations using separating 4-cycles, pairs of adjacent degree-4 vertices, and isolated degree-4 vertices. While returning from the recursion, undo the steps of the decomposition and build a cycle cover for the Eulerian triangulation.

5. Convert the cycle covers into regular edge labelings by a simple local pattern matching rule.

6. For each pair of colors x and y in the rainbow partition, construct the subgraph Δ_{xy} formed by edges with those two colors, oriented by reversing the orientations of one of the two colors from the orientation given by the regular edge labeling, and find an st-numbering of each such graph using breadth-first search.

7. For each graph dual to one of the 4-connected Eulerian triangulations, use the st-numbering to construct a representation of the graph as a corner polyhedron: the coordinates of each vertex of the corner polyhedron are triples of numbers from the st-numbering, one from each of the three bichromatic subgraphs of Δ.

8. Glue the corner polyhedra together to form orthogonal polyhedra dual to each non-4-connected Eulerian triangulation.

9. Glue 3-connected polyhedra together to form arbitrary simple orthogonal polyhedra.

Theorem 6 *We may construct a representation of a given graph as a corner polyhedron, xyz polyhedron, or simple orthogonal polyhedron, when such a representation exists, in $O(n)$ randomized expected time, or deterministically in $O(n(\log\log n)^2/(\log\log\log n))$ time with linear space.*

The detailed descriptions of each step together with the running time analyses are given in [7].

6 Conclusions

We have defined three interesting classes of orthogonal polyhedra, and provided exact graph-theoretic characterizations of the graphs that may be represented by these polyhedra. In particular, every bipartite cubic polyhedral graph has a representation as an orthogonal polyhedron.

The following problems remain open for additional investigation:

- Is there a simple condition on the position of the dual separating triangles that characterizes orthogonally convex simple orthogonal polyhedra, and can we test this condition in polynomial time?

- The *orthostacks* defined by Biedl et al. [2] are also intermediate between corner polyhedra and *xyz* polyhedra. Can we characterize their graphs?

- Given the hardness of nonplanar *xyz* graph recognition [6] it seems likely that it will also be difficult to determine whether a given graph is the graph of an orthogonal polyhedron with nonzero genus but what about graphs for which an *xyz* graph representation is already known? In that case, how difficult is it to determine whether the faces of the *xyz* representation can be untangled to form a polyhedral representation?

Acknowledgements

Work of David Eppstein was supported in part by NSF grant 0830403 and by the Office of Naval Research under grant N00014-08-1-1015. We thank Patrizio Angelini, Mike Dillencourt, Fabrizio Frati, Mike Goodrich, Maarten Löffler, and Brendan McKay for helpful conversations related to the subject of this paper.

References

[1] M. L. Balinski. On the graph structure of convex polyhedra in *n*-space. *Pacific J. Math.* 11(2):431–434, 1961.

[2] T. Biedl, E. Demaine, M. Demaine, A. Lubiw, M. Overmars, J. O'Rourke, S. Robbins, and S. Whitesides. Unfolding some classes of orthogonal polyhedra. *Proc. 10th Canadian Conference on Computational Geometry (CCCG'98)*, 1998.

[3] A. Császár. A polyhedron without diagonals. *Acta Sci. Math. Szeged* 13:140–142, 1949.

[4] G. Di Battista and R. Tamassia. Incremental planarity testing. *Proc. 30th Symp. Foundations of Computer Science (FOCS 1989)*, pp. 436–441, 1989.

[5] G. Di Battista and R. Tamassia. On-line graph algorithms with SPQR-trees. *Proc. 17th Internat. Colloq. Automata, Languages and Programming (ICALP 1990)*, pp. 598–611. Springer-Verlag, LNCS 443, 1990.

[6] D. Eppstein. The topology of bendless three-dimensional orthogonal graph drawing. *Proc. 16th Int. Symp. Graph Drawing (GD 2008)*, pp. 78–89. Springer-Verlag, LNCS 5417, 2008, arXiv:0709.4087.

[7] D. Eppstein and E. Mumford. Steinitz Theorems for Orthogonal Polyhedra. Electronic preprint arXiv:0912.0537v1, 2009.

[8] B. Grünbaum. *Convex Polytopes*. Graduate Texts in Mathematics 221. Springer-Verlag, 2nd edition, 2003.

[9] C. Gutwenger and P. Mutzel. A linear time implementation of SPQR-trees. *Proc. 8th Int. Symp. Graph Drawing (GD 2000)*, pp. 77–90. Springer-Verlag, LNCS 1984, 2001.

[10] J. Hopcroft and R. Tarjan. Dividing a graph into triconnected components. *SIAM J. Comput.* 2(3):135–158, 1973.

[11] D. A. Huffman. Impossible objects as nonsense sentences. *Machine Intelligence 6*, pp. 295–323. Edinburgh University Press, 1971.

[12] L. M. Kirousis and C. H. Papadimitriou. The complexity of recognizing polyhedral scenes. *Journal of Computer and System Sciences* 37(1):14–38, 1988.

[13] S. Mac Lane. A structural characterization of planar combinatorial graphs. *Duke Math. J.* 3(3):460–472, 1937.

[14] E. Steinitz. Polyeder und Raumeinteilungen. *Encyclopädie der mathematischen Wissenschaften, Band 3 (Geometries)*, pp. 1–139, 1922.

[15] W. P. Thurston. Conway's tiling groups. *Amer. Mathematical Monthly* 97(8):757–773, 1990.

[16] D. Waltz. Understanding line drawings of scenes with shadows. *The Psychology of Computer Vision*, pp. 19–91. McGraw-Hill, 1975.

[17] G. M. Ziegler. *Lectures on Polytopes*. Graduate Texts in Mathematics 152. Springer-Verlag, Berlin, Heidelberg, and New York, 1995.

Evacuation of rectilinear polygons

Sándor Fekete* Chris Gray* Alexander Kröller*

Abstract

We investigate the problem of creating fast evacuation plans for buildings that are modeled as grid polygons, possibly containing exponentially many cells. We study this problem in two contexts: the "confluent" context in which the routes to exits remain fixed over time, and the "non-confluent" context in which routes may change. Confluent evacuation plans are simpler to carry out, as they allocate contiguous regions to exits; non-confluent allocation can possibly create faster evacuation plans. We give results on the hardness of creating the evacuation plans and a strongly polynomial algorithm for finding confluent evacuation plans when the building has two exits. We also give a pseudo-polynomial time algorithm for non-confluent evacuation plans.

1 Introduction

A proper evacuation plan is an important requirement for the health and safety of all people inside a building. When we optimize evacuation plans, our goal is to remove people from a building as quickly as possible. In the best case, each of a building's exits would serve an equal number of the building's inhabitants. However, there might be cases in which this can not happen.

In this research, we study the computation of evacuation plans for buildings that are modeled as grid polygons. We make the assumption that every grid square is occupied by exactly one person and that at most one person can occupy a grid square at any given time. The first assumption may seem a bit contrived, but in many cases it is impossible for a building designer to know exactly where people will be in the building in the moments before an evacuation, and this pessimistic view of the situation is the only sensible one to take. Also remember that in some cases, such as in airplanes, the situation in which nearly every bit of floor space is occupied before an evacuation is more common than the alternative.

Evacuation plans can be divided into two distinct types. In the first, signs are posted that direct every person passing them to a specific exit. In the second, every person is assigned to a distinct exit that does not necessarily depend on the exits to which his or her neighbors are assigned. The first type of evacuation plan generates what is known as a *confluent flow* and the second generates a *non-confluent flow*. More precise definitions of these terms will be given later.

We first show that the problem of finding an optimal confluent flow belongs to the class of NP-complete problems if the polygon has "holes"—i.e., if it represents a building with completely enclosed rooms or other spaces. We then give an algorithm with a running time linear in the description complexity of the region (which can be exponentially smaller than the number of cells) that computes an evacuation plan for buildings without holes that have two exits; a generalization to a constant number of exits is more complicated, but seems plausible.

1.1 Preliminaries

We are given a rectilinear polygon P on a grid. There exists, on the boundary of P, a number of special grid squares known as *exits*. We call the set of exits $\mathcal{E} = \{e_1, \ldots, e_k\}$. We assume that every grid square in P contains a person. A person can move vertically or horizontally into an empty grid square or an exit. The goal is to get each person to an exit as quickly as possible. When an exit borders more than one grid square, we specify which squares people can enter the exit from.

The area of P is denoted by A and the number of vertices of P is denoted by n. Note that A can be exponential in n. The set of people that leave P through the exit e is called the *e-exit class*. We also write *exit class* to refer to the set of people who leave through an unspecified exit. The grid squares that are adjacent to the boundary of P are known as *boundary squares*.

There are two versions of the problem that we consider. We call these *confluent flows* and *non-confluent flows*. In the first, we add the restriction that every grid square has a unique successor. Thus, for every grid square s, people passing through s leave s in only one direction. This restriction implies that evacuation plans are determined by space only. It does not exist for non-confluent flows. It can be argued that informing people which exit to use is easier in the case of confluent flows since a sign can be placed in every grid square, informing the people who pass through it which exit to use. However, we show in the full

*Department of Computer Science, TU Braunschweig, Germany. {fekete,gray,kroeller}@ibr.cs.tu-bs.de This research was funded by the German Ministry for Education and Research (BMBF) under grant number 03NAPI4 "ADVEST", which fully funded Chris Gray.

version of the paper that non-confluent flows can lead to significantly faster evacuations.

The major difficulty in the problem comes from bottlenecks. We define a *k-bottleneck* to be a rectangular subpolygon B of P such that two parallel boundary edges of B are the same as two edges of P, where the distance between the common edges is k.

1.2 Related Work

The problem when restricted to confluent flows is similar in many ways to the unweighted Bounded Connected Partition (or 1-BCP) problem [8]. It has been shown independently by Lovász [6] and Györi [5] that a solution to the 1-BCP problem can be found for every graph that is k-connected. However, their proofs are not algorithmic.

Unfortunately, our graph can be 1-connected in the case in which there are 1-bottlenecks. Also, since the dual of the grid contained in our polygon can have size exponential in the complexity of the polygon, we would probably need to merge nodes and then assign weights to the nodes of the newly-constructed graph. The addition of weights, however, makes the BCP problem NP-complete, even in the restricted case in which the graph is a grid [3].

Given that our problem is on a grid, it can be seen as a "discrete" problem. The "continuous" version of our problem—that is, splitting polygons into subpolygons of equal area—has also been studied. One interesting result from this study is that finding such a decomposition while minimizing the lengths of the segments that do the partitioning is NP-hard even when the polygons are orthogonal [1]. However, polynomial algorithms exist for the continuous case when that restriction is removed [7].

Baumann and Skutella [2] consider evacuation problems modeled as earliest-arrival flows with multiple sources. They achieved a strongly-polynomial-time algorithm by showing that the function representing the number of people evacuated by a given time is submodular. Such a function can be optimized using the parametric search technique of Megiddo. Their approach is different from ours in that they are given an explicit representation of the flow network as input. We are not given this, and computing the flow network that is implicit in our input can take exponential time. Also, their algorithm takes polynomial time in the sum of the input and output sizes. However, the complexity of the output can be exponential in the input size.

2 Confluent Flows

As mentioned in the introduction, in a confluent flow, every grid square has the property that all people that pass through it use the same exit.

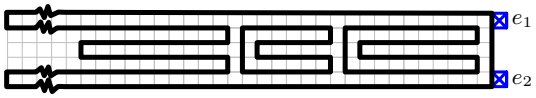


Figure 1: The polygon P given a PARTITION instance of $\{11, 6, 9\}$. To keep the picture a manageable size, the elements have not been scaled and the left ends of the first and fifth rows are truncated.

In this section, we present our results related to confluent flows. First, we show the NP-completeness of the problem of finding an optimal evacuation plan with confluent flows in a polygon with holes. This holds even for polygons with two exits. We then give a linear-time algorithm for polygons with two exits.

2.1 Hardness

Weak NP-hardness with two exits. We first show that the evacuation problem with confluent flows is NP-hard if we allow P to have holes. We reduce from the problem PARTITION, which is well-known to be NP-complete [4]. In this problem, we are given a set $S = \{c_1, c_2, \ldots, c_m\}$ of integers and we are asked to determine whether we can find $S_1, S_2 \subseteq S$ such that $\sum_{c_h \in S_1} c_h = \sum_{c_i \in S_2} c_i$.

We note that if we scale all of the numbers in a PARTITION instance by an integer ℓ, the answer remains the same—that is, a partition can be found in the new set if and only if a partition could be found in the old set—but the difference between the size of non-optimal sets is at least ℓ. This is because

$$\sum_{c_h \in S_1} \ell c_h - \sum_{c_i \in S_2} \ell c_i = \ell \left(\sum_{c_h \in S_1} c_h - \sum_{c_i \in S_2} c_i \right) \geq \ell$$

if the sums are not equal.

To transform the PARTITION problem into our problem, we first scale the input by a factor of $2m+1$. We then do the following to make the polygon P.

We first make a rectangle whose width is $m + 1 + \sum_{c_i \in S} c_i$ and whose height is 5. We then remove all the grid squares of P on the second and fourth rows except those that are at position $\sum_{i=1}^{j} c_i + j$ for all $0 < j \leq m$. After this, we remove all grid squares from the third row that are at position $\sum_{i=1}^{j} c_i + j + 1$ for all $0 < j \leq m$. Then we add a large number of squares (at least equal to the current area of P) to the left end of the first and fifth rows. Finally, we add an exit e_1 to the right end of the first row and an exit e_2 to the right end of the fifth row.

See Figure 1 for a small example. We say that the connected sets of grid squares in the third row each correspond to one of the elements of the given PARTITION instance. This leads to the following.

Lemma 1 *The polygon P with holes can be divided into two confluent exit classes of equal size if and only if the given instance of* PARTITION *has a "yes" answer.*

We define the decision version of the problem of evacuation with confluent flows to be: "Given a grid polygon P with k exits and a natural number ℓ, can a confluent flow be found in which the largest exit class has size at most ℓ?"

Since areas of polygons can be computed in time proportional to their number of vertices, we can verify if a solution is correct in $O(kn)$ time (where n is the number of vertices of the polygon and k is the number of exits). This, along with Lemma 1, implies that the decision problem is NP-complete in polygons with holes. We summarize this result in the following theorem.

Theorem 2 *The problem of finding an optimal confluent flow in a polygon with holes is NP-complete.*

In the full version of the paper, we also prove that finding an optimal confluent flow in a polygon with holes is *strongly NP-complete* (meaning that the size of the numbers does not affect the hardness of the problem). However, the proof requires that the polygon have $O(n)$ exits.

Theorem 3 *The problem of finding an optimal confluent flow in a polygon with holes and $O(n)$ exits is strongly NP-complete.*

2.2 Algorithm for Simple Polygons with Two Exits

When the polygon P with no holes has only two exits, e_1 and e_2, we can find an optimal confluent flow in $O(n)$ time. We give an algorithm that uses to the *rotating calipers* paradigm [10].

We begin by computing a subset ω' of the overlay ω of the vertical and horizontal decompositions of P. This subset contains only the rectangles that touch the boundary of P. This subset can be computed in linear time.

We create two pointers i_1 and i_2 with which we walk through the intervals in ω'. For each pair of intervals pointed to by i_1 and i_2 that we visit, we measure the number of squares that must be in the e_1- and e_2-exit classes if we assume that the endpoints of their connections to the boundary of P begin and end in i_1 and i_2. We call these areas A_1 and A_2 respectively. If we ever visit a pair of intervals for which A_1 and A_2 are both less than $A/2$, then we divide the remaining squares so that both A_1 and A_2 are $A/2$—see the full paper for details—and return the results. Otherwise, we return the exit class that has size greater than $A/2$, but whose size is minimal.

To begin with, we set i_1 and i_2 to be the interval containing e_2, so that the endpoints of the connection of the e_1-exit class are on either side of e_2. We call the area that the e_1-exit class must have A_1 and the area that the e_2-exit class must have A_2. We then move i_1 closer to e_1 until it either reaches e_1 or would cause A_2 to be greater than $A/2$. As we progress, we simply update A_1 and A_2 and keep track of the smallest value for A_1.

Once we have done this for i_1, we do the same for i_2. Finally, we move i_1 back towards e_2. For each interval that we move i_1 towards e_2, we move i_2 as much as possible towards e_1 so that A_1 is as small as possible without causing A_2 to be larger than $A/2$. As before, we keep track of the smallest value for A_1. When i_1 reaches e_2 or i_2 reaches e_1, we stop.

When we have completed the algorithm for e_1, we repeat the process, switching e_1 and e_2. This gives us the following theorem that we prove in the full paper.

Theorem 4 *In the confluent setting, the above algorithm finds the optimal evacuation plan for a polygon P with two exits in $O(n)$ time, where n is the number of vertices in P.*

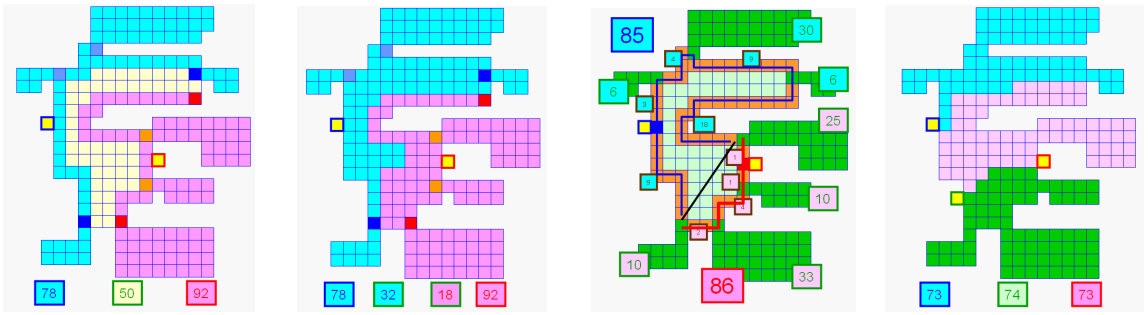
We conjecture that one can use algorithms similar to the one given above to compute the evacuation of any polygon with a constant number of exits, but the details become much more involved. We therefore leave this question to future work.

3 Pseudo-Polynomial Algorithm for Non-Confluent Flows

Compared with confluent flows, non-confluent flows are clearly a stronger model. We note that any confluent flow is a non-confluent flow, but not *vice versa*.

In contrast to the case with confluent flows, for which we showed that finding an assignment of people to exits is strongly NP-complete when we are dealing with polygons with holes and $O(n)$ exits, we can show that, for non-confluent flows, a pseudo-polynomial algorithm exists.

The algorithm is based on the technique of using time-expanded networks to compute flows over time [9]. Therefore, we compute a flow network from the input polygon as follows. We create a source vertex s and a sink vertex t. For each grid square in P, we create two vertices—an *in* vertex and an *out* vertex. We connect the in vertex to the out vertex with an edge that has capacity 1 for every grid square. We then make, for some integer $T \geq 1$, T copies of the polygon $P_1, \ldots, P_T$, where each copy has these vertices and edges added. For every grid square of P_1, we connect s to the in vertex of the grid square with an edge that has capacity 1. We then connect the out vertex of every grid square in P_i to the in vertex of all its neighbors in P_{i+1} for all $1 \leq i \leq T-1$. Again, the edges we use all have capacity 1. Finally, we connect



(a) (b) (c) (d)

Figure 2: (a) Assigning corner pixels to exits induces a split that leaves some pixels unassigned. (b) Splitting the unassigned pixels optimally. (c) The idea for the linear-time algorithm. (d) Illustrating a solution for $k = 3$ exits.

the out vertex of every exit to t with an edge that has capacity 1. We call this flow network G.

It is fairly easy to see that if we are able to find a maximum flow of value A through G, then we are able to evacuate P in T time steps. However, we note that both T and $|G|$ can be exponential in the complexity of P, making this a pseudo-polynomial algorithm.

Theorem 5 *There exists a pseudo-polynomial algorithm to find an evacuation of a polygon with a non-confluent flow.*

4 Conclusions

We have discussed evacuations in grid polygons. We first showed that finding evacuations with confluent flows in polygons with holes is hard, even for polygons with only two exits. We then looked at algorithms to find evacuations with confluent flows.

Our work raises some questions that require further study. For simple polygons, there is evidence that a constant number of exits allows strongly polynomial solutions, even though some of the technical details are complicated. What is the complexity of finding an evacuation plan with a confluent flow when the number of exits is not constant? Can we find a polynomial algorithm that gives the optimal evacuation using non-confluent flows? Note that it is not even clear that the output size of such an algorithm is always polynomial. Finally, we conjecture that the worst-case ratio between the evacuation times for confluent flows and non-confluent flows for polygons with k exits is $2 - \frac{1}{k}$.

Acknowledgments

We thank Estie Arkin, Michael Bender, Joe Mitchell, and Martin Skutella for helpful discussions; Martin Skutella is also part of ADVEST.

References

[1] H. Bast and S. Hert. The area partitioning problem. In *Proc. 12th Can. Conf. Comput. Geom. (CCCG '00)*, pages 163—171, Fredericton, NB, Canada, 2000.

[2] N. Baumann and M. Skutella. Earliest arrival flows with multiple sources. *Math. Oper. Res.*, 34(2):499–512, 2009.

[3] R. Becker, I. Lari, M. Lucertini, and B. Simeone. Max-min partitioning of grid graphs into connected components. *Networks*, 32(2):115–125, 1998.

[4] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[5] E. Györi. On division of graphs to connected subgraphs. In *Combinatorics*, pages 485–494, Keszthely, 1978.

[6] L. Lovász. A homology theory for spanning trees of a graph. *Acta Mathematica Hungarica*, 30(3):241–251, 1977.

[7] V. Lumelsky. Polygon area decomposition for multiple-robot workspace division. *Int. Journal of Computational Geometry and Applications*, 8(4):437–466, 1998.

[8] L. R. Salgado and Y. Wakabayashi. Approximation results on balanced connected partitions of graphs. *Electr. Notes in Discrete Mathematics*, 18:207–212, Dec. 2004.

[9] M. Skutella. An introduction to network flows over time. In *Research Trends in Combinatorial Optimization*, pages 451–482. Springer-Verlag, 2009.

[10] G. Toussaint. Solving geometric problems with the rotating calipers. *In Proc. IEEE MELECON '83*, pages 10—02, 1983.

Robot Swarms for Exploration and Triangulation of Unknown Environments

Sándor P. Fekete* Tom Kamphans*† Alexander Kröller* Christiane Schmidt*‡

Abstract

We consider a robot swarm in an unknown polygon. All robots have only a limited communication range. We then look for a triangulation of the polygon using the robots as vertices such that the number of robots used for the triangulation is minimized. All edges in the triangulation have a length smaller than the communication range. For this *Online Minimum Relay Triangulation Problem*, we present a lower bound of $\frac{9}{8}$ on the competitive ratio for any online algorithm. Moreover, we give an algorithm that is $\frac{21}{4}$-competitive for simple polygons and 6-competitive for general polygons.

1 Introduction

Exploring and guarding polygonal regions are classical problems that have been investigated for decades. Hoffmann et al. [6] considered the online exploration of simple polygons with unlimited vision, Icking et al. [7] and Fekete et al. [5] exploration with limited and time-discrete vision, respectively. Exploration with both limited and time-discrete vision is presented by Fekete et al. [4]. Placing stationary guards was first considered by Chvátal [1], see also O'Rourke [8].

In this paper, we combine both problems, motivated as follows. Consider a static sensor network that needs to react to different scenarios by adding further mobile sensors, e.g. sensor nodes attached to mobile robots as in Figure 1. Typically, these sensor nodes have a limited communication range, and no common orientation or coordinate system is available; furthermore, the expanded network has to be well connected, asking for a triangulated network.

Classical triangulation problems (see, e.g., [8, 2]) ask for a triangulation of all vertices of a polygon, but allow arbitrary length of the edges in the triangulation. This differs from our problem, in which a limitation on the edge length (given by the communication length) is given. Placing vertices of the triangulation

Figure 1: A robot swarm consisting of iRobot Roombas.

(robots) on arbitrary positions in the polygon in order to achieve this limited length is closely related to the *relay placement problem*, in which a set of sensors is to be connected by relays with limited range [3]. We use the terms *robots* and *relays* synonymously.

The rest of the paper is organized as follows. The following Section 2 provides definitions. We present a lower bound on the competitive ratio in Section 3. In Section 4 we describe a 6-competitive algorithm for polygons (with holes) and prove that this algorithm is $\frac{21}{4}$-competitive for simple polygons. In the final Section 5 we discuss possible implications and extensions.

2 Notation and Preliminaries

We are given an (unknown) polygon P with n vertices. The length of P's boundary is denoted by D (in case of a simple polygon, D is the perimeter of P).

Every robot in the swarm has a (circular) communication range r. Within this range, each robot can perceive other robots and communicate with them. For the ease of description we assume that r is equal to 1 (and scale the polygon accordingly).

Given an unknown polygon P, the *Online Minimum Relay Triangulation Problem* (OMRTP) asks for a triangulation of P that covers P. The triangulation must not contain edges crossing the boundary of P, reflecting the impossibility to communicate through walls. The triangulation therefore contains all vertices of P, plus a number of relay points. The latter are needed because edges in the triangulation must not have a length exceeding r. The objective is to minimize the number of robots; that is, vertices of

*Algorithms Group, Braunschweig Institute of Technology, Germany. Email: {s.fekete, t.kamphans, a.kroeller, c.schmidt}@tu-bs.de, http://www.ibr.cs.tu-bs.de/alg

†Supported by 7th Framework Programme contract 215270 (FRONTS).

‡Supported by DFG Focus Program "Algorithm Engineering" (SPP 1307) project "RoboRithmics" (Fe 407/14-1).

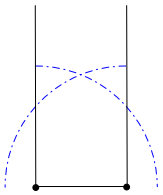


Figure 2: The polygonal corridor of width 3/4. The dash-dotted lines indicate parts of the transmission ranges of the relays located at the vertices.

the triangulation. This is equivalent to minimizing the number of relays. Let R_{OPT} denote the number of relays used by the optimum.

The robots start from a given point located at the boundary of the unknown environment. Each robot is allowed to move through the area, and will then decide a new location for a vertex of the triangulation. There it stops moving and becomes part of the static triangulation. This is motivated in the application: It is desirable to partially fix the triangulation as it is constructed, to begin location services in this area even if the polygon is not fully explored yet. This is a crucial property if we assume a huge area that is explored over long times, determined by the rate of new robots being added to the system.

3 Lower Bound

For the lower bound we use a polygonal corridor of width 3/4, see Figure 2. For a complete triangulation, relays must be placed at the vertices, i.e., the position of the first two relays is fixed.

In case the algorithm places the next relay on the boundary (w.l.o.g. we assume that it places the relay on the right boundary, otherwise a mirrored construction is used), the polygonal corridor will be constructed as depicted in Figure 3(b). Hence, the optimum needs 8 relays, the algorithm uses 9.

If, on the other hand, the algorithm locates the next relay in the center, see Figure 4(a), the polygonal corridor will be constructed as depicted in Figure 4(b). Consequently, the optimum needs again 8 relays for the triangulation, while the algorithm uses 9.

An arbitrary number of these polygonal pieces can be joined using small triangular structures as depicted in Figures 3(b), 3(c), 4(b) and 4(c), as the vertices require relays. Thus, we have:

Theorem 1 *No deterministic algorithm for the online minimum relay triangulation problem can be better than $\frac{9}{8}$-competitive.*

4 Online Triangulation

In the following, we describe our algorithm for the online minimum relay triangulation problem. We split

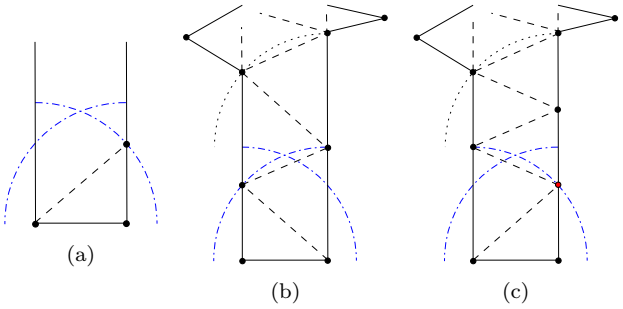


Figure 3: In case the algorithm places the next relay on the boundary (w.l.o.g. on the right boundary) (a), the optimum needs 8 relays (b), the algorithm 9 (c). The dashed lines indicate the edges of the triangulation.

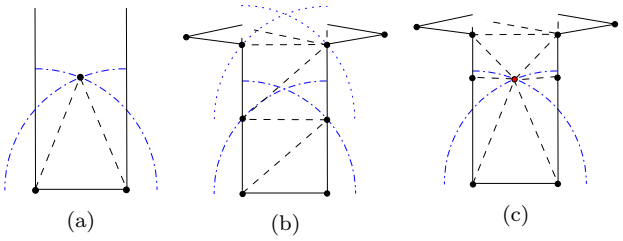


Figure 4: In case the algorithm places the next relay in the center (a), the optimum needs 8 relays (b), the algorithm 9 (c). The dashed lines indicate the edges of the triangulation.

the construction in two parts: (i) a triangulation along P's boundary and (ii) a triangulation of the interior.

For the boundary (i.e., the polygon's outer boundary and the boundary of holes in the environment) we place relays within distance 1 along the boundary and on vertices. Furthermore, we add a "second layer" along the boundary by placing relays within a distance of (at most) $\frac{\sqrt{3}}{2}$ to the boundary, and within distance of (at most) 1 to the relays located on the boundary, see Figure 5. Assuring this triangulated layer of width $\frac{\sqrt{3}}{2}$ also at vertices, we need to have a closer look at reflex vertices. The critical case for placing many relays arises from a reflex vertex with interior angle close to 360°, see Figure 7. Thus, the maximum number of additional relays located at a reflex vertex is 3, while we do not need any additional relays at non-reflex vertices. So, we add at most $3n$ relays. Consequently, the triangulation along P's boundary does not use more than $2D + 3n$ relays.

We still need to take care of (ii): a triangulation of the interior. Parts of the polygon of width less than $\sqrt{3}$ are already covered, so we do not need to take these into account, see Figure 8. For the remaining polygon we use a triangular point grid with side length 1, see Figure 6. In case points of this grid would lie inside the triangulated $\frac{\sqrt{3}}{2}$-layer along P's boundary, they get dragged outside of this layer, as depicted in Figure 9, in order to assure that the re-

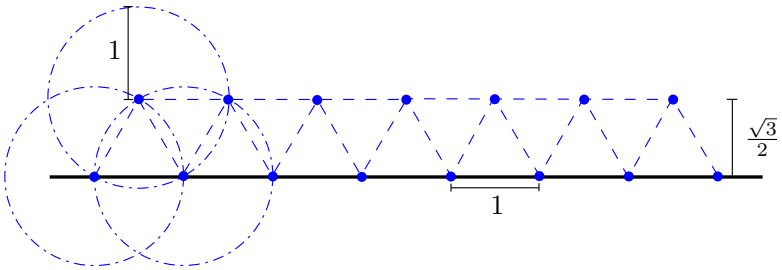


Figure 5: Example for the triangulation along the boundary, with cost at most $2D$. The bold line represents the boundary of P, the dashed lines indicate the edges of the triangulation, the dashed-dotted lines indicate again the communication range for some relays.

sulting two parts can be glued into one triangulation (i.e., in order to assure a distance of at most $r = 1$). In case points of this triangular point grid would coincide with the points from the boundary construction, they will simply not be placed, so that no degenerate triangles occur. Let k be the number of relays used for this construction.

Altogether, we have:

$$R_{\text{ALG}} \leq k + 2D + 3n \tag{1}$$

On the other hand, we can establish lower bounds on the number of relays for an optimal solution for OMRTP (i.e., lower bounds for R_{OPT}).

First, an easy observation yields the lower bound in (2): For a complete triangulation of P we need to place a relay on every vertex of the polygon.

$$R_{\text{OPT}} \geq n \tag{2}$$

Moreover, the triangulation needs to establish edges along all edges of the polygon P. As the maximum distance of relays is $r = 1$, we have:

$$R_{\text{OPT}} \geq D \tag{3}$$

As described earlier, k is the number of relays used when overlaying P with a triangular point grid with side length 1 such that all points are located inside of P (i.e., in the interior, on edges or on vertices), see

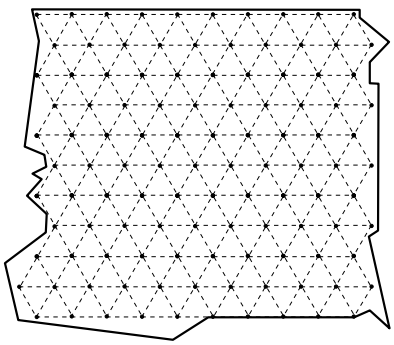


Figure 6: Example for a cover of the interior of P, using k relays.

Figure 6. Obviously, k is not uniquely defined, but for any such overlay the optimum cannot use less than k relays to triangulate P, i.e., we get a lower bound of

$$R_{\text{OPT}} \geq k \tag{4}$$

Combining Equations 1–4 yields:

Theorem 2 *There is a 6-competitive strategy for the online minimum relay triangulation problem in polygons (even with holes).*

Note that we can find the places for relays in an online fashion: From the given starting point, relays move along the boundary, assuring the placement of the relays for the triangulation along P's boundary. Then, again starting from the given start point, an overlay with a triangular point grid is constructed. local adjustments of the type in Figure 9 assure the placements of the relays in accordance with the strategy. When a hole is encountered during the construction of the triangular grid for the interior the boundary, a $\frac{\sqrt{3}}{2}$-layer is constructed around the hole.

Simple Polygons. We can achieve a factor better than 6 for simple polygons by a more careful analysis of the relays that we place at vertices. Let n_{240} be the number of reflex vertices whose interior

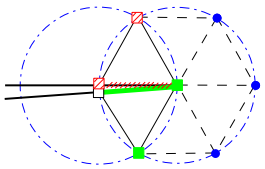


Figure 7: Example for the triangulation at a 360° reflex vertex. The black semi-bold lines represent the boundary of P (for clarity the two boundary lines are not drawn parallel but one is slightly offset from its actual position). The circular points are relays charged to the reflex vertex. Squares get charged to the perimeter—e.g., the shaded squares to the shaded piece of boundary and the filled squares to the bold piece of boundary. At most 3 additional relays are used.

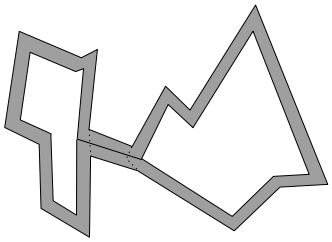


Figure 8: A polygon P. The layer of width $\frac{\sqrt{3}}{2}$ is depicted in gray.

angle, α_i, is greater than 240° and let n_{nr} be the number of non-reflex vertices. For the sum of the interior angles in a simple polygon, $\Sigma\alpha_i$, we have: $\Sigma\alpha_i = (n-2)\cdot 180° < n \cdot 180°$. This equation yields a bound on n_{240}:

$$\begin{aligned} & \Sigma\alpha_i && \geq n_{240}\cdot 240° + n_{\text{nr}}\cdot 0° \\ & && \quad +(n - n_{\text{nr}} - n_{240})\cdot 180° \\ \Leftrightarrow\ & (n-2)\cdot 180° && \geq (n-n_{\text{nr}})\cdot 180° + n_{240}\cdot 60° \\ \Leftrightarrow\ & (n_{\text{nr}}-2)\cdot 3 && \geq n_{240} \\ \Rightarrow\ & n_{240} && \leq 3\cdot n_{\text{nr}} \end{aligned}$$

For each of the n_{240} reflex vertices we need at most 3 additional relays, for the n_{nr} non-reflex vertices no additional relays are placed. For each of the remaining $(n-n_{\text{nr}}-n_{240})$ reflex vertices (with an interior angle $\leq 240°$) we place one additional relay. Hence, for the total number of relays added at vertices, R_{V}, we have:

$$\begin{aligned} R_{\text{V}} &\leq (n - n_{\text{nr}} - n_{240})\cdot 1 + n_{240}\cdot 3 + n_{\text{nr}}\cdot 0 \\ &\leq (n-n_{\text{nr}})\cdot 1 + 6\cdot n_{\text{nr}} = n + 5\cdot n_{\text{nr}} \end{aligned}$$

We distinguish two cases as follows.

1. $n_{\text{nr}} \leq \frac{n}{4} \Rightarrow R_{\text{V}} \leq n + \frac{5}{4}n = \frac{9}{4}n$
2. $n_{\text{nr}} > \frac{n}{4}$. In this case, we need at most 3 additional relays for at most $\frac{3}{4}$ vertices: $n - n_{\text{nr}} \leq \frac{3}{4}n \Rightarrow R_{\text{V}} \leq 3\cdot\frac{3}{4}n = \frac{9}{4}n$

Altogether, we have:

$$R_{\text{ALG}} \leq k + 2D + \frac{9}{4}n \leq \frac{21}{4}R_{\text{OPT}} \qquad (5)$$

Theorem 3 *A simple polygon allows a $\frac{21}{4}$-competitive strategy for the online minimum relay triangulation problem.*

5 Conclusion

We introduced the online minimum relay triangulation problem. We gave a lower bound of $\frac{9}{8}$ for the competitive ratio for any online algorithm (even in simple polygons). For polygons we presented a 6-competitive algorithm, and showed that it is $\frac{21}{4}$-competitive for simple polygons. Considering the gap between the lower bound and the given competitive ratio an open question is whether we are able to improve the ratio.

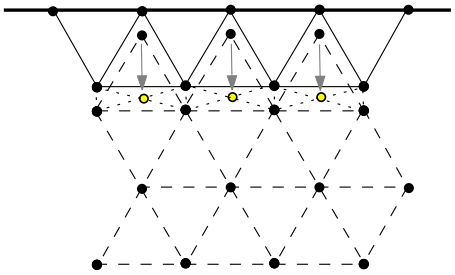


Figure 9: An example for the combination of the triangulation along P's boundary and the triangulation of the interior into one triangulation.

Another closely related problem arises from considering a limited number of robots; the *Maximum Coverage Triangulation Problem* (OMCTP) asks for a triangulation using ℓ robots, such that the area within an unknown polygon P that is covered is maximized.

Acknowledgements

This work was partially supported by the EU within the 7th Framework Programme under contract 215270 (FRONTS).

References

[1] V. Chvátal. A combinatorial theorem in plane geometry. *J. Combin. Theory Ser. B*, 18:39–41, 1975.

[2] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 2008.

[3] A. Efrat, S. P. Fekete, P. R. Gaddehosur, J. S. Mitchell, V. Polishchuk, and J. Suomela. Improved approximation algorithms for relay placement. In *Proc. 16th Annu. Europ. Sympos. Algor.*, pages 356–367. Springer, 2008.

[4] S. P. Fekete, J. S. B. Mitchell, and C. Schmidt. Minimum covering with travel cost. In *Proc. 20th Internat. Sympos. Algor. Comput.*, volume 5878 of *LNCS*, pages 393–402. Springer, 2009.

[5] S. P. Fekete and C. Schmidt. Polygon exploration with time-discrete vision. *Comput. Geom.*, 43(2):148–168, 2010.

[6] F. Hoffmann, C. Icking, R. Klein, and K. Kriegel. The polygon exploration problem. *SIAM J. Comput.*, 31:577–600, 2001.

[7] C. Icking, T. Kamphans, R. Klein, and E. Langetepe. Exploring simple grid polygons. In *11th Internat. Comput. Combin. Conf.*, volume 3595 of *LNCS*, pages 524–533. Springer, 2005.

[8] J. O'Rourke. *Art Gallery Theorems and Algorithms*. The Internat. Series of Monographs on Computer Science. Oxford University Press, 1987.

Coding and Counting Arrangements of Pseudolines

Stefan Felsner*

Institut für Mathematik
Technische Universität Berlin
Strasse des 17. Juni 136
D-10623 Berlin, Germany

Pavel Valtr†

Department of Applied Mathematics and
Institute for Theoretical Comp. Sci. (ITI),
Charles University,
Malostranské nám. 25, 118 00 Praha 1,
Czech Republic

The full paper is available at:
http://www.math.tu-berlin.de/~felsner/Paper/new-pla.pdf

Abstract. Arrangements of lines and pseudolines are important and appealing objects for research in discrete and computational geometry. We show that there are at most $2^{0.657\,n^2}$ simple arrangements of n pseudolines in the plane. This improves on previous work by Knuth who proved an upper bound of $3^{\binom{n}{2}} \cong 2^{0.792\,n^2}$ in 1992 and the first author who obtained $2^{0.697\,n^2}$ in 1997. The argument uses surprisingly little geometry. The main ingredient is a lemma that was already central to the argument given by Knuth.

1 Introduction

Arrangements of pseudolines are the topic of a chapter in the Handbook on Discrete and Computational Geometry [7]. The monograph [3] is another general reference.

It is convenient to think of a *pseudoline* as an unbounded x-monotone curve in the Euclidean plane. An *arrangement of pseudolines* is a family of pseudolines with the property that each pair of pseudolines has a unique point of intersection where the two pseudolines cross. An arrangement is *simple* if no three pseudolines have a common point of intersection. Due to the x-monotonicity of the pseudolines there is a unique unbounded cell above all pseudolines of an arrangement, the *north-cell*. Arrangements with a distinguished unbounded cell, e.g., a north-cell, are called *marked arrangements*.

Two arrangements are *isomorphic*, i.e., considered the same, if they can be mapped onto each other by a homeomorphism of the plane. In the case of marked arrangements it is required that an isomorphism respects the distinguished cell and preserves orientations. Note that every marked arrangement is isomorphic to an arrangement of x-monotone pseudolines with the north-cell as distinguished cell.

*Partially supported by DFG grant FE-340/7-1
†Work by P.V. was supported by the projects 1M0545 and MSM0021620838 of the Ministry of Education of the Czech Republic.

In this paper we are interested in the number B_n of marked simple arrangements of n pseudolines. It is known that $B_n \in 2^{\Theta(n^2)}$. Our interest is in the multiplicative constant hidden in the $\Theta(n^2)$. Knuth [8] considers the counting problem for several related classes of arrangements, e.g. arrangements without marking or projective arrangements, their numbers only differ by polynomial factors.

We are going to study the growth of $b_n = \log_2(B_n)$. An easy lower bound construction is given in [10, sec. 6.2]; it yields $b_n > \frac{1}{9}n^2$. Knuth [8, page 37] shows $b_n > \frac{1}{6}n^2 - O(n)$. In Section 3 we use enumeration results for rhombic tilings to prove $b_n > 0.188\,n^2$.

The upper bound $B_n \leq 3^{\binom{n}{2}}$, i.e., $b_n \leq 0.7924\,n^2$ was shown by Knuth [8, page 39]. At the end of this monograph Knuth [8, page 96] comments that an improved bound of $b_n \leq 0.7194\,n^2$ can be obtained from the the sharpest version of the zone theorem. Felsner [2] obtained the bound $b_n \leq 0.6974\,n^2$. In Section 2 we review the idea in Knuth's proof and add a new simple idea to get $b_n \leq 0.6609\,n^2$. In the full paper we refine the analysis and prove the bound $b_n \leq 0.6571\,n^2$ stated in Theorem 4.

There are several nice representations and encodings of simple arrangements of pseudolines. We close the introduction by explaining three of them. Given a marked arrangement $\mathcal{A}$ of n pseudolines we label the pseudolines with $1, \ldots, n$ such that they cross a vertical line west of all intersections in increasing order from bottom to top; see Figure 1 (left).

Local sequences. Associate with pseudoline i the permutation α_i of $\{1, .., n\} \setminus i$ reporting the order from left to right in which the other pseudolines cross line i. The family $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ is called the family of *local sequences* of the arrangement.

Wiring diagrams. Goodman [6] introduced a class of drawings of simple arrangements called *wiring diagrams* to get well-arranged pictures of arrangements. The idea is to specify a set of n horizontal lines (wires)

and confine the pseudolines to these wires except for positions where they cross another pseudoline and thereby change to an adjacent wire. Figure 1 shows an example.

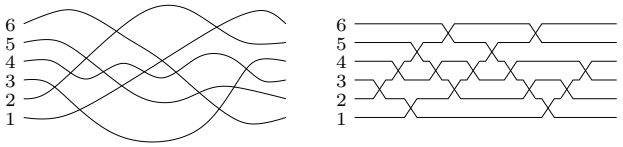


Figure 1: An arrangement $\mathcal{A}$ and a wiring diagram of $\mathcal{A}$.

Zonotopal tilings. A *zonotopal tiling* $\mathcal{T}$ is a tiling of a regular $2n$-gon with vertices $x_0, x_1, \ldots, x_{2n-1}$ in clockwise order starting with the highest vertex x_0. The tiles of $\mathcal{T}$ are rhombi $R(i,j)$, $1 \leq i < j \leq n$, such that $R(i,j)$ has one side which is a translate of the segment $[x_{i-1}, x_i]$ and one side which is a translate of the segment $[x_{j-1}, x_j]$. The tiles are not allowed to be rotated.

Zonotopal tilings can be viewed as normalized drawings of the duals of marked simple arrangements. Figure 2 shows an example. For additional information on zonotopal tilings and their relation to arrangements see [3] and [1].

Proofs of equivalence of the three representations are detailed in [3]. The basic tool for the proof of equivalence is to sweep the representations, resp. the arrangement, from left to right.

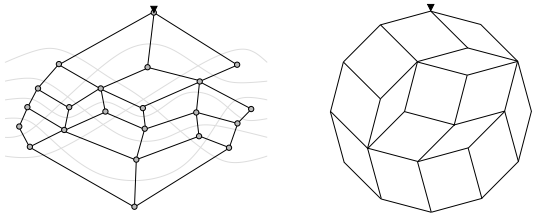


Figure 2: Arrangement $\mathcal{A}$ with its dual and the corresponding zonotopal tiling.

2 The upper bound

The idea in [2] was based on 'horizontal encodings' of arrangements. The first step was to replace the numbers in α_i by bits, a 1 for numbers j with $j < i$ and a 0 for $j > i$.

The proof of Knuth [8] takes a 'vertical' approach. Let $\mathcal{A}$ be an arrangement of $n+1$ pseudolines and consider pseudoline $n+1$ drawn into the wiring diagram of the arrangement $\mathcal{A}'$ induced by the first n pseudolines of $\mathcal{A}$. The *course* of pseudoline $n+1$ describes a *cutpath* descending from the north-cell to the south-cell of $\mathcal{A}$. Looking at the zonotopal tiling representation of $\mathcal{A}'$ as a graph a cutpath corresponds to a vertically decreasing path from the highest vertex x_0 to the lowest x_n. See [8] for details.

The number of arrangements $\mathcal{A}$ such that $\mathcal{A} \setminus n+1$ equals $\mathcal{A}'$ is exactly the number of different cutpaths of $\mathcal{A}'$. Define γ_n as the maximal number of cutpaths of an arrangement of n pseudolines can have. It then follows that

$$B_{n+1} \leq \gamma_n \cdot B_n. \tag{1}$$

Knuth proves that $\gamma_n \leq 3^n$, he also notes that the 'bubblesort arrangement' (see Figure 3) of size n has approximately $n\ 2^{n-2}$ cutpaths. Knuth also conjectures that the bubblesort arrangement is the maximizing example. The bubblesort arrangement is a particular Euclidean arrangement corresponding to the projective cyclic arrangement, cf. [11].

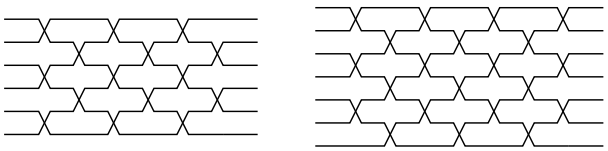


Figure 3: Wiring diagrams of the bubblesort arrangements of 6 and 7 lines.

In social choice theory a set T of permutations of $[n]$ is called an *acyclic set* if for all $i,j,k \in [n]$ at most two of ijk, jki, kij appear as a restriction of a permutation in T to $\{i,j,k\}$. The interest in acyclic sets comes from the fact that they avoid Condorcet cycles. That is, if voters are constrained to preference lists from an acyclic set T, then the majority digraph on the alternatives is acyclic. It has been shown in [5] that the set of cutpaths of an arrangement $\mathcal{A}$ is an acyclic set. Fishburn [4] introduced the *alternating scheme* as a large acyclic set. It turned out that the permutations in the alternating scheme correspond to the cutpaths of the bubblesort arrangement (Figure 3). Galambos and Reiner [5] gave a precise formula for the size of the alternating scheme and conjectures that this is the largest size of an acyclic set that can be obtained as the set of cuthpaths of an arrangement, i.e., in a different context they came up with the same conjecture as Knuth.

In the remainder of this section we present the main lemma of Knuth and show how to use it to bound the number γ_n of cutpaths of an arrangement.

Consider a cutpath p descending through the wiring diagram of $\mathcal{A}$. Having reached a cell c the path has to continue by crossing the wire w bounding c from below. The cells that can be reached from c by crossing w are ordered from left to right as $c_1, c_2, .., c_d$. Let their number d be the *degree* of c. When $d \geq 2$ we let c_1 be the *left successor* of c, and c_d be the *right successor* of c. The other cells $c_2, \ldots, c_{d-1}$ are called

middle successors of c. When $d = 1$ we let c_1 be the *unique successor* of c.

When the cutpath p of $\mathcal{A}$ traverses a cell c such that there is a middle successor cell c' of c separated from c by pseudoline j we say that p *sees a middle of color* j *at* c. If p descends from c to c' we say that p *has crossed pseudoline* j *as a middle.*

Lemma 1 (Knuth) *For every pseudoline* j *and every cutpath* p *it holds:* p *sees a middle of color* j *at most once.*

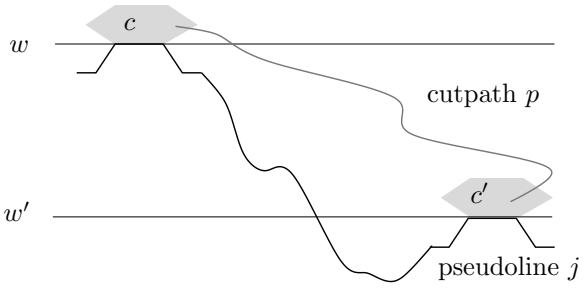


Figure 4: Illustrating the proof of Lemma 1

Proof. Suppose a cutpath p sees a middle of color j at different cells c and c'. Assuming that p visits c before c' we have a situation as sketched in Figure 4. Let t be the number of wires strictly between w and w'. Between the visits of the borders of c and c' pseudoline j has to change at least $(t+1)+2$ times from a wire to another, i.e., pseudoline j has at least $t+3$ crossings in this range. Every pair of pseudolines has only one crossing. Therefore, every pseudoline crossing j between c and c' also has to be traversed by the cutpath p on its way from c to c'. The cutpath p only intersects $t+1$ wires on its way from c to c', a contradiction. □

We use the lemma to encode cutpaths of an arrangement $\mathcal{A}$. With a cutpath p we associate two combinatorial objects:

- A set $M_p \subset [n]$ consisting of all j such that pseudoline j is crossed by p as a middle.
- A binary vector $\beta_p = (b_p(0), b_p(1), \ldots, b_p(n-1))$ such that $b_p(i) = 1$ only if leaving the cell between wire i and $i+1$ cutpath p proceeds to the left successor of c.

Claim I. The mapping $p \to (M_p, \beta_p)$ is injective from cutpaths of $\mathcal{A}$ to pairs consisting of a subset M of $[n]$ and a binary vector of length n.

From the claim it immediately follows that $\gamma_n \le 2^n\, 2^n = 4^n$. To improve the bound we use two simple observations:

- Every j taken as a middle has a corresponding entry in β_p that is not used.
- The lookups of entries of β_p are done in increasing order of indices.

It follows that we can take β_p to be a binary string of length $n - |M_p|$ and agree that lookups are always taken at the fist unused position of β. This improved encoding yields:

$$\gamma_n \;\le\; \sum_{k=0}^{n} \binom{n}{k} 2^{n-k} \;=\; 2^n\,(1+\frac{1}{2})^n \;=\; 3^n. \qquad (2)$$

This is the upper bound of Knuth, only the arithmetics in our derivation is simpler.

Note that our estimate for the length of β_p does not yet taking into account that some cells may have degree one. Define $\Gamma_{\mathcal{A}}(k,r)$ as the set of cutpaths in $\mathcal{A}$ that take k middles and visit r cells of degree one. From the above considerations we immediately have

$$|\Gamma(k,r)| \;\le\; \binom{n}{k} 2^{n-k-r} \qquad (3)$$

With the next lemma we show how to make use of this.

Lemma 2 $|\Gamma(k,r)| \;\le\; \min\left\{\binom{n}{k}, \binom{n}{r}\right\} 2^{n-k-r}.$

Proof. Paths in $\Gamma_{\mathcal{A}}(k,r)$ can also be encoded as cutpaths in the arrangement $\hat{\mathcal{A}}$ obtained from $\mathcal{A}$ via a 180° rotation of the plane. A cutpath p of $\mathcal{A}$ takes a middle to change from c to c' exactly if the rotated cutpath $\hat{p}$ of $\hat{\mathcal{A}}$ is reaching cell c as the unique successor of c'. In other words middles of p and uniques of $\hat{p}$ are in bijection as well as middles of $\hat{p}$ and uniques of p. This yields $\Gamma_{\mathcal{A}}(k,r) = \Gamma_{\hat{\mathcal{A}}}(r,k)$ and the lemma follows from formula (3). □

Using this lemma we get

$$\begin{aligned}
\gamma_n &\le \sum_{k,r} |\Gamma_{\mathcal{A}}(k,r)| && (4)\\
&\le \sum_{k,r} \min\left\{\binom{n}{k}, \binom{n}{r}\right\} 2^{n-k-r} \\
&\le 2\cdot 2^n \sum_{k=0}^{n} \binom{n}{k} 2^{-k} \sum_{r\ge k} 2^{-r} && (5)\\
&= 2^{n+1} \sum_{k=0}^{n} \binom{n}{k} 2^{-2k} \sum_{j\ge 0} 2^{-j} \\
&= 2^{n+2} \left(1+\frac{1}{4}\right)^n \;=\; 4\left(\frac{5}{2}\right)^n. && (6)
\end{aligned}$$

Combining this with (1) we get:

Theorem 3 *The number* B_n *of arrangements of* n *pseudolines is at most* $4^{n-1}\left(\frac{5}{2}\right)^{\binom{n-1}{2}}$, *hence for* n *large enough* $b_n \le 0.6609\, n^2$.

In the full paper we include a more careful analysis of the distribution of middles along cutpaths. This yields an improved bound on the size of $\Gamma(k,r)$ and the theorem:

Theorem 4 *If B_n is the number of arrangements of n pseudolines and $b_n = \log_2 B_n$, then $b_n \leq 0.6571\, n^2$ for large enough values of n.*

3 A lower bound

Given three numbers i, j and k we consider the set of $i+j+k$ pseudolines $1, 2, \ldots, i+j+k$ partitioned into the following three parts: $\{1, \ldots, i\}$, $\{i+1, .., i+j\}$, and $\{i+j+1, .., i+j+k\}$. A partial arrangement on this set is called *consistent* if any two pseudolines from different parts cross while any two pseudolines from the same part do not cross. The zonotopal duals of consistent partial arrangements are rhombic tilings of the centrally symmetric hexagon $H(i,j,k)$ with side lengths i, j and k; Figure 5 shows an example.

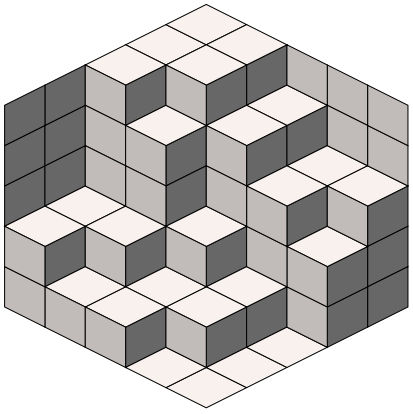


Figure 5: The hexagon $H(5,5,5)$ with one of its rhombic tilings.

The enumeration of rhombic tilings of $H(i,j,k)$ is a classical combinatorial problem solved by MacMahon [9]. There are

$$PP(i,j,k) = \prod_{a=0}^{i-1} \prod_{b=0}^{j-1} \prod_{c=0}^{k-1} \frac{a+b+c+2}{a+b+c+1} \qquad (7)$$

such tilings.

Consider a consistent partial arrangement with three parts of size n. Such an arrangement can be completed to a 'full' arrangement of $3n$ pseudolines by adding any arrangement of n lines for each of the three parts. This shows that

$$B_{3n} \geq PP(n,n,n)\, {B_n}^3. \qquad (8)$$

To find the growth rate of $PP(n,n,n)$ we first note that $PP(n,n,n) = T(n)/T(0)$ where $T(k) = \prod_{a=0}^{n-1} \prod_{b=0}^{n-1} (a+b+k+1)$. Let $t(k) = \ln T(k)$ and approximate $t(k)$ by an integral

$$\begin{aligned} t(k) &= \ln T(k) = \sum_{a=0}^{n-1} \sum_{b=0}^{n-1} \ln(a+b+k+1) \\ &\approx \int_{x=0}^{n} \int_{y=0}^{n} \ln(x+y+k+1)\, dy\, dx. \end{aligned} \qquad (9)$$

From this approximation it can be concluded that

$$\ln PP(n,n,n) = t(n) - t(0) \approx \left(\frac{9}{2}\ln(3) - 2\ln(2)\right) n^2. \qquad (10)$$

Combining this with Formula (8) we get:

Proposition 1 *The number B_n of arrangements of n pseudolines is at least $2^{0.1887\, n^2}$.*

References

[1] A. Björner, M. Las Vergnas, N. White, B. Sturmfels, and G. M. Ziegler, *Oriented Matroids*, Cambridge University Press, 1993.

[2] S. Felsner, *On the number of arrangements of pseudolines*, Discr. and Comput. Geom., 18 (1997), pp. 257–267.

[3] S. Felsner, *Geometric Graphs and Arrangements*, Advanced Lectures in Mathematics, Vieweg Verlag, 2004.

[4] P. Fishburn, *Acyclic sets of linear orders*, Soc. Choice Welfare, 14 (1997), pp. 113–124.

[5] Á. Galambos and V. Reiner, *Acyclic sets of linear orders via the Bruhat orders*, Soc. Choice Welfare, 30 (2008), pp. 245–264.

[6] J. E. Goodman, *Proof of a conjecture of Burr, Grünbaum and Sloane*, Discr. Math., 32 (1980), pp. 27–35.

[7] J. E. Goodman, *Pseudoline arrangements*, in Handbook of Discrete and Computational Geometry, Goodman and O'Rourke, eds., CRC Press, 1997, pp. 83–110.

[8] D. E. Knuth, *Axioms and Hulls*, vol. 606 of Lect. Notes Comput. Sci., Springer-Verlag, 1992.

[9] P. A. MacMahon, *Combinatory analysis. Vol. II.*, Chelsea, 1960. Reprint of the 1916 edition.

[10] J. Matoušek, *Lectures on Discrete Geometry*, vol. 212 of Graduate Texts in Mathematics, Springer-Verlag, 2002.

[11] G. M. Ziegler, *Higher Bruhat orders and cyclic hyperplane arrangements*, Topology, 32 (1993), pp. 259–279.

Even and quasi-even triangulations of point sets in the plane

I. Fernández*‡ C.I. Grima†‡ A. Márquez†‡ A. Nakamoto§ R. Robles†‡ J. Valenzuela†‡

Abstract

Given a point set in the plane, among all its triangulations, we call even triangulations to those with all the vertices with even degree, and quasi-even triangulations to those with all interior vertices with even degree. The former are the 3-colorable triangulations of a point set. We study some problems of those triangulations.

1 Introduction

A triangulation is said to be *even* or *eulerian* if each vertex has even degree (*e-triangulation* for short); e-triangulations have been studied since a seminal result by Whitney

Theorem 1 *([Whitney])*
A plane triangulation is 3-colorable if and only if all its vertices have even degree.

Hoffmann and Kriegel proved an important combinatorial theorem [4]: Every 2-connected bipartite plane multigraph G without 2-cycle faces has a triangulation in which all vertices have even degree. Combined with Theorem 1, this result implies that every such graph has a 3-colorable plane triangulation. Using this theorem, Hoffmann and Kriegel significantly improved the upper bounds of several art gallery and prison guard problems. In fact, e-triangulations are very much related with coloring problems [9], another interesting reference is [6].

Recently, Nakamoto and some collaborators [7, 10] have proven that, using sequences of some local transformation (flips), it is possible to transform any e-triangulation on certain surfaces into another.

In general, e-triangulations have been studied from a topological point of view, meaning that the position of the vertices in the plane or the realization of the edges as line segments has not been taken into account. In this paper, we try to study this problems for geometric triangulations of point sets in the plane. By geometric we understand that the vertices are actual points (with fixed coordinates) in the plane, and the edges are represented by non-crossing line segments.

Thus, given a point set S in the plane, we consider, among all its (geometric) triangulations, two special classes, the geometric even triangulations (e-triangulations hereafter), and the geometric quasi-even triangulations (*q-e-triangulations*) where all the vertices with odd degree lie on the outerface (or, equivalently, they are extreme points of the convex hull).

Although the study of geometric e-triangulations is natural by itself (see [5], for instance), it is easy to see that geometric q-e-triangulations are exactly those triangulations of a point set that can be 3-colorable.

Theorem 2 *A (geometric) triangulation T of a point set S is 3-colorable if and only if it is a q-e-triangulation.*

Proof. We can place the vertices of the convex hull of S on the equator of an sphere, and make two copies of the interior vertices of S, one in the northern hemisphere and the other in the southern hemisphere, to obtain a new set S' on the sphere. Obviously T induces in a natural way a triangulation T' of S' that is a triangulation of the sphere, and T' is even if and only if T is even. Now, applying Theorem 1 we obtain our result.

□

Thus, in some way, q-e-triangulations are a good generalization of triangulations of polygons (in some cases some results of triangulations of polygons–as the original Art Gallery Theorem– are obtained just using that they admit a 3-coloring).

2 e-triangulations of convex sets

In this section, we consider e-triangulations of point sets in convex position. We study three main problems: the existence of such a triangulation, the number of e-triangulations, and the connectivity of the graph of e-triangulations using some local transformations known as *N-flips*.

Throughout this section, S will be a convex set with $n \geq 3$ vertices. In general, we label these vertices in a clockwise direction starting by 1 (and ending in n).

*Research partially supported by MCYT-FEDER research project MTM2007-65249 and Junta de Andalucía Grant P06-FQM-01642

†Research partially supported by MCYT-FEDER research project MTM2008-05866-C03-01 and Junta de Andalucía Grant P06-FQM-01649

‡Dept. Matemática Aplicada I, Universidad de Sevilla, Seville, Spain ({isafer,grima,almar,rafarob,jesusv}@us.es)

§Dept. of Mathematics, Yokohama National University, Yokohama, Japan (nakamoto@edhs.ynu.ac.jp)

2.1 Existence of e-triangulations in convex sets

In this subsection we study the convex sets admitting an e-triangulation.

We call a *piece* P of a triangulation of S to three consecutive vertices u, a, v with $deg(u) = deg(v) = 2$ and $deg(a) = 4$, the vertex a is called the *apex* of P. The following result can be proved using induction and the dual graph:

Lemma 3 *Any e-triangulation T on S has at least one piece; moreover, if $|S| \geq 9$, then it has at least two pieces.*

As a consequence of this lemma, we can obtain several results. In particular, if a triangulation is colored as a chessboard as in [1] (every triangle is either black or white and edge-adjacent triangles have different colors), it is possible to give another characterization of e-triangulations.

Lemma 4 *Let T be a triangulation of a convex polygon P. The following conditions are equivalent*

1. *T is an e-triangulation*
2. *All the edges in P belong to triangles with the same color.*
3. *Every vertex in P belongs to an odd number of triangles.*

In fact, some of the results of the above lemma (but not all of them) are mentioned in [1], but it lacks in a proof and it is not mentioned explicitly that those conditions are equivalent.

Another result that it can be deduced from Lemma 3 using induction is

Lemma 5 *S admits an e-triangulation if and only if $|S| = 3k$.*

In fact Lemma 5 is saying us how to obtain a generic e-triangulation: starting in a triangle, we add a piece to one of the edges of the outerface, and so on. Of course, if we have a polygon with a number of vertices that is not a multiple of 3, we can think in adding Steiner points in its interior in order to get an e-triangulation, but this is hopeless again as a consequence of the proof of Lemma 3:

Proposition 6 *Let S be a point set that admits an e-triangulation, then $|CH(S)| = 3k$.*

2.2 Counting e-triangulations of convex sets

From now on T_{3k} will denote the number of e-triangulations of a convex set S with $|S| = 3k$ points.

Remark 1 *Observe that in an e-triangulation of a convex set, if we remove any edge uv, and we consider for u and v the degrees to the left and to the right of uv, then the four (in principle different) degrees so obtained have the same parity.*

Lemma 7 *The number f_{3k} of e-triangulations of a convex set S with $|S| = 3k$ points such that all of them contain the triangle $1, 2, 3k$, verifies*

$$f_{3k} = T_3 T_{3(k-1)} + T_{3\cdot 2} T_{3(k-2)} + \ldots + T_{3(k-1)} T_3$$

$T_3 = 1$, $T_6 = 2$.

Proof. (sketch) If we have an e-triangulation T of S with the triangle $1, 2, 3k$, then the edge $3k, 2$ is interior and must belong to another triangle $3k, 2, v$. First of all, observe that, by Remark 1, that T induces e-triangulations in the subsets $S_1 = \{2, 3, \ldots, v\}$ and $S_2 = \{v, v+1, \ldots, 3k\}$. Then the result is obtained from the following observations a) Since S_1 and S_2 admit e-triangulations, then $v = 3m+2$. b)The number of e-triangulations of S containing both triangles $1, 2, 3k$ and $3k, 2, v$ is the number of e-triangulation of S_1 (T_{3m}) times the number of e-triangulations of S_2 (T_{3k-3m}). □

Proposition 8 *The number T_{3k} can be computed by the following recursion*

$$T_{3k} = f_3 f_{3k} + f_{3\cdot 2} f_{3(k-1)} + \ldots + f_{3k} f_3$$

$$f_{3k} = T_3 T_{3(k-1)} + T_{3\cdot 2} T_{3(k-2)} + \ldots + T_{3(k-1)} T_3$$

with $T_3 = f_3 = f_6 = 1$.

From this proposition, we can compute the values of T_{3k}

k	1	2	3	4	5	6	7
f_{3k}	1	1	4	22	140	969	7084
T_{3k}	1	2	9	52	340	2394	17710

k	8	9	10
f_{3k}	53820	420732	3362260
T_{3k}	135720	1068012	8579560

In the same way, we can obtain another method to obtain an explicit formula given the number of e-triangulations (check [1]).

Indeed, if we set $a_n = f_{3(n+1)}$ and $b_n = T_{3(n+1)}$, where $a_0 = b_0 = 1$, then we have

$$\begin{aligned} a_n &= b_0 b_{n-1} + b_1 b_{n-2} + \ldots + b_{n-1} b_0 = \\ &= \textstyle\sum_{k=0}^{n-1} b_k \; b_{n-k-1} \end{aligned}$$

$$\begin{aligned} b_n &= a_0 a_n + a_1 a_{n-1} + \ldots + a_n a_0 = \\ &= \textstyle\sum_{k=0}^{n} a_k \; a_{n-k} \end{aligned}$$

Proposition 9 *The generating series* $A(x) = \sum_{n=0}^{+\infty} a_n\ x^n$ *and* $B(x) = \sum_{n=0}^{+\infty} b_n\ x^n$ *verifies the algebraic equations*

$$A(x) = 1 + x\ B(x)^2\ ,\ B(x) = A(x)^2$$

These algebraic generating series can be expressed as finite sums of a kind of hypergeometric series in several variables using well-known techniques to obtain (see [1])

$$A(x) = \sum_{n=0}^{+\infty} \binom{4n}{n} \frac{x^n}{3n+1}$$

$$B(x) = \sum_{n=0}^{+\infty} \binom{4n+2}{n} \frac{x^n}{2n+1}$$

This implies

Corollary 10 $T_{3k} = \binom{4k-2}{k-1}\frac{1}{2k-1}$ *and* $f_{3k} = \binom{4k-4}{k-1}\frac{1}{3k-2}$

On the other hand, it can be seen that the number $f_{3k} = \binom{4k-4}{k-1}\frac{1}{3k-2}$ is related with the number of dissections of a polygon, the number of rooted loopless n-edge maps in the plane (planar with a distinguished outside face) [8], the number of lattice paths from $(1,0)$ to $(3n+1,n)$ which, starting from $(1,0)$, only utilize the steps $+(1,0)$ and $+(0,1)$ and, additionally, the paths lie completely below the line $y = 1/3x$ (i.e. if (a,b) is in the path, then $b < a/3$) , enumerates quartic trees (rooted, ordered, incomplete) with n vertices (including the root), and is the Pfaff-Fuss-Catalan sequence C_n^m for $m = 4$ [3, 11]. See [2] for a general reference on these sequences.

2.3 The graph of e-triangulations of a convex polygon

In several papers [7, 10], Nakamoto and others have studied the graph of (topological) e-triangulations with respect to N-flips. Suppose we have a hexagonal region in an e-triangulation T $v_1, v_2, v_3, v_4, v_5, v_6$ with diagonals $\overline{v_1v_3}$, $\overline{v_3v_6}$, and $\overline{v_4v_6}$ and no inner vertices. The N-flip of the path v_1, v_3, v_6, v_4 is to replace the diagonals $\overline{v_1v_3}$, $\overline{v_3v_6}$, and $\overline{v_4v_6}$ by $\overline{v_1v_5}$, $\overline{v_2v_5}$, and $\overline{v_4v_2}$ in the hexagonal region (see Figure 1).

We define the graph of e-triangulations of a set S as the graph having as many vertices as different triangulations has S, and with an edge between two vertices if it is possible to transform each corresponding e-triangulation into the other using one N-flip.

Theorem 11 *The graph of e-triangulations of a set S in convex position is connected.*

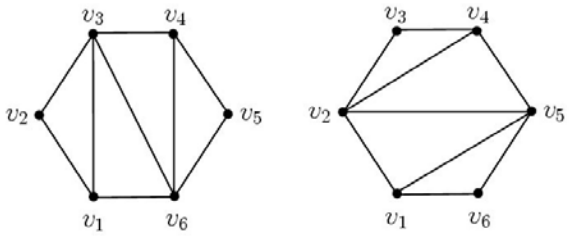


Figure 1: An N-flip.

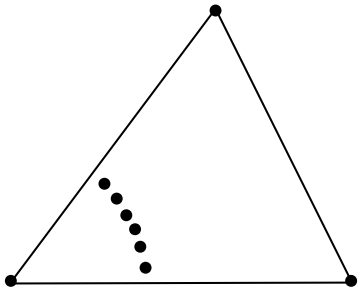


Figure 2: If the number of interior points in even, this configuration has not q-e-triangulation.

3 q-e triangulations

Of course, there exists configurations of points that does not admit q-e-triangulations, for instance four points in non-convex position. It can be thought that if there are many interior points, then we can always obtain a q-e-triangulation, but this is not the case.

Lemma 12 *There exist point sets (see Figure 2) arbitrarily large without q-e-triangulations.*

3.1 Extending a geometric graph to a q-e-triangulation

As it has been mentioned in the Introduction, those geometric graphs that can be extended to a q-e-triangulation can be used in some applications as extensions of the Art Gallery Theorem. For topological graphs Hoffmann and Kriegel [4] proved that every 2-connected bipartite plane multigraph G without 2-cycle faces can be extended to an even triangulation. But this result does not hold in the case of geometric graphs, as shown in Figure 3. The bipartite geometric graph depicted in that figure has only two extensions to triangulations and none of them is a q-e-triangulation. Even more, we can prove, using a reduction from PLANAR-3-SAT, the following result

Theorem 13 *Given a geometric graph G with n vertices, to decide if G can be extended to a q-e-triangulation is a NP-complete problem.*

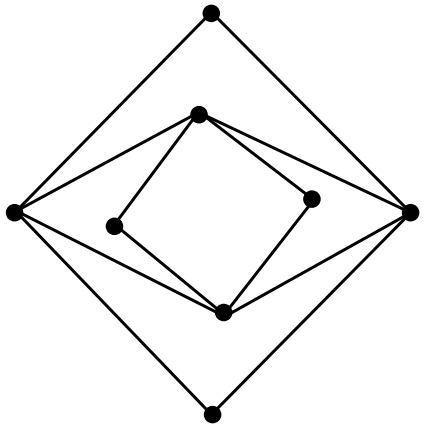


Figure 3: This geometric graph cannot be extended to a q-e-triangulation.

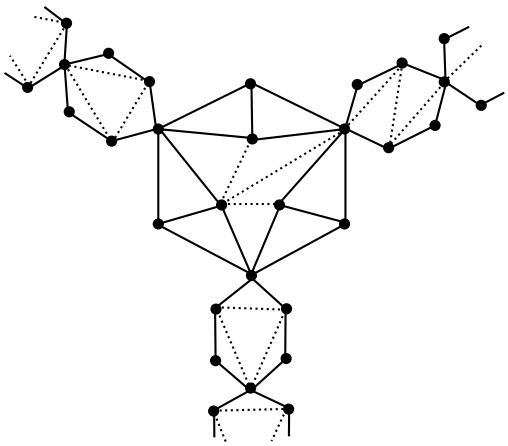


Figure 4: Clauses gadget (only solid lines).

Proof. (sketch) We make a reduction from PLANAR-3-SAT, using the following gadgets: One gadget for the clauses (see Figure 4) to three specific vertices we add pipelines (basically hexagons joined by opposite vertices) until we reach the variable gadget (see Figure 5). Completing the outerface in a proper way, we can check that this graph can be extended to a q-e-triangulations if and only if the original PLANAR-3-SAT entry is satisfacible.

□

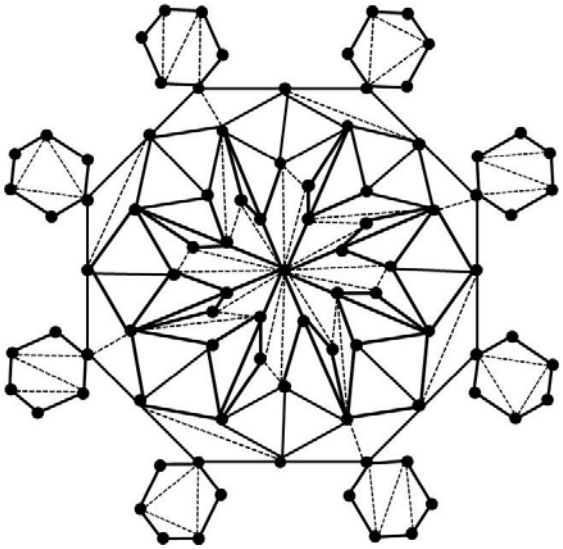


Figure 5: Variable gadget (only solid lines).

References

[1] R. Bacher. Fair triangulations. eprint arXiv:0710.0960v1. http://arxiv.org/abs/0710.0960.

[2] J. Cooper. On the on-line encyclopedia of integer sequences. http://www.research.att.com/~njas/ sequences/A002293.

[3] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1994.

[4] F. Hoffmann and K. Kriegel. A graph-coloring result and its consequences for polygon-guarding problems. *SIAM J. Discret. Math.*, 9(2):210–224, 1996.

[5] A. Hoffmann-Ostenhof. Mosaics, even triangulations and quadrangulations of the sphere. In *22nd British Combinatorial Conference*, 2009.

[6] J. Hutchinson, R. B. Richter, and P. Seymour. Colouring eulerian triangulations. *J. Comb. Theory Ser. B*, 84(2):225–239, 2002.

[7] K. Kawarabayashi, A. Nakamoto, and Y. Suzuki. N-flips in even triangulations on surfaces. *J. Comb. Theory Ser. B*, 99(1):229–246, 2009.

[8] V. A. Liskovets and T. R. Walsh. Enumeration of unrooted maps on the plane. Technical report, UQAM, Canada, 2005.

[9] A. Nakamoto. 5-chromatic even triangulations on surfaces. *Discrete Mathematics*, 308(12):2571–2580, 2008.

[10] A. Nakamoto, T. Sakuma, and Y. Suzuki. N-flips in even triangulations on the sphere. *J. Graph Theory*, 51(3):260–268, 2006.

[11] G. Pólya and G. Szegö. *Problems and Theorems in Analysis*, volume 1. Springer-Verlag, Heidelberg, New York, 1972.

Combinatorial Proof for fast Pivoting in K-matrix Linear Complementarity

Jan Foniok*† Komei Fukuda*‡ Lorenz Klaus*§

January 8, 2010

Abstract

We use a purely combinatorial argument to show that simple principal pivot methods applied to linear complementarity problems with K-matrices converge very quickly. The proof is conducted in the setting of oriented matroids.

1 Introduction

We are concerned with the *linear complementarity problem (LCP)*, which is for a given matrix $M \in \mathbb{R}^{n\times n}$ and a given vector $q \in \mathbb{R}^n$ to find two vectors w and z in $\mathbb{R}^n$ so that

$$\begin{aligned} w - Mz &= q, \\ w, z &\geq 0, \\ w^{\mathrm{T}} z &= 0. \end{aligned} \tag{1}$$

There is a broad range of applications. Many optimization problems such as *linear programming (LP)*, convex *quadratic programming (QP)* and finding a Nash Equilibrium of *bimatrix games* can be expressed as LCPs.

It is NP-complete to decide whether an LCP has a solution [2]. One in practice and theory likewise important class of LCPs are the instances where the matrix M is a *P-matrix*, i.e., a matrix with positive principal minors. We denote such an LCP as a P-LCP. It is well-known that every P-LCP(M, q) has a unique solution for every right-hand side q [6]. Further, if P-LCP were NP-hard, then NP = coNP [5]. Nevertheless, up to now no polynomial-time algorithm is known which finds the unique solution of every P-LCP.

In this work we study the behavior of *simple principal pivoting methods* in a combinatorial abstraction. These methods share their essential idea with the simplex algorithm. Given some LCP(M, q), we choose any set $B \subseteq [n]$ in the beginning. We then build a matrix A_B in $\mathbb{R}^{n\times n}$ as follows: the ith column is the ith column of $-M$ if $i \in B$ and the ith column of the n-dimensional identity matrix otherwise. If A_B is invertible, we call B a basis; if M is a P-matrix, every subset $B \subseteq [n]$ is a basis. Now, if $A_B^{-1} q \geq 0$, we have discovered the solution: let

$$(w_i, z_i) := \begin{cases} (0, (A_B^{-1}q)_i) & \text{if } i \in B, \\ ((A_B^{-1}q)_i, 0) & \text{if } i \notin B. \end{cases}$$

If on the other hand $A_B^{-1} q \not\geq 0$, then w and z defined above satisfy $w - Mz = q$ and $w^T z = 0$, but $w, z \geq 0$ fails. In simple principal pivoting, one tries to improve the situation by replacing the basis B with the symmetric difference $B \oplus \{i\}$, where i is selected by a *pivot rule* from the set of "bad indices" $\{i : (A_B^{-1}q)_i < 0\}$. Some pivot rules terminate on every P-LCP. Unfortunately, for many rules instances with non-polynomial runtime are known. Because of that we focus on a subclass of P-LCPs for which pivoting methods run provably fast. A *Z-matrix* is a matrix with non-positive off-diagonal elements. A *K-matrix* is a P-matrix which is also a Z-matrix. For K-LCPs, a recent result [4] shows that simple principal pivoting methods run in a number of pivot steps linear in n and thus in polynomial time. In the following we prove this result in a purely combinatorial way. Consider the set

$$\mathcal{V} = \Big\{ \operatorname{sgn} x : \begin{bmatrix} I_n & -M & -q \end{bmatrix} x = 0 \Big\},$$

where $\operatorname{sgn} x$ is a vector in $\{-, 0, +\}^{2n+1}$ defined as $(\operatorname{sgn} x)_i := \operatorname{sgn} x_i$. We claim that the collection $\mathcal{V}$ of sign vectors builds the set of vectors of an *oriented matroid* on $2n+1$ elements. We build up on Todd's [7] approach which consists in combinatorially generalizing LCPs by formulating the complementarity problem of oriented matroids (OMCP). First, we define P-matroids, Z-matroids and K-matroids, combinatorial analogues of the LCP classes mentioned above.

Starting with algebraic properties of K-matrices, given by Fiedler and Pták [3], we extract the combinatorics and derive properties of K-matroids, see Theorem 5. Finally, in Theorem 6 we show that simple principal pivot methods solve the OMCP on K-matroids in a linear number of pivot steps.

Observe that Theorem 6 implies the aforesaid result of [4]. Our proof rests upon purely combinatorial arguments, which is nice since we avoid the machinery of matrix algebra.

*ETH Zurich, Institute for Operations Research, 8092 Zurich, Switzerland
†foniok@math.ethz.ch
‡fukuda@math.ethz.ch
§lklaus@math.ethz.ch

2 Oriented matroids

The theory of oriented matroids provides a natural concept which generalizes combinatorial properties of many geometric configurations.

2.1 Definitions and basic properties

We restrict the presentation to the bare minimum. For more on oriented matroids consult, for instance, the book [1].

Let E be a finite set of size n. A *sign vector* on E is a vector X in $\{+,0,-\}^E$. We define $X^- = \{e \in E : X_e = -\}$, $X^\ominus = \{e \in E : X_e = - \text{ or } X_e = 0\}$, and the sets X^0, $X^\oplus$ and X^+ analogously. For any subset F of E we write $X_F \geq 0$ if $F \subseteq X^\oplus$, and $X_F \leq 0$ if $F \subseteq X^\ominus$; furthermore $X \geq 0$ if $X_E \geq 0$ and $X \leq 0$ if $X_E \leq 0$. The *support* of a sign vector X is $\underline{X} = X^+ \cup X^-$. The *opposite* $-X$ of X is the sign vector with $(-X)^+ = X^-$, $(-X)^- = X^+$ and $(-X)^0 = X^0$. The *composition* of two sign vectors X and Y is given by

$$(X \circ Y)_e = \begin{cases} X_e & \text{if } X_e \neq 0, \\ Y_e & \text{otherwise.} \end{cases}$$

The *product* $X \cdot Y$ is of two sign vectors is the sign vector given by

$$(X \cdot Y)_e = \begin{cases} 0 & \text{if } X_e = 0 \text{ or } Y_e = 0, \\ + & \text{if } X_e = Y_e \text{ and } X_e \neq 0, \\ - & \text{otherwise.} \end{cases}$$

Definition 1 An *oriented matroid* on E is a pair $\mathcal{M} = (E, \mathcal{C})$ where $\mathcal{C}$ is a set of sign vectors satisfying the following axioms:

(C1) $0 \notin \mathcal{C}$.
(C2) If $C \in \mathcal{C}$, then $-C \in \mathcal{C}$.
(C3) For all $C, D \in \mathcal{C}$, if $\underline{C} \subseteq \underline{D}$, then $C = D$ or $C = -D$.
(C4) If $C, D \in \mathcal{C}$, $e \in C^+ \cap D^-$ and $f \in (C^+ \backslash D^-) \cup (C^- \backslash D^+)$, then there is a $Z \in \mathcal{C}$ with $Z^+ \subseteq (C^+ \cup D^+) \backslash \{e\}$, $Z^- \subseteq (C^- \cup D^-) \backslash \{e\}$ and $Z_f \neq 0$.

The collection $\mathcal{C}$ is the set of *circuits* of the matroid and the property (C4) is called *strong circuit elimination*. An oriented matroid is equivalently described by specifying the set $\mathcal{V}$ of *vectors*; a vector is obtained by taking all finite compositions of circuits, i.e., $\mathcal{V} = \{C_1 \circ \cdots \circ C_k : k \geq 0,\ C_1, \ldots, C_k \in \mathcal{C}\}$. On the other hand a non-zero vector $C \in \mathcal{V}$ is a circuit if and only if there is no non-zero vector $X \in \mathcal{V}$ satisfying $\underline{X} \subset \underline{C}$.

A *basis* of an oriented matroid $\mathcal{M}$ is an inclusion-maximal set $B \subseteq E$ for which there is no circuit C with $\underline{C} \subseteq B$. Every basis B has the same size, called the *rank* of $\mathcal{M}$.

Proposition 1 *Let B be a basis of an oriented matroid $\mathcal{M}$. For every e in $E \backslash B$ there is a unique circuit $C(B,e)$, called* fundamental circuit, *for which $\underline{C(B,e)} \subseteq B \cup \{e\}$ and $C(B,e)_e = +$.*

Two sign vectors X and Y are *orthogonal* if the set $\{X_e \cdot Y_e : e \in E\}$ either equals $\{0\}$ or contains both $+$ and $-$. We then write $X \perp Y$.

Proposition 2 *For every oriented matroid $\mathcal{M} = (E, \mathcal{C})$ of rank n there is a unique oriented matroid $\mathcal{M}^* = (E, \mathcal{C}^*)$ of rank $|E| - n$ given by*

$$\mathcal{C}^* = \left\{ Y \subseteq \{-,0,+\}^E : X \perp Y \text{ for every } X \in \mathcal{C} \right\}.$$

This $\mathcal{M}^*$ is called the *dual* of $\mathcal{M}$. Note that $(\mathcal{M}^*)^* = \mathcal{M}$. The circuits of $\mathcal{M}^*$ are the *cocircuits* of $\mathcal{M}$ and the vectors of $\mathcal{M}^*$ are the *covectors* of $\mathcal{M}$.

For any $F \subseteq E$, the vector $X \backslash F$ denotes the subvector $(X_e : e \in E \backslash F)$ of X. Then let

$$\mathcal{C} \backslash F := \{X \backslash F : X \in \mathcal{C} \text{ and } X_f = 0 \text{ for all } f \in F\}.$$

It is easy to see that the pair $\mathcal{M} \backslash F = (E \backslash F, \mathcal{C} \backslash F)$ is an oriented matroid; it is said to be obtained from $\mathcal{M}$ by *deletion* of F.

Definition 2 A matroid $\hat{\mathcal{M}} = (E \cup \{q\}, \hat{\mathcal{C}})$ with $q \notin E$ is a *one-point extension* of $\mathcal{M}$ if $\hat{\mathcal{M}} \backslash \{q\} = \mathcal{M}$.

2.2 Complementarity in oriented matroids

From now on we are considering oriented matroids endowed with a special structure. The set of elements E_{2n} is a $2n$-element set with a fixed partition $E_{2n} = S \cup T$ into two n-element sets and a mapping $e \mapsto \overline{e}$ from E_{2n} to E_{2n} which is an involution, that is, $\overline{\overline{e}} = e$ for every $e \in E_{2n}$, and for every $e \in S$ we have $\overline{e} \in T$. The element $\overline{e}$ is called the *complement* of e. For a subset F of E_{2n} let $\overline{F} = \{\overline{e} : e \in F\}$. A subset F of E_{2n} is called *complementary* if $F \cap \overline{F} = \emptyset$.

The matroids we are working with are of the kind $\mathcal{M} = (E_{2n}, \mathcal{V})$, where the set $S \subset E_{2n}$ is a basis of $\mathcal{M}$. In addition, we study their one-point extensions $\hat{\mathcal{M}} = (\hat{E}_{2n}, \hat{\mathcal{V}})$, where $\hat{E}_{2n} = E_{2n} \cup \{q\}$ for some element $q \notin E_{2n}$. We consider only extensions $\hat{\mathcal{M}}$ that have the same rank as $\mathcal{M}$.

Definition 3 For an oriented matroid $\hat{\mathcal{M}}$ the *oriented matroid complementarity problem (OMCP)* is to find a vector X so that

$$X \in \hat{\mathcal{V}}, \tag{2a}$$
$$X \geq 0,\ X_q = +, \tag{2b}$$
$$X_e \cdot X_{\overline{e}} = 0 \quad \text{for every } e \in E_{2n}, \tag{2c}$$

or to report that no such vector exists. A vector X which satisfies (2b) is called *feasible*, one which satisfies (2c) is *complementary*. If an $X \in \hat{\mathcal{V}}$ satisfies (2b) and (2c), then X is a *solution* to the OMCP($\hat{\mathcal{M}}$).

Now we show how LCPs are related to OMCPs. Finding a solution to the LCP (1) is equivalent to finding an element x of

$$V = \Big\{ x \in \mathbb{R}^{2n+1} : \begin{bmatrix} I_n & -M & -q \end{bmatrix} x = 0 \Big\}$$

such that

$$\begin{aligned} x \geq 0,\ x_{2n+1} = 1, \\ x_i \cdot x_{i+n} = 0 \quad \text{for every } i \in [n]\,. \end{aligned} \tag{3}$$

We set $\hat{\mathcal{V}} = \{\operatorname{sgn} x : x \in V\}$ and consider the OMCP for the matroid $\hat{\mathcal{M}} = (\hat{E}_{2n}, \hat{\mathcal{V}})$. Clearly, if the OMCP has no solution, then V contains no vector x satisfying (3). If on the other hand there is a solution X satisfying (2a)–(2c), then the solution to the LCP can be obtained by solving the system of linear equations $\begin{bmatrix} I_n & -M & -q \end{bmatrix} x = 0$, $x_i = 0$ whenever $X_i = 0$ and $x_{2n+1} = 1$. This correspondence motivates the following definition.

Definition 4 An oriented matroid $\mathcal{M} = (E_{2n}, \mathcal{V})$ is *LCP-realizable* if there is a matrix $M \in \mathbb{R}^{n\times n}$ such that

$$\mathcal{V} = \Big\{ \operatorname{sgn} x : x \in \mathbb{R}^{2n} \text{ and } \begin{bmatrix} I_n & -M \end{bmatrix} x = 0 \Big\}.$$

LCP-realizability of extensions $\hat{\mathcal{M}} = (\hat{E}_{2n}, \hat{\mathcal{V}})$ is defined analogously.

3 P-, Z- and K-matroids

In this section, we investigate what properties of oriented matroids characterize oriented matroids realizable by special classes of matrices.

A matrix $M \in \mathbb{R}^{n\times n}$ is a *P-matrix* if its principal minors are positive, or in other words, if there is no non-zero vector x such that $x_i(Mx)_i \leq 0$ for every $i \in [n]$. In the setting of matroids this translates to:

Definition 5 A vector $X \in \{-,0,+\}^{E_{2n}}$ is *sign-reversing* if $X_e \cdot X_{\overline{e}} \leq 0$ for every $e \in S$. An oriented matroid $\mathcal{M}$ on E_{2n} is a *P-matroid* if it has no sign-reversing circuit.

P-matroids were extensively studied by Todd [7]. Among many characterizations of a P-matroid, it was shown that every rank-preserving one-point extension $\hat{\mathcal{M}}$ of $\mathcal{M}$ contains exactly one positive complementary circuit C with $C_q = +$ if and only if $\mathcal{M}$ is a P-matroid. In other words the OMCP on a P-matroid has exactly one solution. The next statement will become important later on.

Lemma 3 (Todd [7]) *For a P-matroid $\mathcal{M}$ every complementary subset $B \subset E_{2n}$ of cardinality n is a basis.*

The second class of matrices we examine is Z-matrices; the corresponding matroid generalizations are Z-matroids.

Definition 6 A matroid $\mathcal{M}$ on E_{2n} is a *Z-matroid* if for every circuit C of $\mathcal{M}$ we have: If $C_T \geq 0$, then whenever $e \in S$ and $C_e = +$, then also $C_{\overline{e}} = +$.

The definition of a K-matroid is straightforward.

Definition 7 A matroid $\mathcal{M}$ on E_{2n} is a *K-matroid* if it is a P-matroid and a Z-matroid.

In 1962, Fiedler and Pták listed thirteen equivalent conditions for a Z-matrix to be a K-matrix. Some of them concern the sign structure of vectors:

Theorem 4 (Fiedler–Pták [3]) *Let M be a Z-matrix. Then the following conditions are equivalent:*
(a) *$\exists x \geq 0$ such that $Mx > 0$;*
(b) *$\exists x > 0$ such that $Mx > 0$;*
(c) *the inverse M^{-1} exists and $M^{-1} \geq 0$;*
(d) *$\forall x \neq 0$ there is k such that $x_k(Mx)_k > 0$;*
(e) *M is a P-matrix.*

Our combinatorial generalization of the Fiedler–Pták Theorem 4 is the following.

Theorem 5 *For a Z-matroid $\mathcal{M}$ (with vectors $\mathcal{V}$, covectors $\mathcal{V}^*$, circuits $\mathcal{C}$ and cocircuits $\mathcal{D}$), the following statements are equivalent:*
(a) *$\exists X \in \mathcal{V} : X_T \geq 0$ and $X_S > 0$;*
(b) *$\exists X \in \mathcal{V} : X > 0$;*
(c) *$\forall C \in \mathcal{C}$: $C_S \geq 0 \implies C_T \geq 0$;*
(d) *$\nexists$ a s.r. circuit $C \in \mathcal{C}$, i.e., M is a P-matroid;*
(a*) *$\exists Y \in \mathcal{V}^* : Y_S \leq 0$ and $Y_T > 0$;*
(b*) *$\exists Y \in \mathcal{V}^* : Y_S < 0$ and $Y_T > 0$;*
(c*) *$\forall D \in \mathcal{D}$: $D_T \geq 0 \implies D_S \leq 0$;*
(d*) *$\nexists$ a s.p. cocircuit $D \in \mathcal{D}$.*

Proof sketch. The entries (a*)–(d*) are properties of the dual matroid $\mathcal{M}^*$. We mentioned them because matroid duality plays a crucial role in our proof. In order to use duality, let us first define the *reflection* of a matroid $\mathcal{M} = (E_{2n}, \mathcal{V})$ to be the matroid $\Re(\mathcal{M}) = (E_{2n}, \Re(\mathcal{V}))$, where $\Re(\mathcal{V}) = \{\Re(X) : X \in \mathcal{V}\}$ with

$$\big(\Re(X)\big)_e = \begin{cases} X_{\overline{e}} & \text{if } e \in S, \\ -X_{\overline{e}} & \text{if } e \in T. \end{cases}$$

Observe that $\Re\big(\Re(\mathcal{M})\big) = \mathcal{M}$ because of (C2), and that $\Re(\mathcal{M}^*) = \Re(\mathcal{M})^*$; thus

$$\Re\big(\Re(\mathcal{M}^*)^*\big) = \mathcal{M}. \tag{4}$$

The implications (a) ⇒ (b), (b) ⇒ (c), (b) ⇒ (a) and (d) ⇒ (a*) follow from matroid axioms. Then duality comes into play. Notice that a matroid $\mathcal{M}$ satisfies (a*) if and only if the reflection of its dual $\Re(\mathcal{M}^*)$

satisfies (a); analogously for (b*) and (b), (c*) and (c), and (d*) and (d). Thus if $\mathcal{M}$ satisfies (a*), then $\Re(\mathcal{M}^*)$ satisfies (a), hence also (b), and so (using (4)) $\mathcal{M}$ satisfies (b*). The missing implications (b*) $\Rightarrow$ (c*), (c*) $\Rightarrow$ (d*), and (d*) $\Rightarrow$ (a) are proved analogously. □

4 OMCP on K-matroids

Here we prove that the unique solution to every OMCP($\hat{\mathcal{M}}$) where the underlying matroid $\mathcal{M}$ is a K-matroid, is found by every simple principal pivot algorithm in a number of pivot steps linear in n.

In the following we assume that $\hat{\mathcal{M}}$ is a P-matroid extension. To study the algorithmic complexity of the OMCPs($\hat{\mathcal{M}}$), we must specify how the matroid extension is made available to the algorithm. For the present, we assume $\hat{\mathcal{M}}$ to be given by an oracle which, for a basis B of $\hat{\mathcal{M}}$ and a non-basic element $e \in \hat{E}_{2n} \backslash B$, outputs the unique fundamental circuit $C(B, e)$.

A simple principal pivot algorithm takes any complementary basis B^0, for instance $B^0 := S$, and proceeds as follows:

Algorithm 1 SimplePrincipalPivot($\hat{\mathcal{M}}, B^0$)

```
i := 0
C^0 := C(B^0, q)
while (C^i)^- ≠ ∅ do
  choose e^i ∈ (C^i)^- according to a pivot rule R
  B^{i+1} := B^i \ {e^i} ∪ {ē^i}
  C^{i+1} := C(B^{i+1}, q)
  i := i + 1
end while
return C^i
```

Since B^i is complementary, B^{i+1} is also complementary and Lemma 3 asserts that it is indeed a basis. The exchange of an element e^i for its complement $\overline{e^i}$ in the actual basis is called a *pivot step*. If the number of pivot steps is polynomial in n, then the algorithm terminates in polynomial time, provided that the oracle operation is polynomial. In the LCP-realizable case the operation is polynomial; in fact, it consists in solving a system of n linear equations in $2n + 1$ variables.

The number of pivots generally depends on the applied pivot rule and on some P-matroid extensions some rules may even enter an infinite loop. If the input is a K-matroid extension, though, then the SimplePrincipalPivot method is fast. We prove that

Theorem 6 *The simple principal pivot algorithm runs in at most $2n$ pivot steps on every K-matroid extension no matter what pivot rule is applied.*

Proof sketch. First, one can show that the following two properties hold. The proof of the first one is trivial whereas the proof of the second property requires Theorem 5.

- If $\hat{\mathcal{M}}$ is a P-matroid extension, then $C^{i+1}_{\overline{e^i}} \geq 0$ for every $i \geq 0$.
- If $\hat{\mathcal{M}}$ is a K-matroid extension, then for every $f \in T$: If $C^h_f \geq 0$ for some $h \geq 0$, then $C^k_f \geq 0$ for every $k \geq h$.

We prove that, no matter which pivot rule R one applies, every element $e \in E_{2n}$ is chosen at most once as the pivot element. Consider any pivot step h in the SimplePrincipalPivot algorithm. First suppose that the pivot element e^h is in S. According to the first remark $C^{h+1}_{\overline{e^h}} \geq 0$. Moreover, by the second remark, for every $k \geq h$ we have $C^k_{\overline{e^h}} \geq 0$ and $C^k_{e^h} = 0$. In other words, the elements e^h and $\overline{e^h}$ cannot become pivot elements in later steps. Secondly, if the pivot e^h is in T, the argumentation from above fails. It may eventually happen for some k that $\overline{e^h}$ is chosen as pivot e^k. However if so, our first argument applies for pivot step k and neither $\overline{e^h}$ nor e^h can become pivot elements again. □

References

[1] Anders Björner, Michel Las Vergnas, Bernd Sturmfels, Neil White, and Günter M. Ziegler. *Oriented Matroids*, volume 46 of *Encyclopedia of mathematics and its applications*. Cambridge University Press, second edition, 1999.

[2] Sung-Jin Chung. NP-completeness of the linear complementarity problem. *J. Optim. Theory Appl.*, 60(3):393–399, 1989.

[3] Miroslav Fiedler and Vlastimil Pták. On matrices with non-positive off-diagonal elements and positive principal minors. *Czechoslovak Math. J.*, 12 (87):382–400, 1962.

[4] Jan Foniok, Komei Fukuda, Bernd Gärtner, and Hans-Jakob Lüthi. Pivoting in linear complementarity: Two polynomial-time cases. *Discrete Comput. Geom.*, 42(2):187–205, 2009.

[5] Nimrod Megiddo. A note on the complexity of P-matrix LCP and computing an equilibrium. RJ 6439, IBM Research, Almaden Research Center, 650 Harry Road, San Jose, California, 1988.

[6] Hans Samelson, Robert M. Thrall, and Oscar Wesler. A partition theorem for Euclidean n-space. *Proc. Amer. Math. Soc.*, 9:805–807, 1958.

[7] Michael J. Todd. Complementarity in oriented matroids. *SIAM J. Alg. Disc. Meth.*, 5:467–485, 1984.

Fitting Flats to Points with Outliers

Guilherme D. da Fonseca*

Abstract

Determining the best shape to fit a set of points is a fundamental problem in many areas of computer science. We present an algorithm to approximate the k-flat that best fits a set of n points with $n - m$ outliers. This problem generalizes the smallest m-enclosing ball, infinite cylinder, and slab. Our algorithm gives an arbitrary constant factor approximation in $O(n^{k+2}/m)$ time, regardless of the dimension of the point set. For many practical sets of inliers, the running time is reduced to $O(n^{k+2}/m^{k+1})$, which is linear when $m = \Omega(n)$.

1 Introduction

Determining the best shape to fit a set of points is a fundamental problem in statistics, machine learning, data mining, computer vision, clustering, and pattern recognition. The case of fitting a lower-dimensional space deserves special attention since it can be used to minimize the effects of the curse of dimensionality. A widely used measure of how well a shape S fits a set P of n points in d-dimensional space is $\max_{p\in P} \min_{s\in S} \|ps\|$, the maximum Euclidean distance between any point $p \in P$ and the shape S. Unfortunately, this measure is very sensitive to the presence of outliers.

A more robust measure in the presence of $n-m$ outliers and m inliers consists of minimizing the following cost function: given a parameter $m \leq n$, the cost is the m-th smallest distance between a point in P and the shape S. In this paper, we consider an approximation to the case when S is a k-dimensional flat, for a given value of $k \in \{0, \ldots, d-1\}$. We show that, for an arbitrary $\varepsilon > 0$, we can find in $O_\varepsilon(n^{k+2}/m)$ time[1], with constant probability, a k-dimensional flat S with cost at most $1+\varepsilon$ times the optimum. We refer to this problem as *flat fitting*. We assume that the dimensions k, d are constants, but $1/\varepsilon$ is an asymptotic quantity. It is noteworthy that the complexity depends only on the target dimension k, regardless of the dimension of the point set. Our algorithm is Monte Carlo, but can be made deterministic at the expense of an $O(m)$ factor in the running time.

*Universidadade Federal do Estado do Rio de Janeiro (UNIRIO), Brazil, fonseca@uniriotec.br

[1]We use the $O_\varepsilon(\cdot)$ notation to hide polynomial ε-dependencies.

In the most interesting case when m is a constant fraction of n, the running time of our Monte Carlo algorithm is $O_\varepsilon(n^{k+1})$. While the running time is close to the $\Omega(n^k)$ lower bound, the algorithm is still super-linear for $k \geq 1$. We show that when the set of inliers satisfies some density criterion, the running time is reduced to $O(n^{k+2}/m^{k+1})$, which is linear for $m = \Omega(n)$. This way, we show that despite the high worst-case complexity of the problem, there is a feasible solution for some practical large data sets.

Related work. The case of $k = 0$ corresponds to the well-studied problem of approximating the smallest ball enclosing m points. The problem can be solved in $O(n/\varepsilon^{d-1})$ expected time by using techniques from [2, 5, 9]. An easier variation of this problem, when an inlier is known, is used as a base case for our algorithm.

The case of $k = d - 1$ corresponds to approximating the narrowest slab enclosing m points. In contrast to the linear complexity of the $k = 0$ case, the most efficient solution for $k = d - 1$ is a high probability Monte Carlo algorithm [6] with running time $O(n^d(\log^{O(1)} \frac{1}{\varepsilon})/m\varepsilon)$. Major improvements are unlikely, since there is a lower bound of $\Omega((n-m)^{d-1} + (n/m)^d)$ for obtaining a constant approximation [6].

The case of $k = 1$ corresponds to approximating the smallest infinite cylinder enclosing m points, which is stated as an open problem by Har-Peled and Mazumdar [9]. A linear time solution for arbitrary values of m is unlikely, since even the planar approximation problem is 3SUM-hard [8]. To see that, note that it is 3SUM-hard to decide if there are three points on a line and that there is a planar cylinder of radius 0 enclosing three points if and only if there are three points on a line.

When the number $n - m$ of outliers is small compared to n, we can use the coreset framework to reduce the number of points to $O((n-m)/\varepsilon^{(d-1)/2})$ and then solve the problem in the reduced point set [1]. The case when d is an asymptotic variable is considered in [10], where an algorithm linear in d but exponential in $1/\varepsilon$ is presented. Approaches based on random sampling such as RANSAC [7] are widely used in practice, but do not guarantee approximation with respect to the optimum.

The non-robust version of the problem (when $m = n$) is generally solved using coresets [4]. The case when d is an asymptotic variable is considered in [11].

When $k = 0$, it is well known that the non-robust exact version can be solved in $O(n)$ time. Exact solutions for other values of k are considerably less efficient, even in the non-robust version. Chan [3] mentions an $O(n^{\lceil d/2 \rceil})$ algorithm for $k = d - 1$ and an $O(n^{2d-1+\delta})$ algorithm for $k = 1$, where δ is an arbitrarily small constant.

The exact robust version seems even harder. A trivial solution takes $O(n^{(d-k)(k+1)+2})$ time, by counting the number of points for each potential set of up to $(d-k)(k+1)+1$ farthest inliers. When $k = d-1$, the problem can be solved in $O(n^d)$ expected time [6], improving the trivial solution by two $O(n)$ factors, one by efficiently counting the number of points using arrangements, and one by using Chan's randomized optimization [2].

A lower bound of $\Omega((n-m)^{d-1} + (n/m)^d)$ for obtaining a constant approximation when $k = d - 1$ in presented in [6]. The lower bound is based on a conjecture for the complexity of the affine degeneracy problem. We can linearly reduce the flat fitting problem with $k = d-1$ to the flat fitting problem in higher dimension $d' \geq d$ and the same value of k. Therefore, the lower bound for $k = d - 1$ implies a lower bound of $\Omega((n-m)^k + (n/m)^{k+1})$ for arbitrary k. In the most interesting case when m is a constant fraction of n, the lower bound is $\Omega(n^k)$ and we present an upper bound of $O(n^{k+1})$.

Next, we present approximate algorithms for the flat fitting problem. We present a Monte Carlo algorithm with running time $O_\varepsilon(n^{k+2}/m)$ and a deterministic algorithm with running time $O_\varepsilon(n^{k+2})$. In Section 3, we show how to reduce the running time of the Monte Carlo algorithm to $O_\varepsilon(n^{k+2}/m^{k+1})$ for some typical sets of inliers. Concluding remarks and open problems are discussed in Section 4.

2 Approximate Algorithm

The general idea of the algorithm consists of finding a vector v that is approximately parallel to the best fitting flat and then projecting the points onto a hyperplane perpendicular to v and recursively solving a lower dimensional problem. We use $k = 0$ as a base case. Actually, the algorithm computes a somewhat small set of vectors that contains v and recurses for each vector in the set, returning the best solution found. We start by providing some definitions.

Let $S_{k,d}(P)$ and $c_{k,d}(P)$ respectively denote the optimal k-dimensional flat for point set P in d-dimensional space and its cost. We refer to the m points $P' \subseteq P$ within distance $c_{k,d}(P)$ of $S_{k,d}(P)$ as *inliers*. Given a d-dimensional set of points P and a vector v, let $P_{|v}$ denote a $(d-1)$-dimensional point set obtained by projecting P onto a hyperplane perpendicular to v. Given a vector v let v' be the unit length projection of v onto the optimal flat $S_{k,d}(P)$,

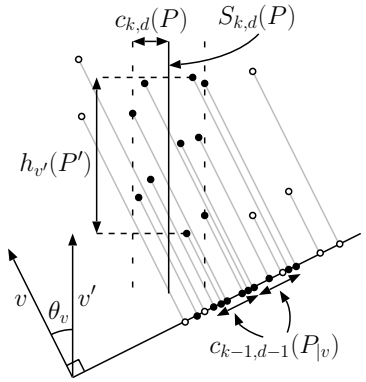


Figure 1: Definitions used to state Lemma 1. The $m = 10$ inliers are represented by solid circles.

$h_{v'}(P') = \max_{p \in P'} v' \cdot p - \min_{p \in P'} v' \cdot p$ be the directional width in direction v' of the inliers, and θ_v be the acute angle between v and v'. See Figure 1 for a diagram of the previous definitions. The following lemma follows from simple trigonometric arguments and shows how to use the solution of a lower dimensional problem in order to approximate the original problem.

Lemma 1 *For any vector v we have*

$$c_{k,d}(P) \leq c_{k-1,d-1}(P_{|v}) \leq c_{k,d}(P) + h_{v'}(P')\theta_v.$$

By Lemma 1, it is possible to obtain a $(1+\varepsilon)$-approximation by finding a vector v with angle

$$\theta_v \leq \frac{\varepsilon \, c_{k,d}(P)}{d \, h_{v'}(P')} = \phi$$

and recursively solving the lower dimensional problem. Our algorithm considers a set of vectors that contains a vector u with $\theta_u < \phi$, returning the solution of minimum cost found. The following lemma is the key to obtain such set.

Lemma 2 *For every inlier $p \in P'$, there is an inlier $q \in P'$ such that the vector $v = q - p$ has*

$$\theta_v \leq \frac{4c_{k,d}(P)}{h_{v'}(P')} \text{ and}$$

$$\frac{h_{v'}(P')}{2} \leq \|v\| \leq 2c_{k,d}(P) + h_{v'}(P').$$

Proof. (sketch) Consider the inlier $q \in P'$ that realizes the maximum directional distance $\max_{q \in P'} |v' \cdot p - v' \cdot q|$ and use simple geometric arguments (see Figure 2). □

Say we have a vector v satisfying the properties of Lemma 2. If $c_{k,d}(P) \geq h_{v'}(P')$, then we obtain a set of size $O(1/\varepsilon^{d-k})$ containing a vector u with $\theta_u \leq \phi$ in the following manner. The intersection

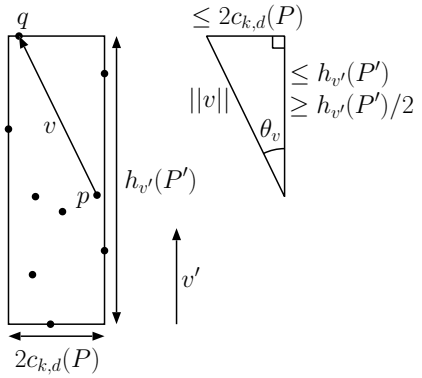


Figure 2: Proof of Lemma 2.

of a $(d-k+1)$-flat F in general position and the optimal flat $S_{k,d}(P)$ is a line ℓ. Using a standard grid of directions, we create a set of $O(1/\varepsilon^{d-k})$ vectors in F that contain a vector u within angle at most ε/d of ℓ, and consequently has $\theta_u \leq \phi$. Next, we focus on the more interesting case when $c_{k,d}(P) < h_{v'}(P')$.

By Lemma 2, we have that $\|v\|$ is a constant factor approximation of $h_{v'}(P')$. By Lemma 1, we can recursively solve the $(d-1)$-dimensional problem with point set $P_{|v}$ in order to obtain a constant factor approximation to $c_{k,d}(P)$. Putting both approximations together, we obtain a constant factor upper bound to θ_v. We use the approximation of θ_v to obtain a set of size $O(1/\varepsilon^{d-k})$ containing a vector u with $\theta_u \leq \phi$. The set is defined by a grid of directions in a $(d-k+1)$-flat as before, but noting that the angle between v and u is upper bounded by the approximation of θ_v.

Now, assume we know an inlier $p \in P'$. By Lemma 2, the set $V = \{p - q : q \in P\}$ of size $O(n)$ contains a vector satisfying the condition of Lemma 2. Therefore, we can obtain a set U of size $O(n/\varepsilon^{d-k})$ that contains a vector u with $\theta_u \leq \phi$.

For each vector $u \in U$, we project the points onto a hyperplane perpendicular to u and recursively solve the lower dimensional problem. Next, we discuss how to solve the base case $k = 0$, given an inlier p. The base case consists of approximating the smallest ball enclosing m points, including the inlier p.

Using techniques from [5, 9], the base case problem can be solved in time $O(n+m(\log\frac{1}{\varepsilon})/\varepsilon^{d-1})$. If we use Chan's randomized optimization [2], we obtain a Las Vegas algorithm with expected time $O(n+m/\varepsilon^{d-1})$. Actually, there is a very practical and straightforward solution with running time $O(n+m/\varepsilon^{d})$, which we present next for completeness. (i) Obtain a 2-approximation a of the radius by finding the m-th farthest point from p. (ii) Create a set Q containing the $\Theta(m)$ points within distance $2a$ of p. (iii) Consider a grid with cells of diameter εa. Compute the radius of the ball enclosing m points from Q centered at each of the $O(1/\varepsilon^{d})$ grid vertices within distance a from p, returning the smallest radius found.

Plugging the previous results together, the expected running time $t_{k,d}$ of the flat fitting algorithm, given an inlier is

$$t_{k,d} = \begin{cases} O(n/\varepsilon^{d-k})t_{k-1,d-1} & \text{if } k > 0 \\ O(n+m/\varepsilon^{d-1}) & \text{if } k = 0. \end{cases}$$

Consequently,

$$t_{k,d} = O\left(\frac{n^{k+1}}{\varepsilon^{k(d-k)}} + \frac{n^k m}{\varepsilon^{(k+1)(d-k)-1}}\right).$$

To get rid of the requirement of knowing an inlier, we apply the following random sampling technique used in [9]. Note that the set P contains m inliers. Therefore, a random element of P is an inlier with probability m/n and a random sample of n/m elements of P contains an inlier with probability at least $1-1/e$. Also, the set P of $O(n)$ elements is guaranteed to contain an inlier.

Theorem 3 *There is a Monte Carlo algorithm to compute, with constant probability, a $(1+\varepsilon)$-approximation of the k-flat that best fits m out of n points in d-dimensional space in time $O_\varepsilon(n^{k+2}/m)$ and, showing ε-dependencies,*

$$O\left(\frac{n^{k+2}}{m\varepsilon^{k(d-k)}} + \frac{n^{k+1}}{\varepsilon^{(k+1)(d-k)-1}}\right).$$

There is also also a deterministic algorithm with running time $O_\varepsilon(n^{k+2})$ and

$$O\left(\frac{n^{k+2}}{\varepsilon^{k(d-k)}} + \frac{n^{k+1}m\log(1/\varepsilon)}{\varepsilon^{(k+1)(d-k)-1}}\right).$$

3 Outer-dense Inliers

In this section, we show that for many data sets a random pair of inliers define a vector v satisfying the properties of Lemma 2 with constant probability. Consequently, we obtain a Monte Carlo algorithm with running time $O_\varepsilon(n^{k+2}/m^{k+1})$, which is linear for $m = \Omega(n)$.

We say that a halfspace H with normal vector v' is *deep* if $h_{v'}(P' \cap H) \geq h_{v'}(P')/4$. For a constant $\alpha \leq 1/2$, we say that the set P' is *α-outer-dense* if any deep halfspace H has $|P' \cap H| \geq \alpha|P'|$. The set P' is *outer-dense* if there is a constant α such that P' is α-outer-dense. The following lemma is analogous to Lemma 2 when the set P' is α-outer-dense.

Lemma 4 *If the inliers P' are α-outer-dense, then the vector $v = q - p$ defined by two random elements $p, q \in P'$ has*

$$\theta_v \leq \frac{4c_{k,d}(P)}{h_{v'}(P')} \text{ and}$$

$$\frac{h_{v'}(P')}{2} \leq \|v\| \leq 2c_{k,d}(P) + h_{v'}(P')$$

with probability at least $2\alpha^2$.

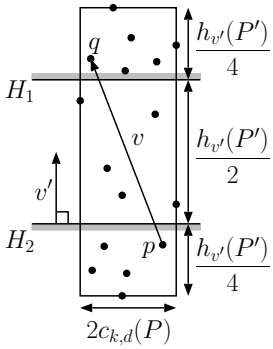


Figure 3: Proof of Lemma 4.

Proof. (sketch) Consider two disjoint deep halfspaces H_1, H_2 with normal vector v' such that v' is parallel to the optimal flat $S_{k,d}(P)$ and $h_{v'}(P' \cap H_1) = h_{v'}(P' \cap H_2) = h_{v'}(P')/4$ (see Figure 3). Since P' is outer-dense $|P' \cap H_1|, |P' \cap H_2| \geq \alpha |P'|$. Therefore, the probability that two random elements $p, q \in P'$ are one in H_1 and the other in H_2 is at least $2\alpha^2$. The lemma follows from the same trigonometric arguments as Lemma 2. □

Note that if a set of points is α-outer-dense, then the projection of the set onto a $(d-1)$-dimensional hyperplane is α-outer-dense in dimension $d-1$. Therefore, we obtain a Monte Carlo algorithm by sampling $n/m\alpha^2$ pairs of points at each step, and then solving the lower dimensional problems.

Theorem 5 *When the set of inliers is outer-dense, there is a Monte Carlo algorithm to compute, with constant probability, a $(1+\varepsilon)$-approximation of the k-flat that best fits m out of n points in d-dimensional space in time $O_\varepsilon(n^{k+2}/m^{k+1})$ and, showing ε-dependencies,*

$$O\left(\frac{n^{k+2}}{m^{k+1}\varepsilon^{k(d-k)}} + \frac{n^{k+1}}{m^{k+1}\varepsilon^{(k+1)(d-k)-1}}\right).$$

4 Conclusions and Open Problems

We address a generalization to several natural problems such as the smallest m-enclosing ball ($k=0$), infinite cylinder ($k=1$), and slab ($k=d-1$). Except for the two extreme cases, we present the first solution for the flat fitting problem. When m is a constant fraction of n, the gap between the lower bound and our Monte Carlo upper bound is only $\Theta(n)$.

We show that if the set of inliers is outer-dense, then the problem becomes exceedingly easier, with a linear time solution. Many practical sets of inliers are outer-dense. For example, point sets uniformly distributed in a convex region and on the boundary of a convex region are outer-dense with high probability.

A related decision problem which may be useful to reduce the running time of our Monte Carlo algorithm for general point sets by an $O_\varepsilon(n)$ factor is the following. Given a set P of n points in d-dimensional space and an integer $m \leq n$, determine if there is a line ℓ that passes through the origin and is within distance 1 from m points of P. The algorithm may give an approximate answer in the sense that points within distance between 1 and $1+\varepsilon$ may be counted either way. Except for the planar case, we know of no near linear solution, nor do we know if the problem is 3SUM-hard.

References

[1] P. K. Agarwal, S. Har-Peled, and H. Yu. Robust shape fitting via peeling and grating coresets. *Discrete Comput. Geom.*, 39(1):38–58, 2008.

[2] T. M. Chan. Geometric applications of a randomized optimization technique. *Discrete Comput. Geom.*, 22(4):547–567, 1999.

[3] T. M. Chan. Approximating the diameter, width, smallest enclosing cylinder, and minimum-width annulus. *Internat. J. Comput. Geom. Appl.*, 12(1/2):67–85, 2002.

[4] T. M. Chan. Faster core-set constructions and data-stream algorithms in fixed dimensions. *Comput. Geom.*, 35(1):20–35, 2006.

[5] C. M. H. de Figueiredo and G. D. da Fonseca. Enclosing weighted points with an almost-unit ball. *Inform. Process. Lett.*, 109:1216–1221, 2009.

[6] J. Erickson, S. Har-Peled, and D. M. Mount. On the least median square problem. *Discrete Comput. Geom.*, 36(4):593–607, 2006.

[7] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

[8] A. Gajentaan and M. H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Comput. Geom.*, 5(3):165–185, 1995.

[9] S. Har-Peled and S. Mazumdar. Fast algorithms for computing the smallest k-enclosing circle. *Algorithmica*, 41(3):147–157, 2005.

[10] S. Har-Peled and K. R. Varadarajan. Projective clustering in high dimensions using core-sets. In *Proc. 18th Annu. ACM Sympos. Comput. Geom.*, pages 312–318, 2002.

[11] S. Har-Peled and K. R. Varadarajan. High-dimensional shape fitting in linear time. *Discrete Comput. Geom.*, 32(2):269–288, 2004.

Hardness of discrepancy and related problems parameterized by the dimension

Panos Giannopoulos*† Christian Knauer*† Magnus Wahlström‡§ Daniel Werner*¶

Abstract

We study the complexity of the problem of star and box discrepancy, monochromatic and bichromatic empty box, and maximum empty subinterval, parameterized by the dimension d of the underlying space. We show that they are all W[1]-hard, and hence not solvable in $\mathcal{O}(f(d)n^c)$ time, for any computable function f and constant c (unless FPT=W[1]). We also establish NP-hardness of the box discrepancy problem and show that the largest empty subinterval problem is W[1]-hard even to approximate.

1 Introduction

We study the complexity of the following problems parameterized by the dimension d:
(i) Given two point sets R, B in $\mathbb{R}^d$, compute an axis-aligned box that does not contain any point of R and that contains as many points of B as possible (Bichromatic-Rectangle).
(ii) Given a set P in $[0,1]^d$, compute

- the largest empty axis-aligned box inside $[0,1]^d$ that contains the origin (Maximum-Empty-Subinterval).
- the largest empty axis-aligned box inside $[0,1]^d$ (Maximum-Empty-Box).
- the star discrepancy of P (Star-Discrepancy).
- the box discrepancy of P (Box-Discrepancy).

These problems (apart from Box-Discrepancy) are known to be NP-hard when the dimension d is part of the input and all known exact algorithms run in $n^{\mathcal{O}(d)}$ time, where n is the total number of objects in the input. Here, we ask whether any of these problems can be solved in $\mathcal{O}(f(d)n^c)$ time, for some computable function f and some constant c, i.e, whether they are *fixed-parameter tractable* with respect to d.

Results. We prove that all these problems are W[1]-hard with respect to d. Moreover, problem (i) is shown to be W[1]-hard when parameterized with both d and the size of the solution k, i.e., the number of points in B that the box contains. These results are obtained by fpt-reductions from the W[1]-complete [5] k-clique problem in general graphs, based on the general framework by Cabello et al. [3]. Moreover, since the fpt-reductions are actually polynomial (in both k and n), they also show the NP-hardness; for Box-Discrepancy this was not previously known. We also show that Maximum-Empty-Subinterval is W[1]-hard to approximate by a factor of $(1/2)^n$ (with respect to parameter d).

Related work. Bichromatic-Rectangle was shown to be NP-hard by Eckstein et al. [7]; in the same paper an $\mathcal{O}(n^{2d+1})$ algorithm was given. Aronov and Har-Peled [1] gave an $(1-\epsilon)$-approximation algorithm that runs in $\mathcal{O}(n^{\lceil d/2 \rceil}(\epsilon^{-2}\log n)^{\lceil d/2+1 \rceil})$ time.

Maximum-Empty-Box has only recently been shown to be NP-hard [2] when the dimension is part of the input, and the fastest exact algorithm runs in $\mathcal{O}(n^d \log^{d-2} n)$ [2]. Also recently, Dumitrescu and Jiang [6] gave an $\mathcal{O}((8ed\epsilon^{-2})^d \cdot n \log^d n)$-time $(1-\epsilon)$-approximation algorithm for this problem. The hardness of Maximum-Empty-Subinterval was shown by Gnewuch et al. [8].

When the dimension is part of the input, Star-Discrepancy has been shown to be NP-hard by Gnewuch et al. [8]. A non-trivial algorithm that still runs in $\mathcal{O}(n^{1+d/2})$ time was given in [4]. Thiémard [9] has given an approximation algorithm that achieves additive error and runs in fpt-time with respect to the error and the dimension.

2 The Bichromatic Rectangle Problem

We consider here Bichromatic Rectangle, for which the least technical details are required. We will then show how to adapt the construction to the other problems. The parameterized decision problem is defined as follows:

*Institut für Informatik, Freie Universität Berlin, Takustraße 9, D-14195 Berlin, Germany, {panos, knauer, dwerner}@inf.fu-berlin.de.

†This research was supported by the German Science Foundation (DFG) under grant Kn 591/3-1.

‡Max-Planck-Institut für Informatik, D-66123 Saarbrücken, Germany, wahl@mpi-inf.mpg.de.

§This research was supported by the German Research Foundation (DFG) via its priority programme "SPP 1307: Algorithm Engineering", grant DO 749/4-1.

¶This research was funded by Deutsche Forschungsgemeinschaft within the Research Training Group (Graduiertenkolleg) "Methods for Discrete Structures"

Definition 1 *k–d–*BICHROMATIC-RECTANGLE
Given: *A set R of red points and a set B of blue points in $\mathbb{R}^d$ and a $k \in \mathbb{N}$.*
Question: *Is there a hyperrectangle (box) $H = [l_1, r_1] \times \cdots \times [l_d, r_d]$ s.th. $H \cap R = \emptyset$ and $|H \cap B| \geq k$?*

A box that does not contain any point from R will be called *feasible*.

The idea. For a given simple graph $G = ([n], E)$ we will construct sets $R_{G,k}$ and $B_{G,k}$ in $\mathbb{R}^{2k}$ such that G has a clique of size k if and only if there is a feasible rectangle that contains $k+1$ blue points.

In addition to the (blue) origin 0, we will put points into k pairwise orthogonal planes. Each plane will contain n blue points, corresponding to the vertices of the graph, and $n+1$ red points which are placed such that it will not be possible for any feasible box to contain more than one blue point from a single plane. Then we forbid rectangles corresponding to vertices of G that are not connected to be chosen at the same time by putting red points into the pairwise product of the respective planes (which are four-dimensional subspaces).

For $1 \leq i \leq k$, we define

$$\mathbb{R}^2_i = \{(x_1, y_1, \ldots, x_k, y_k) \mid x_j = y_j = 0, j \neq i)\} \subseteq \mathbb{R}^{2k}.$$

For $1 \leq i < j \leq k$, we set $\mathbb{R}^4_{ij}$ to be the product of $\mathbb{R}^2_i$ and $\mathbb{R}^2_j$, i.e., $\mathbb{R}^4_{ij} = \mathbb{R}^2_i \times \mathbb{R}^2_j$. For $p \in \mathbb{R}^2_i$ and a point $q \in \mathbb{R}^2_j$, observe that the unique point in $\mathbb{R}^4_{ij}$ that (orthogonally) projects to p (into $\mathbb{R}^2_i$) and to q (into $\mathbb{R}^2_j$) is $p + q$.

Encoding vertices. Let $\varepsilon = 1/4$. For a vertex $1 \leq v \leq n$, we define the point $b_i(v) \in \mathbb{R}^2_i$ as $b_i(v) = (v + 1 - \varepsilon, n + 2 - v - \varepsilon)$. Then we set

$$B^{\text{scaffold}}_i = \{b_i(1), \ldots, b_i(n)\} \subseteq \mathbb{R}^2_i.$$

Choosing a (rectangle containing) point $b_i(v)$ will correspond to choosing vertex v from G. Let $B^{\text{scaffold}} = \uplus_{1 \leq i \leq k} B^{\text{scaffold}}_i$ be the set of all these blue points.

As we want the feasible boxes to contain at most one point from each $\mathbb{R}^2_i$, we add a set of red points as follows: For $1 \leq v \leq n+1$, we define $r_i(v) = (v, n + 2 - v)$ and set

$$R^{\text{scaffold}}_i = \{r_i(1), \ldots, r_i(n), r_i(n+1)\} \subseteq \mathbb{R}^2_i.$$

Then we set $R^{\text{scaffold}} = \uplus_{1 \leq i \leq k} R^{\text{scaffold}}_i$. See Figure 1 for an example of the scaffold construction.

It is easy to verify that any feasible box can contain at most one point from each B^{scaffold}_i.

Encoding edges. We will place several red points between all pairs of $\mathbb{R}^2_i$s that forbid certain blue points to be selected at the same time. For a vertex $1 \leq v \leq n$, we define the point $r^{\text{kill}}_i(v) \in \mathbb{R}^2_i$ as having coordinates $(v + 1 - 3\varepsilon, n + 2 - v - 3\varepsilon)$. These points are themselves *not* added to the set of red points. Observe that any rectangle that contains $b_i(v)$ also "contains" the point $r^{\text{kill}}_i(v)$. Then, for $1 \leq i \leq j \leq k$ and two vertices $1 \leq u, v \leq n$, we define the point

$$r^{\text{kill}}_{ij}(uv) = r^{\text{kill}}_i(u) + r^{\text{kill}}_j(v) \in \mathbb{R}^4_{ij}.$$

The set of all killing points in $\mathbb{R}^4_{ij}$ is then

$$R^{\text{E}}_{ij} = \{r^{\text{kill}}_{ij}(uv), r^{\text{kill}}_{ij}(vu) \mid uv \notin E\}.$$

As the graph contains no loops, all points of the form $r_{ij}(uu)$ are also added. Finally, we let $R^{\text{E}} = \uplus_{1 \leq i \neq j \leq k} R^{\text{E}}_{ij}$. See Figure 1.

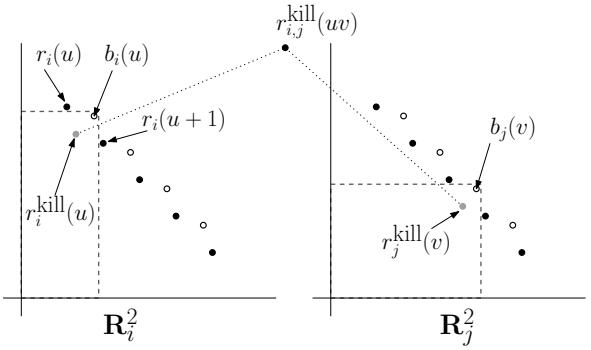


Figure 1: $r^{\text{kill}}_i(u)$ is the projection of $r^{\text{kill}}_{ij}(uv)$ to $\mathbb{R}^2_i$ and $r^{\text{kill}}_j(v)$ is the projection of $r^{\text{kill}}_{ij}(uv)$ to $\mathbb{R}^2_j$.

The overall construction. For $G = ([n], E)$ and $k > 0$ we construct point sets $R_{G,k} = R^{\text{scaffold}} \cup R^{\text{E}}$ and $B_{G,k} = \{0\} \cup B^{\text{scaffold}}$.

The two sets have size $\mathcal{O}(k^2 n^2)$ and the coordinates of the points can be encoded by $O(\log kn)$ many bits. Clearly the construction can be performed in time polynomial in both k and n.

Lemma 1 *G has a k–clique if and only if $H(R_{G,k}, B_{G,k}) = k + 1$.*

Proof. By the previous remarks, any feasible box H can contain at most $k+1$ points.

Let $v_1, \ldots, v_k$ be a clique of size k. We choose a box H with upper right coordinates (x_i, y_i) as $b_i(v_i) + (\varepsilon/2, \varepsilon/2)$.

H contains exactly one blue point from each of the $\mathbb{R}^2_i$, and also the origin, making it a total of $k+1$ points. We show that H is feasible. First, by definition H contains no point of R^{scaffold}, as $H_i \cap R^{\text{scaffold}}_i = \emptyset$. As the vertices are pairwise connected, H also does not contain any point from R^{E}.

Now assume that there is no clique of size k. Let H be any box containing $k+1$ points. We show that H is infeasible. If H contains a red point from one of

the $\mathbb{R}_i^2$'s, we are done. Otherwise, it can contain at most one blue point from each $\mathbb{R}_i^2$. Let v_i and v_j be two vertices corresponding to blue points contained in H that are not connected in G. As there is no k–clique, such a pair must exist. Then H also contains $r_{ij}^{\text{kill}}(v_iv_j)$, and thus H is infeasible. □

Theorem 2 *k–d–Bichromatic-Rectangle is $W[1]$-hard when parameterized with both the dimension d and the size of the solution k.*

3 The Maximum Empty Subinterval Problem

In this section, we consider a closely related problem.

Definition 2 *d–Maximum-Empty-Subinterval*
Given: *A set $R \subset \mathbb{R}^d$ and a rational number V*
Question: *Is there a box $H = [0, r_1) \times \cdots \times [0, r_d) \subset [0, 1)^d$ containing none of the points whose volume is at least V?*

Since the origin is included in the box, the planes will be considered separately again. In this construction, the analog of the rectangles containing a blue point from one of the $\mathbb{R}_i^2$ is now a rectangle that is large (of size C for some $0 < C < 1$ to be determined later). In each plane, there will be n large rectangles to choose from, corresponding to the vertices of G. It will only be possible to choose large rectangles from two different planes, if the corresponding vertices are connected in G.

The construction. Let $\mu > 1$ be a parameter to be specified later. In each $\mathbb{R}_i^2$ we put points

$$r_i(u) = \left(C\mu^{u-1}, \frac{1}{\mu^u}\right), 0 \le u \le n.$$

We set $R_i^{\text{scaffold}} = \{r_i(u) \mid 0 \le u \le n\}$ and $R^{\text{scaffold}} = \uplus_{1 \le i \le k} R_i^{\text{scaffold}}$. Then the rectangles with upper right corner $c_i(u) = \left(C\mu^{u-1}, 1/\mu^{u-1}\right), 1 \le u \le n$ are empty and have volume C.

If a rectangle has its upper right point anywhere else on $(x, C/x)$ or above, it contains a point from R^{scaffold}, and any other feasible rectangle has smaller size. So there are exactly n feasible rectangles of size C in each $\mathbb{R}_i^2$, each supported by two blocking points from R_i^{scaffold}. See Figure 2.

If the vertices corresponding to two different large rectangles in the planes $\mathbb{R}_i^2$ and $\mathbb{R}_j^2$ are not connected, we will add a point in the product $\mathbb{R}_{ij}^4$ that forbids these two rectangles to be chosen at the same time. Let $r_i^{\text{kill}}(u) = (C\mu^{u-2}, 1/\mu^u)$ (again, these points are themselves not added to the set R) and then define $r_{ij}^{\text{kill}}(uv) = r_i^{\text{kill}}(u) + r_j^{\text{kill}}(v)$. Then the set of all kill points is $R^{\text{E}} = \{r_{ij}^{\text{kill}}(uv) \mid i \ne j, uv \notin E\}$. Finally, the set of all points is defined as $R = R^{\text{E}} \cup R^{\text{scaffold}}$.

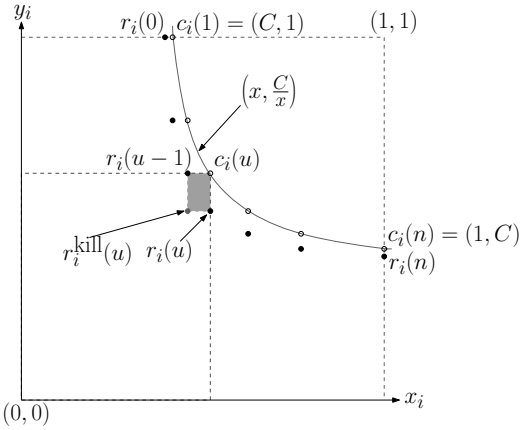


Figure 2: The plane $\mathbb{R}_i^2$. A rectangle selecting vertex u is indicated.

The size of R is $\mathcal{O}(n^2k^2)$, and if we set $\mu = 2$, all coordinates have size polynomial in the size of the input. Clearly the construction can be performed in time polynomial in k and n. The proof of the next lemma is analog to Lemma 1. The second part comes from the fact that if there is no k–clique, in at least one of the $\mathbb{R}_i^2$ we have to chose a rectangle of size at most C/μ.

Lemma 3 *G has a k–clique if and only if there is an empty box of size C^k. Otherwise, the largest empty box is of size at most C^k/μ.*

Theorem 4 *d–Maximum-Empty-Subinterval is $W[1]$-hard with respect to the dimension d.*

By Lemma 3, the ratio between a large box for a point set constructed from a positive instance and one constructed from a negative instance is at least μ. Since we can even choose $\mu = 2^{\mathcal{O}(|R|)}$ (this takes only polynomially many bits), we can even derive a result that is much stronger than Theorem 4:

Corollary 5 *It is $W[1]$-hard to approximate d–Maximum-Empty-Subinterval by a factor of $(1/2)^n$, where n is the number of points, and the parameter is the dimension d.*

4 The Star Discrepancy Problem

Now we show that computing the star discrepancy of a point set inside the unit cube is $W[1]$-hard.

Definition 3 *The* star discrepancy *of a point set $R \subseteq [0, 1]^d$ is defined as*

$$D^*(R) = \sup_{Y \in \mathcal{Y}} \left| vol(Y) - \frac{|Y \cap R|}{|R|} \right|$$

where $\mathcal{Y} = \{[0, x_1) \times \cdots \times [0, x_d) \mid x_i \in [0, 1]\}$ is the set of all half–open boxes containing the origin.

In order to adapt the above proof, there are two issues to deal with: First, we have to make sure that the maximum is attained for a large box with few points inside. Then, we have to argue that such a box really does not contain any points at all. For a graph G, let N be the total number of points in our construction from the previous section. Observe that any box that contains all points must have a volume of 1, as there are points with $x_i = 1$ and $y_i = 1$ for all $1 \leq i \leq k$. Thus, for all boxes B we have $\frac{|B \cap R|}{|R|} - vol(B) \leq \frac{N-1}{N}$, If we choose $\mu = 1 + \frac{1}{t}$ with $t = 2knN$ then the largest empty box is of size at least $C^k > (N-1)/N$ and thus the discrepancy is attained for an empty box: Adding a point can increase the volume by at most $1 - C^k$. But as $C^k > (1 - \frac{1}{N})$, picking such an extra point cannot increase the discrepancy. Constructing the set R with this value of μ, we see that G has a clique of size k, if and only if $D^*(R) = C^k$. Hence,

Theorem 6 *d–Star-Discrepancy is $W[1]$-hard with respect to the dimension d.*

5 Largest Empty Box Problem

Next, we consider a slightly more general problem where we have to find an empty box that does not necessarily contain the origin. Observe that in the previous construction, the box $[\varepsilon, 1)^{2k}$ for arbitrarily small ε does not contain any points but has volume almost 1. Still, by using a little trick we can fix this. From a graph G, we first construct the set R with the constant $C^k = 2/3$. Then, define the function lift: $\mathbb{R}^{2k} \to \mathbb{R}^{2k}$ as follows

$$\text{lift}(x) = (x'_1, \ldots, x'_{2k}) \text{ with } x'_i = \begin{cases} x_i & \text{if } x_i \neq 0 \\ \frac{1}{2} & \text{otherwise} \end{cases}$$

Now we apply the function lift to all points in the R. For the lifted point x, we call the $\mathbb{R}^2_i$ that the point was lifted from the *corresponding* $\mathbb{R}^2_i$.

Lemma 7 *Any box having volume at least $2/3$ contains a point x if and only if the projection onto the corresponding $\mathbb{R}^2_i$ contains the projection of x.*

Further, any box of volume $2/3$ must have its lower left endpoint inside $[0, 1/2)^{2k}$. Thus, any maximum empty box will contain the origin. As above, we now get that there is an empty box of volume C^k if and only if G has a k–clique.

Theorem 8 *d–Maximum-Empty-Box is $W[1]$-hard with respect to d.*

6 The Box Discrepancy Problem

The box discrepancy $D(R)$ of a point set R is defined analogously to the star discrepancy, except that now the boxes do not have to contain the origin. By merging the previous proofs, we can show the hardness of the corresponding decision problem: First, we add one point at the origin and one point at $(1, 1, \ldots, 1)$. Then, we construct the point set with the constants determined in Section 4 and lift it as in Section 5. Because of the two additional points, no box not having volume 1 can contain all points from R. Arguing as above, the box discrepancy is attained for a maximum empty box, and using the analog of Lemma 7, we get:

Theorem 9 *d–Box-Discrepancy is $W[1]$-hard with respect to the dimension d. The problem is also NP-hard when d is part of the input.*

References

[1] B. Aronov and S. Har-Peled. On approximating the depth and related problems. *SIAM J. Comput.*, 38(3):899–921, 2008.

[2] J. Backer and J.M. Keil. he mono- and bichromatic empty rectangle and square problems in all dimensions. In *Proc. of the 9th LATIN*, 2010. To appear.

[3] S. Cabello, P. Giannopoulos, C. Knauer, D. Marx, and G. Rote. Geometric clustering: fixed-parameter tractability and lower bounds with respect to the dimension. *ACM Trans. on Algorithms*. To appear.

[4] D.P. Dobkin, D.Eppstein, and D.P. Mitchell. Computing the discrepancy with applications to supersampling patterns. *ACM Trans. Graph.*, 15(4):354–376, 1996.

[5] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, November 1999.

[6] A. Dumitrescu and M. Jiang. On the largest empty axis-parallel box amidst n points, 2009. Manuscript.

[7] J. Eckstein, P. L. Hammer, Y. Liu, M. Nediak, and B. Simeone. The maximum box problem and its application to data analysis. *Comput. Optim. Appl.*, 23(3):285–298, 2002.

[8] M. Gnewuch, A. Srivastav, and C. Winzen. Finding optimal volume subintervals with k points and calculating the star discrepancy are NP-hard problems. *J. Complexity*, 25:115–127, 2009.

[9] E. Thiémard. An algorithm to compute bounds for the star discrepancy. *J. Complexity*, 17(4):850–880, 2001.

Nearest-neighbor queries with well-spaced points

Chris Gray*

Abstract

We assume we are given points in $\mathbb{R}^3$ that have the property that their Voronoi diagram restricted to some suitable bounding box consists of only fat cells. We call such points *well-spaced points*. We give a linear-sized data structure for finding the nearest neighbor to a query point among well-spaced points in $O(\log n)$ time. We further show how to make this data structure dynamic and how to extend the results to higher dimensions.

1 Introduction

In recent years, there has been much research in the computational-geometry community on what is known as *realistic-input models*. In this research, one usually limits some aspect of the input (for example, the number of "large" polyhedra that can intersect a "small" region) to some minimum or maximum value. The claim is that in most applications such a bound is almost never violated. In some cases, this restriction allows us to design algorithms that have better performance than lower bounds on unrestricted input would allow.

Much of the research to date on algorithms for realistic input has been done on groups of polygons or polyhedra. Some examples include bounds on the union complexity of [2], data structures for ray-shooting in [3], and algorithms for computing the depth orders of [1] collections of fat objects. However, there has been relatively less done on point clouds. One reason for this is that it is not obvious which aspect of the point cloud to limit. However, in the meshing community, one nice realistic-input model has emerged: well-spaced points. We define these terms more precisely in Section 1.1, but intuitively a point set P (usually in $\mathbb{R}^3$, but some of our results also apply in higher dimensions) is well-spaced if the Voronoi diagram of P satisfies the condition that every cell is fat.

In this paper, we provide a data structure for finding the nearest point in P to a query point. We then show how to make this data structure dynamic. We also describe a modification of the Voronoi diagram that allows us to obtain similar results in terms of space and query-time complexity in higher dimensions.

*Department of Computer Science, TU Braunschweig. gray@ibr.cs.tu-bs.de This research was funded by the German Ministry for Education and Research (BMBF) under grant number 03NAP14 "ADVEST".

1.1 Definitions

The *Voronoi diagram* of a set of points $P = \{p_1, \ldots p_k\}$ in $\mathbb{R}^d$ is the division of space into cells $\{c_1, \ldots, c_k\}$ (where each cell c_i corresponds to point p_i), where each cell c_i has the property that every point $p \in c_i$ is closer to p_i than to any other point in P. The distance metric in this definition can be changed, but we use the standard Euclidean distance in this paper. Note that some of the cells of any Voronoi diagram are unbounded. The *Delaunay graph* of a point set P is the graph formed by creating an edge (p_i, p_j) if and only if the Voronoi cells of p_i and p_j are neighbors in $\mathrm{VD}(P)$.

The most important realistic input model that we use in this paper is the notion of *fatness*. There are many different definitions of fatness, but most are asymptotically equivalent for convex objects. In this definition, let $\mathrm{vol}(o)$ denote the volume of the object o.

Definition 1 *Let β be a constant with $0 < \beta \leq 1$. An object o in $\mathbb{R}^d$ is defined to be β-fat if, for any ball b that has a center inside o and that does not completely contain o, we have $\mathrm{vol}(b \cap o) \geq \beta \cdot \mathrm{vol}(b)$.*

We say that a set of points $P = \{p_1, \ldots, p_k\}$ is a set of *β-well-spaced points* if, for every point p_i, the cell c_i in the Voronoi diagram of P restricted to a suitable bounding box is β-fat. See Figure 1.

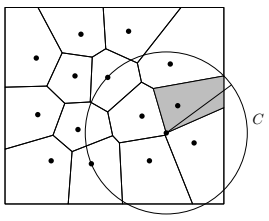


Figure 1: A set of well-spaced points. The circle C determines the fatness of the gray Voronoi cell. Since this is the least-fat cell, this determines the well-spacedness of the point set.

Fatness is only one realistic input model. Another model that we make use of is the *low-density* model. Following Van der Stappen [16], we define the *density* of a set of objects to be the smallest number λ

where any ball B is intersected by at most λ objects o with $\text{size}(o) \geq \text{size}(B)$. The following lemma relates fatness and density.

Lemma 1 (Van der Stappen [16]) *Any set of disjoint β-fat objects has density λ for some value of λ that is a constant multiple of $1/\beta$.*

1.2 Related Work

The notion of well-spaced points was introduced in the thesis of Talmor and the preceding papers [15, 11]. Since then, it has been used in many papers related to meshing [8, 7]. As mentioned earlier, well-spaced points can be seen as a realistic-input model. Other realistic-input models have been proposed for points, and work has been done related to Voronoi diagrams for these point sets. Erickson [6] examined the relationship of the *spread*—the ratio of the largest distance between any pair of points to the smallest distance between any pair of points—of a set of points to the complexity of the Delaunay graph of those points. He showed that the Delaunay graph of a set of points with spread Δ has complexity $O(\Delta^3)$. Spread and well-spacedness, however, are not directly comparable: well-spaced points can have arbitrarily high spread, and that point-sets that have constant spread can be not well-spaced.

The nearest-neighbor search problem in high dimensions has been extensively studied. Exact searches for the nearest neighbor have long been known to be plagued by what is called the "curse of dimensionality". This is a conjecture that states that either the storage or search time for an exact nearest-neighbor search problem must grow exponentially in d for certain values of n. To date, this conjecture appears to be holding true, both in theory and practice. The best currently-known algorithm, by Liu [10], uses $O(n^d)$ space and requires $O(d^5 \log n)$ search time. For more on the subject, as well as an overview of the results obtained for the approximate nearest-neighbor problem, see the survey by Indyk [9].

The data structure recently developed by Hudson and Türkoğlu [8] in some ways is the most similar to ours. In $\mathbb{R}^3$, their data structure can be built in the same time bound as ours, and with a small modification in the query algorithm, can find exact nearest neighbors in $O(\log n)$ time. It is also dynamic (though it only handles insertions). We believe, however, that our data structure compares well with that of Hudson and Türkoğlu. Our results have the advantage that they are based on the Voronoi diagram—a well-studied and widely-used decomposition of space. Also, our results are a bit more general (at least as stated): any set of well-spaced points will do and the order of the input does not matter. The data structure by Hudson and Türkoğlu and its properties are tuned to a specific family of meshing algorithms. Finally, we use definitions for realistic-input models that are more standard in the computational-geometry community. We hope that in doing so, more research into other data structures for well-spaced points will emerge.

2 Size of Voronoi diagrams of well-spaced points

We begin by stating some nice properties of well-spaced points and their Voronoi diagrams. To save space, we omit proofs from this abstract, but include them in the full version of the paper.

Given a set of well-spaced points, we might ask questions about the complexity of their Voronoi diagrams. With the tools that we have introduced so far, it is quite simple to see that this complexity is linear in the number of points.

Lemma 2 *The complexity of the Voronoi diagram of a set P of β-well-spaced points is $O((1/\beta)n)$.*

Note that the complexity of a Voronoi diagram in $\mathbb{R}^d$ without any restrictions on the fatness of the cells is $\Theta(n^{\lceil d/2 \rceil} + n \log n)$ [13]. This implies that the complexity of the Voronoi diagram of well-spaced points in three dimensions is better by a factor of $O(\beta n)$. However, as d increases, the complexity of the cells of the Voronoi diagram also increases: the number of facets of all dimensions in a convex d-dimensional polyhedron with n vertices (or, dually, n $d-1$-dimensional facets) is $O(n^{\lfloor d/2 \rfloor})$ [17]. Since some (large) Voronoi diagram cells can have many (small) neighbors, Voronoi diagrams of well-spaced points[1] can have large complexity.

Since a large Voronoi diagram cell can only have small neighbors that are very near the site of the cell (or else they would be larger), we can define a modified Voronoi diagram cell that has constant complexity. Let p_i be the site of a cell c_i. Further, let s_i be a hypersphere centered at p_i with radius $\varepsilon \cdot \text{size}(c_i)$ for some constant $\varepsilon > 0$. We define the *modified Voronoi cell* of c_i to be $s_i \cup c_i$—see Figure 2.

Lemma 3 *Let $1 > \varepsilon > 0$ be a constant and P be a set of β-well-spaced points. The complexity of the modified Voronoi cell of any point in P is $O(1/(\varepsilon^d \beta))$.*

In Section 3, we give a data structure that uses these modified cells to find the exact nearest neighbor of a query point. This data structure uses $O(n)$ space and has $O(d \log n)$ query time.

[1] This is only true using our (more standard) definition of fatness. When fatness is defined using the ratio of radii of inner- and outer-balls both centered at the site of the Voronoi diagram cell (as in [7]), very large cells cannot be adjacent to very small cells, which means that the complexity of any cell is guaranteed to be constant.

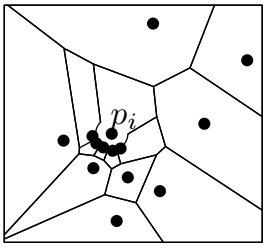


Figure 2: The modified Voronoi cell of the point p_i.

3 Nearest-neighbor query data structure

The data structure that we construct to answer nearest-neighbor queries amongst well-spaced points is essentially an object-BAR-tree [4] where the objects are the cells of the Voronoi diagram of the points. We describe the object-BAR-tree fairly thoroughly because we use features in this and later sections.

The object-BAR-tree is, as the name implies, an extension of the BAR-tree. The BAR-tree $\mathcal{T}$ (or *Balanced Aspect Ratio*-tree) introduced by Duncan [5], is a type of binary space partition on points with the following properties (among others):

- Each cell of $\mathcal{T}$ has constant fatness and constant complexity.
- Each leaf node of $\mathcal{T}$ contains a constant number of points.
- The depth of $\mathcal{T}$ given n input points is $O(\log n)$.

The object-BAR-tree maintains these properties for objects with low density. To be more precise, the depth of $\mathcal{T}$ is $O(\lambda \log n)$, so if λ is constant, then the depth is $O(\log n)$. The object-BAR-tree also has the property that its *size*—the number of fragments that the objects are split into—is linear in the number of objects. The fatness of the cells of $\mathcal{T}$ can be bounded by a constant that is not related to the input objects.

The construction algorithm for an object-BAR-tree in not very complicated. We sketch it here. It begins by associating a number[2] of points, known as *guards*, with every object. A BAR-tree $\mathcal{T}$ is then constructed using these guards as input. The objects are then distributed to *object lists* in the leaf nodes of $\mathcal{T}$. An object o is put on the object list in the leaf node ν if o intersects ν.

Since we assume that the objects have low density and since the cells of $\mathcal{T}$ are fat (and thus also low density), we can easily see that the number of intersections between objects and leaf nodes is linear in the number of objects and leaf nodes. This implies that the average number of such intersections per object is constant. However, in some situations it is desirable to have a stronger guarantee. Thus, we describe a modification where each object is in at most a constant number of object lists.

In the modified version of the object-BAR-tree $\mathcal{T}$, we place an object o in an object list of a (possibly non-leaf) node ν of $\mathcal{T}$ if and only if $\text{size}(o) > \text{size}(\nu)$ or ν is a leaf node which is larger than o, or $\text{size}(o) \leq \text{size}(\text{parent}(\nu))$, and o intersects ν.

Using these properties, we obtain:

Lemma 4 *Let o be an object and S be a set of objects where $\{o\} \cup S$ has density λ and where the objects in $\{o\} \cup S$ are disjoint. Assume that $\mathcal{T}$ is a modified object-BAR-tree built on the objects in $\{o\} \cup S$. Then the number of object lists in $\mathcal{T}$ containing o is $O(\lambda)$. The number of objects in any object list is also $O(\lambda)$.*

The query procedure for the modified object-BAR tree is quite simple: for each level of the modified object-BAR tree, find the node ν containing the query point p. For each of the Voronoi diagram cells in the object list of ν, find the cell with the site that is closest to p. Since there are $O(\lambda)$ objects in the object list of ν, we can find the closest site by simply finding the distance between p and every site in the object list. This takes $O(d\lambda)$ time per level, meaning that it takes $O(d\lambda \log n)$ time overall. This procedure works with the modified Voronoi cells as well.

4 Dynamic data structure

The BAR-tree data structure can be made to support insertion and deletion operations using a technique known as *partial rebuilding* [12]. Essentially, we insert and delete nodes using the standard tree operations, allowing the tree to become slightly imbalanced. When we determine that the tree has become too imbalanced, we rebalance the tree by rebuilding part of it from scratch. Since most of the insertion and deletion operations can be completed very quickly, we can argue that the amortized cost of any insertion or deletion is fairly low. This technique was first applied to BAR-trees by Duncan [5]. We describe it for completeness and then show how it can be applied to object-BAR-trees.

To allow for the imbalance in the tree structure, we create a parameter η with $0 < \eta < 1$. We say that a tree is *η-balanced* if and only if, for every node ν, no grandchild of ν has more than η times the number of children of ν. It is easy to see that any tree that is η-balanced has depth $O(\log_{2-\eta} n)$.

To insert a point p in a dynamic η-balanced BAR-tree $\mathcal{T}$, we first find the leaf node ν of the BAR-tree that contains p. We save the path of nodes $\{\nu_1, \ldots, \nu_k\}$ that are visited from the root to ν. We then check whether inserting p as a child of ν would

[2] Unfortunately, the number of guards is quite high in higher dimensions: approximately $2^{3d(d-1)}$. There is no lower bound given, however, and it is not clear that this number could not be reduced.

cause any of the nodes in $\{\nu_1, \ldots, \nu_k\}$ to lose the property that they are η-balanced. If the insertion does not cause any of these nodes to lose the property that they are η-balanced, then we insert p according to the normal rules of the BAR-tree.

If adding p as a child of ν would cause one of the ancestors of ν to lose its η-balanced property, we find the highest node in the tree where this would be the case. Call this node ν_j. We remove the entire subtree rooted at ν_j from $\mathcal{T}$ and build a BAR-tree on the points that have been removed as well as p. We attach this new BAR-tree (which is balanced, since it is a BAR-tree) to ν_j. In so doing, we have added p to $\mathcal{T}$. Deleting a point p from $\mathcal{T}$ can be handled analogously.

Lemma 5 (Duncan [5]) *A point p can be inserted into or deleted from a BAR-tree in $O(\log^2 n)$ amortized time.*

We now move to object-BAR-trees. We argue that the same ideas can be applied to these trees. This question has been briefly considered before in the context of I/O-efficient algorithms by Streppel and Yi [14]. They note that the addition of object lists to the nodes creates a complication: each object can intersect an arbitrarily high number of nodes. However, we avoid this problem by using the modified object-BAR-trees from the last section.

Theorem 6 *An object o can be inserted into or deleted from a modified object-BAR-tree $\mathcal{T}$ in $O(\log^2 n)$ amortized time.*

5 Conclusions

In this paper, we presented a new data structure that we can use to query the nearest neighbor in a set of well-spaced points. We then showed how to make this data structure dynamic.

We also defined a modified data structure that work in higher dimensions. Unfortunately, the dependence of the time and space complexity on the dimension is quite high, making these algorithms tractable only for dimensions that are $O(\sqrt{\log n})$. The bottleneck in this regard is the object-BAR-tree structure. If the relationship between the dimension and the number of guards required by this structure could be reduced to even $O(2^d)$, then our data structure would work in dimensions up to $O(\log n)$.

This work raises some open problems beyond what have been mentioned previously. First, we would like to use the data structures presented here to develop a simpler algorithm to compute Voronoi Diagrams in higher dimensions. We conjecture that one can use the dynamic nearest-neighbor data structure presented in Section 4 to design an incremental algorithm that is simpler than existing algorithms in higher dimensions. We would also like to investigate whether there is any relationship between well-spaced points and other realistic-input models.

Acknowledgments

We thank Don Sheehy for bringing this problem to our attention and Björn Hendriks for simulating discussions related to this topic.

References

[1] B. Aronov, M. de Berg, and C. Gray. Ray shooting and intersection searching amidst fat convex polyhedra in 3-space. *Computational Geometry*, 41:68–76, 2008.

[2] M. de Berg. Improved bounds on the union complexity of fat objects. *Discrete and Computational Geometry*, 40(1):127–140, July 2008.

[3] M. de Berg and C. Gray. Vertical ray shooting and computing depth orders for fat objects. *SIAM Journal on Computing*, 38(1):257–275, 2008.

[4] M. de Berg and M. Streppel. Approximate range searching using binary space partitions. *Computational Geometry*, 33(3):139–151, February 2006.

[5] C. A. Duncan. *Balanced Aspect Ratio Trees.* PhD thesis, Johns Hopkins University, 1999.

[6] J. Erickson. Dense point sets have sparse delaunay triangulations or "... but not too nasty". *Discrete and Computational Geometry*, 33(1):83–115, 2005.

[7] B. Hudson, G. L. Miller, T. Phillips, and D. Sheehy. Size complexity of volume meshes vs. surface meshes. In *SODA '09*, pages 1041–1047, 2009.

[8] B. Hudson and D. Türkoğlu. An efficient query structure for mesh refinement. In *CCCG '08*, 2008.

[9] P. Indyk. Nearest neighbors in high-dimensional spaces. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, pages 877–892. CRC Press, second edition, 2004.

[10] D. Liu. A note on point location in arrangements of hyperplanes. *Inf. Process. Lett.*, 90(2):93–95, 2004.

[11] G. L. Miller, D. Talmor, S.-H. Teng, and N. Walkington. A Delaunay based numerical method for three dimensions: generation, formulation, and partition. In *STOC '95*, pages 683–692, 1995.

[12] M. Overmars. *The Design of Dynamic Data Structures.* 1983.

[13] R. Seidel. On the number of faces in higher-dimensional voronoi diagrams. In *SCG '87*, pages 181–185, New York, NY, USA, 1987. ACM.

[14] M. Streppel and K. Yi. Approximate range searching in external memory. In *Algorithms and Computation*, pages 536–548. 2007.

[15] D. Talmor. *Well-Spaced Points for Numerical Methods.* PhD thesis, CMU, August 1997.

[16] A. F. van der Stappen. *Motion Planning Amidst Fat Obstacles.* PhD thesis, Utrecht University, 1994.

[17] G. M. Ziegler. *Lectures on polytopes.* Springer-Verlag, New York, 1995.

Removing Local Extrema from Imprecise Terrains

Chris Gray* Frank Kammer† Maarten Löffler‡ Rodrigo I. Silveira§

Abstract

In this paper, we study imprecise terrains, that is, triangulated terrains with a vertical error interval in the vertices. We study the problem of removing as many local extrema (minima and maxima) from the terrain as possible. We show that removing only minima or only maxima can be done optimally in $O(n \log n)$ time, for a terrain with n vertices, while removing both at the same time is NP-hard. To show hardness, we exploit a connection to a graph problem that is a special case of 2-Disjoint Connected Subgraphs, a problem that has received quite some attention lately in the graph theory community. This special case of 2-Disjoint Connected Subgraphs is shown NP-hard.

1 Introduction

A triangulated (or polyhedral) terrain is a planar triangulation with a height associated with each vertex. This results in a bivariate and continuous function, defining a surface that is often called a 2.5-dimensional (or 2.5D) terrain.

Even though in computational geometry it is usually assumed that the input data is exact, in practice, terrain data is most of the time imprecise. The sources of imprecision are many, starting from the methods used to acquire the data, which are ultimately based on error-prone measuring devices. Often such methods produce heights with a known error bound or return a height interval rather than a fixed height value. Even though terrain data may contain error also in the x, y-coordinates, we consider imprecision only in the z-coordinate. This simplifying assumption is justified by the fact that error in the x, y-coordinates will most likely produce elevation error. Moreover, often the data provided by commercial terrain data suppliers only reports the elevation error [2].

We describe an *imprecise terrain* by a set of n vertical intervals in $\mathbb{R}^3$, together with a triangulation of the projection. See Figure 1(a). We say that a triangulated terrain is a *realization* of an imprecise terrain if it has the same triangulation in the projection, and exactly one vertex on each interval. The large number of different realizations of an imprecise terrain leads naturally to the problem of finding one that is best according to some criterion. Problems that have been studied in this context include finding smooth realizations and shortest paths [3, 5, 8].

1.1 Removing Local Extrema

In this paper, we attempt to solve the *minimizing-minima*, the *minimizing-maxima*, and the *minimizing-extrema problem on imprecise terrains*, i.e., we attempt to find the realization of an imprecise terrain (by placing the imprecise points within their intervals) that minimizes the number of *local minima*, *local maxima* and *local extrema*, respectively. A local minimum (or pit) is usually defined as a point (or larger area of constant height) that is surrounded by only higher points, or that has no lower neighboring point. A local maximum (or peak) is defined analogously, as a point surrounded by lower points or without higher neighbors. Each local minimum and local maximum is also a local extremum.

When terrains are used for land erosion, landscape evolution, or hydrological studies, it is generally accepted that the majority of local extrema in terrain models are spurious, caused by errors in the data or model production. A terrain model with many pits or peaks does not represent the terrain faithfully, and moreover, in the case of pits, it can create problems in water flow routing simulations. For this reason the removal of local minima from terrain models is a standard preprocessing requirement for many uses of terrain models [11, 13].

Much research has been devoted to the problem of removing local minima from (precise) terrains, although most of the literature assumes a raster (grid) terrain (e.g. [9, 13]). Only a few algorithms have been proposed for triangulated terrains, mainly in the context of optimal higher order Delaunay triangulations [1, 6]. In particular, Gudmundsson *et al.* [6] show that the best possible number of both local minima

*Department of Computer Science, TU Braunschweig, Germany, gray@ibr.cs.tu-bs.de Funded by the German Ministry for Education and Research (BMBF) under grant number 03NAPI4 "ADVEST".

†Institut für Informatik, Universität Augsburg, Germany, kammer@informatik.uni-augsburg.de

‡Computer Science Department, University of California, Irvine, USA, mloffler@uci.edu Funded by the U.S. Office of Naval Research under grant N00014-08-1-1015.

§Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, Spain, rodrigo.silveira@upc.edu Supported by the Netherlands Organization for Scientific Research (NWO).

and local maxima can be removed from first-order Delaunay triangulations in $O(n \log n)$ time. Silveira and Van Oostrum [10] study moving vertices vertically in order to remove all local minima with a minimum cost, but do not assume bounded intervals.

1.2 P2-MaxCon

As we see later, there is a strong connection between the problem of removing local extrema from imprecise terrains and a graph problem, which we will call Planar 2-Disjoint Maximally Connected Subgraphs (or P2-MaxCon, for short). This problem takes as input a planar graph, of which two subsets of the vertices are colored red and blue. The object is to color the remaining vertices in such a way that the total number of connected components of both colors is as small as possible. This problem is very much related to the 2-Disjoint Connected Subgraphs problem, which is the same except that the graph is not required to be planar and the objective is to make the red and blue subgraphs both completely connected. This problem is known to be very hard, and has recently received some attention in the graph theory community. For example, it has been shown that 2-Disjoint Connected Subgraphs is NP-hard even when there are only two red vertices [12]. See also [7] for a related result.

1.3 Results

We first study the minimizing-minima and the minimizing-maxima problem on imprecise terrains. We present a relatively simple algorithm that removes local minima or local maxima optimally in $O(n \log n)$ time. Then we prove that minimizing the number of local extrema is NP-hard. This is achieved in two steps. First we reduce our problem from a graph problem that we call P2-MaxCon, which is a special case of 2-Disjoint Connected Subgraphs. We then show that P2-MaxCon is NP-hard. As a further result in the full version [4], we can show with a more sophisticated proof that the minimizing-extrema problem on imprecise terrains cannot be approximated in polynomial time within a constant unless P=NP.

2 Removing local minima

We propose an efficient algorithm based on the idea of selectively *flooding* parts of the terrain that finds the realization with the smallest number of local minima. The algorithm begins with all vertices as low as possible, and simulates flooding parts of the terrain.

Algorithm We sweep a plane vertically, starting at the lowest point on any interval and moving upwards in the z direction. As the plane moves up, it *pulls* some of the vertices with it, whose height change together with the plane. At any moment during the sweep, each vertex is in one of three states: (i) Moving, if it is currently part of a local minimum, and is moving up together with the sweep plane. (ii) Fixed at a height lower than the current one. (iii) Unprocessed, if it has not been reached by the sweep plane yet.

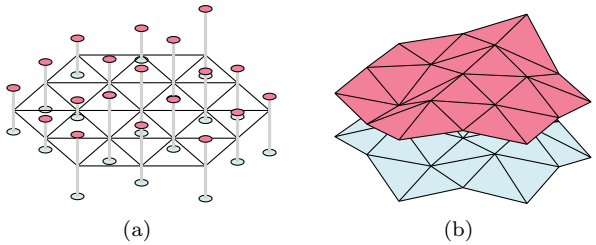


Figure 1: (a) An example of an imprecise terrain. (b) The same terrain, shown by drawing the floor and the ceiling.

As the sweep plane moves upwards, we distinguish two types of events: (i) The plane reaches the beginning (lowest end) of the interval of a vertex, (ii) The plane reaches the end (highest end) of the interval of a vertex. When an event occurs, let v denote the vertex whose interval just began or ended, and let h be the current height of the plane. Note that all fixed vertices are fixed at a height lower than h.[1]

An event of type (i) can create a number of situations. If v has a neighbor that is already fixed, then v will never be a local minimum, thus v is fixed at its lowest possible height, h. Moreover, if some other neighbor of v is currently part of a local minimum (i.e. is moving), then all the vertices that are part of that local minimum become fixed at h, and automatically stop being a minimum. This occurs for each neighbor of v that is currently part of a local minimum. If all neighbors of v are currently unprocessed, then v will become a new local minimum, and will start to move up together with the plane. Finally, if no neighbor is fixed but some neighbor is moving, thus is part of a local minimum, then v will join that existing local minimum and also start to move up together with the plane (note that if there is more than one local minimum that becomes connected to v, at this step they all merge into one).

Events of type (ii), when an interval ends, are easier to handle. If v is fixed, nothing occurs. If v was moving, then v and all of the vertices in the local minimum containing v become fixed at h.

We prove the correctness of this algorithm in the full version [4] using an simple induction argument on the events.

[1] For simplicity we assume in this description that all interval heights are different. The removal of this assumption does not pose any problem for the algorithm.

Sorting the interval ends for the sweep requires $O(n \log n)$ time. The rest of the steps can be implemented in linear time. Every vertex only starts and stops moving once, so events can be charged to these vertices. We can also merge moving minima in constant time by representing each moving local minimum as a tree of components that were merged. A more detailed proof of the running time is left for the full version [4].

Theorem 1 *The minimizing-minima and the minimizing-maxima problem on imprecise terrains can be solved in $O(n \log n)$ time.*

It is interesting to note that when a group of k connected vertices at the same height without any lower neighbors is regarded as k different local minima, the problem can be proved NP-hard. Details are omitted due to lack of space.

3 Removing all local extrema

We now move to the problem of removing both local minima and local maxima, that is, removing as many local extrema as possible at the same time. In the full version [4], we show that we can in fact apply the algorithm of Section 2 to remove the local minima, or to remove the local maxima, and this will result in two solutions that do not intersect, effectively narrowing down the solution space. However, we must now find a compromise between the two. It turns out that finding such a compromise is NP-hard. Firstly, show that this problem is as hard as P2-MaxCon.

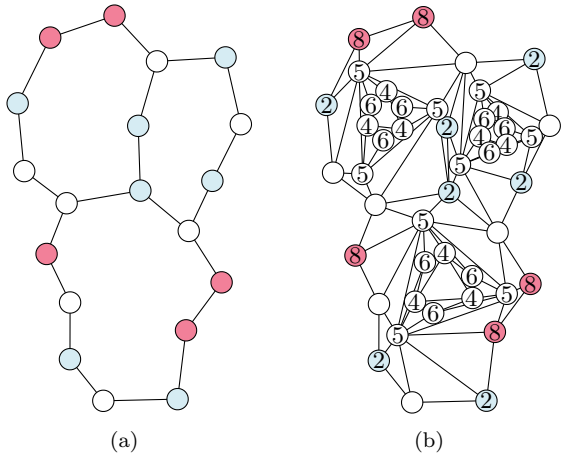
The reduction takes the input to P2-MaxCon—a planar graph with red, blue and white vertices—and builds an imprecise terrain from it. We will first embed the graph in the plane with straight line edges and convex faces. We then turn all red vertices into precise vertices at height 8, and all blue vertices into precise vertices at height 2. Finally, we turn the white vertices into imprecise vertices with interval $[2, 8]$.

The problem of minimizing extrema on this graph is equivalent to that of minimizing connected components after recoloring. This is due to the fact that the only way to remove local minima in this "terrain" is by connecting the minima to each other by assigning the white vertices at height 2. Similarly, the local maxima can only be removed by assigning white vertices at height 8.

In order to have a proper imprecise terrain, we still need to triangulate the graph. We show how to do this in detail in the full version [4], but the idea can be seen in Figure 2(b).

3.1 P2-MaxCon is NP-hard

We prove that P2-MaxCon is NP-hard by a reduction from planar 3-SAT. In this problem, the normal 3-SAT problem is restricted so that the bipartite graph connecting variables and clauses is planar. We call this graph $G_S = ((V \cup C), E)$, where an edge $e = (v, c) \in E$ if variable v is in clause c. As is usual in such reductions, we first embed G_S in the plane so that none of the edges in E cross. We then replace the vertices and edges in the embedding with "gadgets".



(a) (b)

Figure 2: (a) An instance of P2-MaxCon. (b) In the output, we fixed the red vertices at height 8, and the blue at height 2. The remaining white vertices are intervals $[2, 8]$. The rest of the vertices are added to make sure that the graph is triangulated, and that the new vertices do not interfere with the number of local extrema.

The variable gadget is simply a white vertex. We show below that coloring the vertex red is equivalent to setting the corresponding variable to `true` and coloring the vertex blue is equivalent to setting the corresponding variable to `false`.

Another gadget that we use is the inverter gadget, shown in Figure 3(a). This gadget consists of two white vertices and k red and k blue vertices. Each colored vertex is connected to both white vertices. This gadget ensures that one of the white vertices must be colored red and the other one blue, because otherwise there will be k components in the output. To ensure that this is unacceptable, we make k at least as large as the number of gadgets in our construction.

A clause gadget is a collection of 3 inverter gadgets, and 4 extra red vertices. These are all connected as shown in Figure 3(c). The red vertices form one large component as long as at least one of the white vertices adjacent to the central red vertex is colored red.

Finally, we create edge gadgets to connect variable gadgets to clause gadgets. An edge gadget is simply a chain of inverter gadgets. See Figure 3(d). If a variable v is negated in clause c, then we replace the edge (v, c) with a chain of an odd number of inverter gadgets, otherwise, we use an even-length chain. Since the number of inverter gadgets between a variable gadget and one of the clause gadgets that it is con-

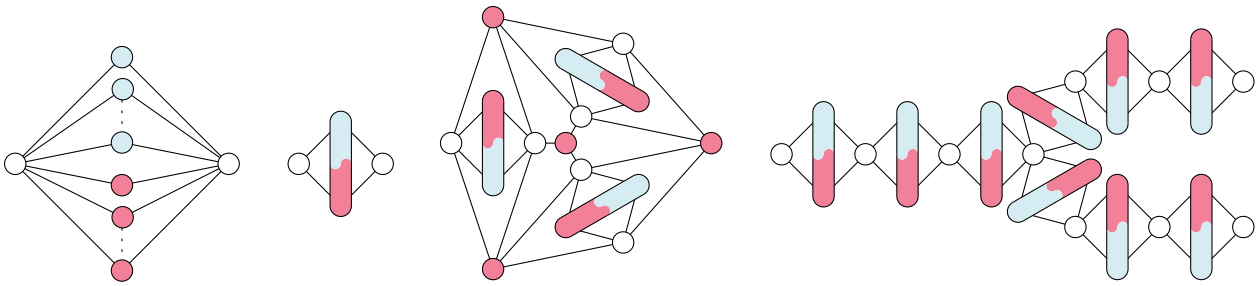


Figure 3: (a) An inverter, consisting of k red and k blue vertices. (b) Symbolic representation of an inverter. (c) In a clause, we connect three inverter gadgets using four extra red vertices. (d) We can chain inverter gadgets. The white vertices in a chain must always be colored alternately red and blue.

nected to determines the color of the final white vertex in the chain, we can see that coloring a variable gadget red corresponds to the final white vertex in a chain to a clause in which that vertex is not negated being colored red. This implies that coloring a variable gadget red is equivalent to setting its value to `true` and that coloring a variable gadget blue is equivalent to setting its value to `false`.

The total number of connected components is equal to the number of white vertices in the construction, minus 2 per clause since the red components are connected, plus the number of unsatisfied clauses. Hence, minimizing the number of connected components involves determining whether the 3-SAT clause can be satisfied completely, which proves the following.

Theorem 2 P2-MaxCon *is NP-hard.*

Corollary 3 *The minimizing-extrema problem on imprecise terrains is NP-hard.*

4 Discussion

We have shown that the minimizing-minima and the minimizing-maxima problem on imprecise terrains can be solved in $O(n \log n)$ time, while solving the minimizing-extrema problem on imprecise terrains cannot be solved in polynomial time unless P=NP. We have also shown that P2-MaxCon is NP-hard, which constitutes the first result about 2-Disjoint Connected Subgraphs for planar graphs.

Acknowledgments

We thank Jeff Phillips for proposing the problem in Section 3.

References

[1] T. de Kok, M. van Kreveld, and M. Löffler. Generating realistic terrains with higher-order Delaunay triangulations. *Comput. Geom. Theory and Appl.*, 36:52–65, 2007.

[2] P. F. Fisher and N. J. Tate. Causes and consequences of error in digital elevation models. *Progress in Physical Geography*, 30(4):467–489, 2006.

[3] C. Gray and W. Evans. Optimistic shortest paths on uncertain terrains. In *Proc. 16th Canadian Conference on Comput. Geom.*, pages 68–71, 2004.

[4] C. Gray, F. Kammer, M. Löffler, and R. I. Silveira. Removing Local Extrema from Imprecise Terrains. `arXiv:1002.2580`, 2010.

[5] C. Gray, M. Löffler, and R. I. Silveira. Smoothing imprecise 1.5D terrains. In *Proc. 6th Int. Workshop on Approx. and Online Alg.*, pages 214–226, 2009.

[6] J. Gudmundsson, M. Hammar, and M. van Kreveld. Higher order Delaunay triangulations. *Comput. Geom. Theory Appl.*, 23:85–98, 2002.

[7] F. Kammer and T. Tholey. The complexity of minimum convex coloring. In *Proc. 19th Int. Symp. on Alg. and Comp.*, pages 16–27, 2008.

[8] Y. Kholondyrev and W. Evans. Optimistic and pessimistic shortest paths on uncertain terrains. In *Proc. 19th Canad. Conf. on Comput. Geom.*, pages 197–200, 2007.

[9] L. W. Martz and J. Garbrecht. An outlet breaching algorithm for the treatment of closed depressions in a raster DEM. *Computers & Geosciences*, 25:835–844, 1999.

[10] R. I. Silveira and R. van Oostrum. Flooding countries and destroying dams. In *Proc. 10th Workshop on Alg. and Data Struc.*, pages 227–238, 2007.

[11] A. Temme, J. Schoorl, and A. Veldkamp. Algorithm for dealing with depressions in dynamic landscape evolution models. *Computers & Geosciences*, 32:452–461, 2006.

[12] P. van 't Hof, D. Paulusma, and G. Woeginger. Partitioning graphs in connected parts. *Theoretical Computer Science*, 410:4834–4843, 2009.

[13] Q. Zhu, Y. Tian, and J. Zhao. An efficient depression processing algorithm for hydrologic analysis. *Computers & Geosciences*, 32:615–623, 2006.

Recursive tilings and space-filling curves with little fragmentation

Herman Haverkort*

Abstract

This paper defines the Arrwwid number of a recursive tiling (or space-filling curve) as the smallest number a such that any ball Q can be covered by a tiles (or curve fragments) with total volume $O(\text{volume}(Q))$. Recursive tilings and space-filling curves with low Arrwwid numbers may be applied to optimise disk, memory or server access patterns when processing sets of points in $\mathbb{R}^d$. This paper presents recursive tilings and space-filling curves with optimal Arrwwid numbers. When $d \geq 3$, regular cube tilings and space-filling curves cannot have optimal Arrwwid number; alternatives with better Arrwwid numbers are presented.

1 Introduction

Consider a set of data points in a bounded region U of $\mathbb{R}^2$, stored on disk. A standard operation on such point sets is to retrieve all points that lie inside a certain query range, for example a circle or a square. To prevent large delays because of disk head movements while answering such queries, it is desirable that the points are stored in a clustered way [1]. Similar considerations arise when storing spatial data in certain types of distributed networks [3] or when scanning objects to render them as a raster image; in the latter case it is desirable that the pixels that cover any particular object are scanned in a clustered way, so that the object does not have to be brought into cache too often [4]. For ease of explanation, we focus on the application of clustering to storing points on disks.

We could try to achieve a good clustering in the following way. We divide U into *tiles*. The tiles could, for example, form a regular grid of hexagons (Fig. 1). Now we store the points in each tile as a contiguous block on disk. To answer a query, say with a region Q bounded by a circle, we compute which tiles intersect Q. For each intersecting tile, we move the disk read head to the position where the first point in that tile is stored, and then we retrieve all points in the tile, scanning them sequentially without further delays from disk head movements. Since some of the tiles that intersect Q may lie partially outside Q, some of the points thus retrieved may be *false answers*: they are no answers to our query and need to be filtered out in post-processing.

*Dept. of Computer Science, Eindhoven University of Technology, the Netherlands, cs.herman@haverkort.net

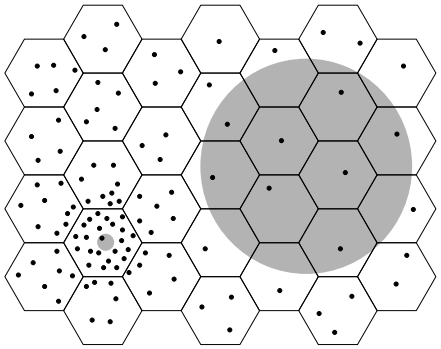


Figure 1: Sorting points into (non-recursive) tiles. Queries with small query ranges (shaded disk on the left) may necessitate scanning the full contents of a crowded tile. Queries with large ranges (shaded disk on the right) may necessitate looking up many tiles.

The approach sketched above may work well if the following conditions are met: (i) A couple of tiles suffice to cover Q (so that we do not have to move the disk read head to the starting point of another tile too often); (ii) The tiles that cover Q are not much larger than Q (so that they do not contain too many false answers). However, as illustrated in Fig. 1, these two conditions can be hard to meet if we choose any fixed tile size and the density of the points stored varies by region or in time. To avoid this problem, we can use *recursive* tilings. A recursive tiling is a subdivision of U into tiles, that are each subdivided into tiles recursively. We store the data points in such a way that for each tile, on any level of recursion, the data points within that tile are stored as a contiguous sequence on the disk. Hopefully we can now cover any disk-shaped region Q with a small set of tiles from the level of recursion where the tiles have size proportional to Q. Thus we would satisfy condition (i) and (ii) at the same time. For a precise problem statement, we define the Arrwwid number of a recursive tiling of a region U as follows:

Definition 1 *The Arrwwid number is the smallest number a such that there is a constant c such that any disk Q that lies entirely in U can be covered by a tiles with total area at most $c \cdot \text{area}(Q)$.*

Thus the Arrwwid number is the maximum number of tiles needed to cover any disk if we require the tiles to be relatively small. We call c the *cover ratio*. The def-

inition is essentially from Asano, Ranjan, Roos, Welzl and Widmayer [1], and we name it *Arrwwid number* in their honour. Considering that moving the disk head once may easily cost as much time as scanning ten thousands of points from disk, it is really important to keep the number of tiles, that is, the Arrwwid number, small. Hence the topic of this study: what recursive tilings have small Arrwwid numbers?

Until now, we only required that the data points within each tile are stored contiguously, but we did not require anything of the order in which the subtiles of any tile are stored with respect to each other. However, it may be possible to improve query efficiency by controlling the order in which tiles are stored. A well-chosen order may have the result that some of the tiles used to cover a query range are stored consecutively on disk, thus eliminating the need to move the disk head when going from one tile to the next. When a recursive tiling is enhanced with a recursive definition of how the subtiles within each tile are ordered relative to each other, the result constitutes the definition of a *recursive scanning order* or a *recursive space-filling curve* (we will use these two terms interchangeably; the full paper explains the subtle differences [2]). The Arrwwid number of a space-filling curve is defined exactly as for recursive tilings, only replacing "tiles" by "curve fragments" (sets of consecutive tiles).

In the above definitions we could exchange disks for squares: this would only affect the cover ratios c but not the Arrwwid numbers a. The definitions naturally extend to higher dimensions, replacing disks by balls, area by volume, and squares by (hyper)cubes.

Previously Asano et al. studied scanning orders based on a recursive tiling with four squares per square [1]. The Arrwwid number of such a tiling is four, and Asano et al. proved that no ordering scheme of this tiling has Arrwwid number less than three. They presented the AR^2W^2 scanning order, which has Arrwwid number three. Other authors considered other ways of assessing how well space-filling curves succeed in keeping the number of fragments needed to cover a query range low (references in full paper [2]).

In this paper, we extend the scope of our knowledge on Arrwwid numbers to different tilings (not necessarily with four squares per square) and to higher dimensions. The results are the following: in two dimensions, no recursive tiling and no recursive scanning order has Arrwwid number less than three if the tiles are simply connected regions in the plane. The paper presents recursive tilings with Arrwwid number three (regardless of the order), and an alternative square-based scanning order with Arrwwid number three but without the diagonal connections that are suspected to harm the performance of the AR^2W^2 curve.

In d dimensions, no recursive tiling has Arrwwid number less than $d + 1$. We prove that in three dimensions, putting the tiles in a certain order will not enable us to get below this bound if the tiles are convex polyhedra. There are recursive tilings and scanning orders with fractal-shaped tiles that have Arrwwid number $d + 1$, and tilings and scanning orders with rectangular blocks that have Arrwwid number $\frac{3}{4} \cdot 2^d$. Any regular (hyper)cube-based tiling has Arrwwid number 2^d, and any scanning order based on it has Arrwwid number at least $2^d - 1$.

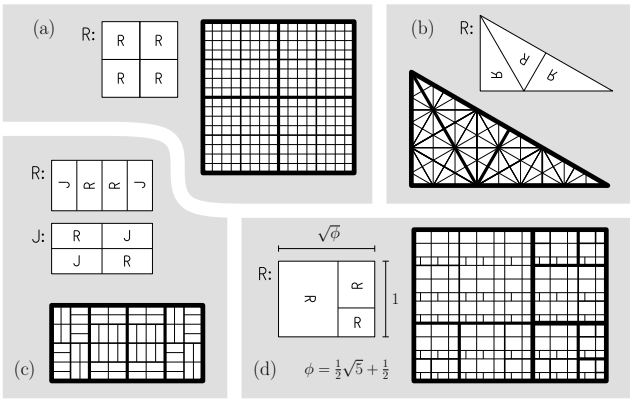


Figure 2: (a) A simple, uniform, regular recursive tiling. (b) A simple, uniform recursive tiling. (c) Uniform but not simple. (d) Simple but not uniform.

Omitted proofs can be found in the full paper [2].

2 Two-dimensional tilings

A recursive tiling of a region U (called the *unit tile*) in the plane is defined by a finite set of recursive tiling rules. Each rule specifies (i) the shape of a finite region to be tiled; (ii) how this region is tiled with a fixed number of tiles; (iii) which rules should be applied to subdivide each of these tiles recursively. Fig. 2 shows some examples. Each rule is identified by a letter, and depicted by drawing the shape of its region, the tiles, and within the tiles, the letters of the rules to be applied to them; each letter is rotated and/or mirrored to reflect the transformations that should be applied to the subtiles of the tile. Next to each set of rules we see the tiling that is produced after expanding the recursion down to tiles of a few millimeters.

Simple tilings are recursive tilings that use only one rule. We define *uniform* tilings as recursive tilings in which all tiles have the same shape, and each rule subdivides such a shape into an equal number of tiles of equal size. The *size* of a uniform tiling is the number of tiles in each rule. By *square* or *rectangular* tilings we mean tilings whose tiles are square or rectangular. *Regular* recursive tilings are those that form a fully regular grid when the recursion is expanded to any fixed depth (Fig. 2(a)). Given any tiling, the *degree* of a point $p \in U$ is the maximum number of interior-disjoint tiles meeting in p. The *vertex degree* of a tiling is the maximum degree of p over all points $p \in U$.

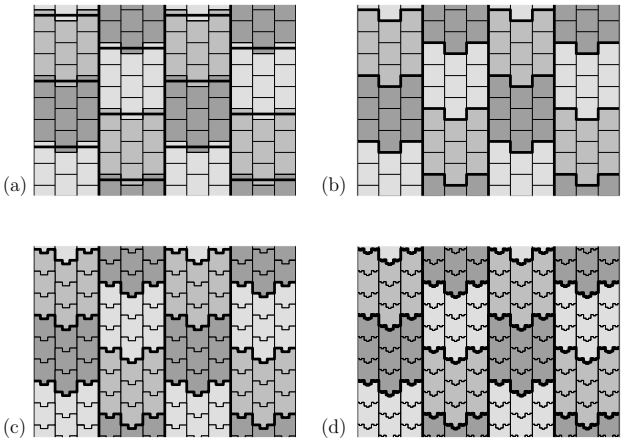


Figure 3: How to turn a "shifted" grid of squares into a recursive tiling with Arrwwid number three.

It is easy to show that the Arrwwid number of a recursive tiling cannot be lower than its vertex degree, and therefore each two-dimensional recursive tiling has Arrwwid number at least three.

To find a tiling that actually has Arrwwid number three, we need to find a tiling with vertex degree at most three. We can modify a grid of squares such that it has vertex degree three: just shift each second column up by half a square's height, like the large squares in Fig. 3(a). However, such a tiling is not recursive: a square in such a grid cannot be subdivided into smaller squares arranged in a similar shifted grid. A recursive tiling can be obtained by recursively approximating larger squares by smaller squares; this procedure will turn the boundaries of the squares into fractals. To do so, we start with a *coarse tiling* and a *fine tiling* as in Fig. 3(a). The fine tiling is the same as the coarse tiling, scaled by a factor 1/3, and aligned with the coarse tiling such that the fine tiling looks the same around each large tile. We assign each small tile to the large tile with which it has the largest overlap. All large tiles of the coarse grid are now replaced by the union of the small tiles assigned to them (Fig. 3(b)). We now replace the small tiles by scaled copies of the large tiles (which are no longer squares), and again replace the large tiles by the unions of the small tiles assigned to them (Fig. 3(c)). When we repeat this process ad infinitum, the boundaries of the tiles converge to fractal shapes such that each large tile is tiled by nine scaled-down copies of itself (Fig. 3(d)). The resulting recursive tiling has Arrwwid number three.

The full paper contains more examples of this technique along with proofs of the Arrwwid numbers [2].

The smallest uniform *rectangular* tiling with Arrwwid number three is the Daun tiling (Fig. 4). The full paper explains how this tiling was found and proves that no four tiles ever meet in one point on any level of recursion [2].

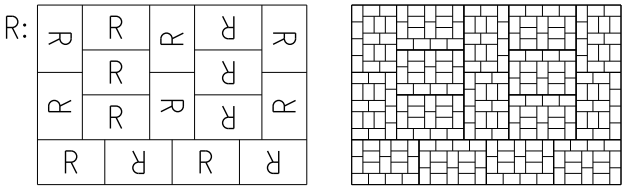


Figure 4: The Daun tiling, with level-two expansion.

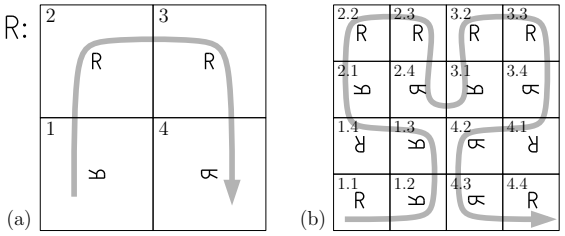


Figure 5: (a) Definition of the Hilbert scanning order. (b) Definition expanded by one level.

3 Two-dimensional space-filling curves

A *recursive scanning order* is a recursive tiling in which the rules also specify an order of the tiles. Fig. 5 shows an example. When defining a scanning order, the recursion within a tile can be rotated and mirrored as with recursive tilings. In addition a recursive rule may be applied with reversed order; we indicate this by a horizontal stroke above the letter that identifies the rule (Fig. 6). As explained in the full paper [2], *scanning orders* are so closely related to *space-filling curves* that for the purposes of this paper, we use these terms interchangeably.

Given any scanning order of a unit tile U, we define the *prefix region* $\mathrm{pre}(x)$ as the region within U that has total area $x \cdot \mathrm{area}(U)$ and comes first in the scanning order; it can be constructed by subdividing U recursively to a sufficiently fine level and collecting tiles in scanning order until tiles with a total area of $x \cdot \mathrm{area}(U)$ have been collected. Let the *fragment* $U[x, y]$ be $\mathrm{pre}(y) \setminus \mathrm{pre}(x)$. Two fragments $U[x, y_1]$ and $U[y_2, z]$ are consecutive if $y_1 = y_2$, and they *connect* in a point p if $U[x, z]$ shrinks to p when x approaches y_1 from below and z approaches y_2 from above.

The Arrwwid number of a curve or scanning order is the smallest number a such that there is a constant c such that any disk Q that lies entirely in U can be covered by a fragments with total area at most $c \cdot \mathrm{area}(Q)$. Since every tile in the underlying recursive tiling constitutes a fragment by itself, the Arrwwid number of a curve is never more than the Arrwwid number of the underlying tiling. However, the Arrwwid number of a curve may be *less* than the Arrwwid number a of the underlying tiling. Asano et al. define the AR^2W^2-curve [1] and prove that it has Arrwwid number three. The *Kochel curve* (Fig. 6) is a new and simpler curve, designed and proven to have Arrwwid number three.

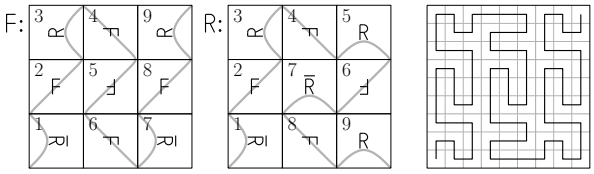


Figure 6: Definition of the Kochel scanning order, with level-two expansion (starting from rule F).

The Kochel curve has the favourable [2] property that consecutive tiles in the order always share an edge.

Theorem 1 *Any space-filling curve based on a recursive tiling with tiles that are topologically equivalent to disks has Arrwwid number at least three.*

Proof. (sketch) Consider any subdivision of a space-filling curve, filling a unit tile U, into a set of fragments F such that each fragment is topologically equivalent to a disk. The boundaries of the fragments in F form a plane graph $\mathcal{G}(V, E)$. By tiles(v) we denote the number of tiles that are incident on the vertex v of $\mathcal{G}$. By conns(v) we denote the number of pairs of consecutive fragments of F that connect to each other in v. We can argue that if the space-filling curve has Arrwwid number less than three, we must have $\text{tiles}(v) - \text{conns}(v) \leq 2$ for every vertex of $\mathcal{G}$ (ignoring effects on the boundary of the outer face). Summing over all vertices this gives (i): $\sum \text{tiles}(v) - |F| \leq 2|V|$. Since tiles($v$) equals the number of edges incident on v we have $\sum \text{tiles}(v) = 2|E|$ and Euler's formula gives (ii): $\frac{1}{2}\sum \text{tiles}(v) - |F| = |V|$ (ignoring additive constants). Subtracting (ii) from (i) twice we get $|F| \leq 0$: a contradiction (the full paper explains how to handle the ignored constants). So the space-filling curve cannot have an Arrwwid number less than three. □

4 Three and more dimensions

Many of the results and techniques presented above generalise to higher dimensions.

Any d-dimensional recursive tiling has Arrwwid number at least four. Similar to the "recursified" shifted-squares tiling in Fig. 3, we can construct a recursified shifted-hypercubes tiling with Arrwwid number $d + 1$. The Daun tiling of Fig. 4 can be used as the basis for d-dimensional tilings and scanning orders with rectangular blocks that have Arrwwid number $\frac{3}{4} \cdot 2^d$. Unfortunately, in three or more dimensions no rectangular tiling with lower Arrwwid number was found, but it might exist. Any uniform (hyper-)cube tiling in d dimensions has Arrwwid number 2^d.

The proof technique of Theorem 1 can be adapted to obtain several results on three-dimensional space-filling curves. The main difficulty in obtaining these results is that Euler's formula cannot be used to derive a contradiction: one needs another way of establishing a relation between $\sum \text{tiles}(v)$, $|F|$, and $|V|$ in a recursive tiling. Using properties of the specific tiling we can prove that any uniform (hyper-)cube scanning order has Arrwwid number at least $2^d - 1$. With an excursion into spherical geometry one can prove that any three-dimensional scanning order based on convex tiles has Arrwwid number at least four.

5 Conclusions

This paper shows that in two dimensions the lowest possible Arrwwid number is three; it is achieved by certain uniform square space-filling curves and by a certain rectangular tiling. Most widely known curves have Arrwwid number four, so the difference is small. However, in higher dimensions the stakes are much higher: no uniform cube space-filling curve can have an Arrwwid number as low as the best known rectangular space-filling curve, and the best known rectangular space-filling curve does not have an Arrwwid number as low as the best known space-filling curve on a fractal tiling. The gap in Arrwwid number increases exponentially with the dimension. However, there may also be an exponentially increasing gap in cover ratio, in favour of the regular hypercube tilings. It is therefore unclear whether the research described in this paper led to anything that might be useful in practice; I hope that this work at least opened some new perspectives on how recursive tilings and space-filling curves can be constructed.

Acknowledgements. I thank Riko Jacob for inviting me for a visit to TU Munich, during which discussion with Tobias Scholl about the merits of the AR^2W^2 curve provided the motivation to start this research. I thank Elena Mumford for many inspiring discussions, and in particular for her contribution to the results on scanning orders with convex tiles in three dimensions.

References

[1] T. Asano, D. Ranjan, T. Roos, E. Welzl, P. Widmayer: Space-Filling Curves and Their Use in the Design of Geometric Data Structures. *Theoretical Computer Science* 181(1):3–15, 1997.

[2] H. Haverkort: *Recursive tilings and space-filling curves with little fragmentation.* Manuscript, arXiv: 1002.1843 [cs.CG], 2010.

[3] T. Scholl, B. Bauer, B. Gufler, R. Kuntschke, A. Reiser, and A. Kemper. Scalable Community-Driven Data Sharing in e-Science Grids. *Future Generation Computer Systems* 25(3):290–300, 2009.

[4] D. Voorhies. Space-filling curves and a measure of coherence. In: J. Arvo (ed.), *Graphics Gems II*, p26–30, Academic Press, 1991.

Straight Skeletons and their Relation to Triangulations*

Stefan Huber† Martin Held†

Abstract

We study straight skeletons of polygons and investigate the dependence of the number of flip events of the classical wavefront propagation by Aichholzer and Aurenhammer on the underlying triangulation. We show that their standard algorithm, applied to a polygon with n vertices, has to cope with at least $\Omega(n^2)$ flip events. In particular, $\Omega(n)$ diagonals of a triangulation may reappear $\Omega(n)$ times each. Still, by allowing a linear number of Steiner points in the triangulation we can avoid flip events completely. As an application of this result we explain how the straight skeleton of a simple polygon with n vertices can be computed in time $O(n^2 \log n)$ by a wavefront-based algorithm that matches the simplicity of the algorithm by Aichholzer and Aurenhammer.

1 Introduction

Since the introduction of straight skeletons by Aichholzer et al. [2] many questions related to straight skeletons have remained unanswered. In particular, there is a significant gap between the known lower and upper bounds for computing straight skeletons: While the best lower bound for the computation of the straight skeleton of a simple polygon with n vertices is $\Omega(n \log n)$, the fastest known algorithms by Eppstein and Erickson [4] and Cheng and Vigneron [3] provide a worst-case runtime of $O(n^{17/11+\epsilon})$ and an expected runtime of $O(n^{3/2} \log n)$, respectively. Both algorithms seem very difficult to implement and, in fact, no implementation is known.

In terms of implementability an approach based on triangulations by Aichholzer and Aurenhammer [1] looks much more promising. They take a triangulation of the input polygon[1] and simulate a wavefront propagation by bookkeeping topological changes of the underlying moving triangulation. Edge and split events of the straight skeleton correspond to topological changes in the triangulation. However, additional topological changes of the triangulation — so-called flip events — appear when a vertex crosses a triangulation diagonal. The handling of the $O(n)$ split and edge events costs $O(n^2 \log n)$ time, while a single flip event can be processed in $O(\log n)$. It is widely believed (but yet unproven) that the number of flip events is bounded by $O(n^2)$, which would yield an overall $O(n^2 \log n)$ bound for their algorithm.

In Sec. 2 and 3 of this paper we present results related to the unproven $O(n^2)$ bound on the number of flip events. Subsequently, in Sec. 4 we use Steiner points to obtain triangulations of polygons that are free of flip events. As an application of this result we explain in Sec. 5 how the straight skeleton of a simple polygon with n vertices can be computed in time $O(n^2 \log n)$ by a wavefront-based algorithm that matches the simplicity of the algorithm by Aichholzer and Aurenhammer [1].

2 How often can diagonals reappear?

Currently, the best known upper bound for the number of flip events is $O(n^3)$. This follows from the fact that three points that move with constant speed along lines in the plane are either never, once, twice or always collinear. (The determinant of the matrix whose columns consist of the homogeneous coordinates of the points is a quadratic expression in time, and a root indicates collinearity.) Consequently, a single diagonal of a triangulation can be crossed at most $2(n-2)$ times by other vertices, and since there are at most $O(n^2)$ possible diagonals, an upper bound of $O(n^3)$ follows. Of course, not every collinearity of three vertices corresponds to a flip event.

This proof links the number of flip events with the number of reappearances of triangulation diagonals. The following lemma highlights that one cannot establish an $O(n^2)$ bound on the number of flip events during the wavefront propagation by attempting to show that the number of reappearances of every single diagonal of an arbitrary triangulation is in $O(1)$.

Lemma 1 *There exists a sequence of polygons P_n, with $\Theta(n)$ vertices, and corresponding triangulations T_n such that $\Omega(n)$ diagonals of T_n reappear $\Omega(n)$ times during the wavefront propagation applied to P_n.*

Proof. Roughly, we come up with an appropriate geometric configuration of moving vertices that realizes a sequence of topological transitions such that diagonals reappear as often as claimed. Our constructive proof is split into three parts. First, we construct a polygon P and a triangulation T where one diagonal

*Work supported by Austrian FWF Grant L367-N15.

†Universität Salzburg, FB Computerwissenschaften, A-5020 Salzburg, Austria, {shuber,held}@cosy.sbg.ac.at

[1] Actually, their algorithm can handle planar straight line graphs as input.

reappears twice. Then we extend this construction such that a single diagonal appears $\Omega(n)$ times. In the third part we extend it such that $\Omega(n)$ diagonals each reappear $\Omega(n)$ times. (Due to the lack of space we omit details of Part 3 of the proof, though.) In the sequel we denote by $V(t)$ the position of the vertex V at time t and by AB the supporting line of the vertices A and B.

Part 1: We start with showing how to make a diagonal AB reappear twice during the movement of six vertices A, B, S_1, S_2, N_1, N_2 as induced by the wavefront propagation. The wavefront propagation starts at time $-\varepsilon$, for a sufficiently small $\varepsilon > 0$, and topological transitions will occur at times $0, 1, 2, 3, 4$ and 5. The initial positions of the six vertices (discussed in detail below) are shown in the top part of Fig. 1, while the upper-left triangulation in the lower part of Fig. 1 shows the initial triangulation of the vertices. The other triangulations show topological transitions needed to recreate AB. Roughly, AB will disappear because the vertex S_1 crosses it. Then S_1 falls back behind AB again. After the recreation of that diagonal $S_2(t)$ will cross it, causing it to disappear again. The corresponding topological transitions are illustrated in the bottom of Fig. 1.

We now discuss details of the geometric configuration of the six vertices. Let the two vertices A, B both

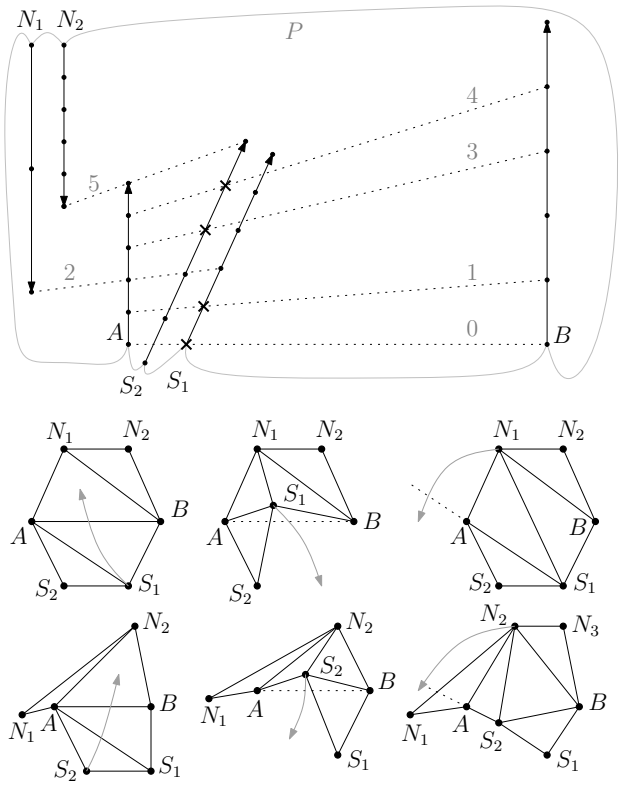


Figure 1: Part 1. Top: geometric configuration. Bottom: Topological transitions at the six points in time depicted as grey numbers in the top sub-figure.

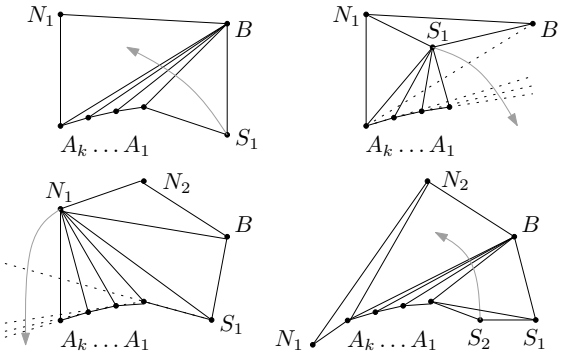


Figure 2: Part 3. Topological transitions of the first reappearance cycle.

move northwards (parallel to the positive y-axis), with B moving strictly faster than A. We want the vertex S_1 to move northeastwards and to cross AB at time 0 and to fall back behind AB a bit later, say at time 1. We achieve this by demanding $S_1(t)$ to lie on $A(t)B(t)$ for $t \in \{0, 1\}$, cf. Fig. 1. Once $S_1(0)$ and $S_1(1)$ are fixed, the movement of S_1 has been specified completely. Since the area of the triangle $\Delta(A(t)B(t)S_1(t))$ is a quadratic expression in t and since $A(t)B(t)S_1(t)$ are oriented clockwise (CW) for some $t > 1$ we conclude that $A(t)B(t)S_1(t)$ is oriented CW for $t \notin [0, 1]$ and counter-clockwise (CCW) for $t \in (0, 1)$. The recreation of the diagonal AB is achieved by letting a vertex N_1 move southwards to the left of A such that it crosses $A(t)S_1(t)$ at, say, time 2, cf. Fig. 1. Now we place a vertex S_2 that moves parallel to S_1 between A and S_1 such that $S_2(t) \in A(t)B(t)$ for $t \in \{3, 4\}$. Again it holds that $A(t)B(t)S_2(t)$ is oriented CW if $t \notin [3, 4]$ and CCW if $t \in (3, 4)$. Thus, the diagonal AB disappears at time 3 and S_2 falls back behind AB again at time 4. Similar to N_1 we place a vertex N_2 that recreates the diagonal AB by requesting $N_2(5) \in A(5)S_2(5)$, cf. Fig. 1. Note that by increasing the inclination of the supporting rays of the vertices S_1, S_2 we can force the start positions of $S_1(-\varepsilon)$ and $S_2(-\varepsilon)$ to get arbitrarily close to the line $A(-\varepsilon)B(-\varepsilon)$. By doing so we can guarantee that a polygon P exists such that the given geometric configuration is achieved. (It is shown as a curve depicted in light grey in Fig. 1.)

Part 2: We add vertices $S_3, \dots, S_m$ from right to left between S_2 and A, according to the construction scheme of part 1. That means that S_{i+1} gets collinear with A and B at time $3i$ and $3i + 1$. Analogously we add vertices $N_3, \dots, N_m$ from left to right next to N_2 such that N_{i+1} crosses AS_{i+1} at time $3i + 2$. Again, note that if the vertices $S_1, \dots, S_m$ are moving nearly vertically then the start positions of $S_1, \dots S_m$ get arbitrarily close to the line AB.

Part 3: The basic idea is to arrange copies $A_2, \dots, A_k$ of A_1 along a reflex chain of the polygon

such that the positions of $A_2(t), \ldots, A_k(t)$ remain sufficiently close to the line $A_1(t)B(t)$ for the entire time span of the wavefront propagation. By doing so one can achieve the topological transitions as sketched by the first reappearance cycle in Fig. 2. □

3 Can we always find good triangulations?

As a byproduct of the previous lemma we obtain polygons and triangulations that lead to $\Omega(n^2)$ flip events. However, this result hinges on meticulously chosen triangulations which are bad in the sense that $\Omega(n)$ diagonals each reappear $\Omega(n)$ times. For example, in the construction scheme above, we could have initially put diagonals between N_m and $A_1, \ldots, A_k$, thus avoiding the reappearance cycles of the diagonals. Can we always find, for every given polygon P, a good triangulation T such that the number of flip events is low, say $o(n^2)$ or even $O(n)$? The following lemma provides a negative answer to this question.

Lemma 2 *There exist polygons with n vertices for which every triangulation leads to $\Omega(n^2)$ flip events.*

Proof. We consider the polygon shown in Fig. 3. The vertices $A, E_1, \ldots, E_k, B$ lie on a reflex chain such that W is the only vertex initially seen by an E_i. Hence every triangulation contains the diagonals WE_i, for $1 \leq i \leq k$. These diagonals are flipped by the notch vertices $N_1, \ldots, N_m$ which move southwards. We ensure that $N_1, \ldots, N_m$ are fast enough that they cross those diagonals before any edge or split event occurs where a vertex $E_1, \ldots, E_k$ is involved. Furthermore, we request that for each $i \in \{1, \ldots, m-1\}$ the vertex N_{i+1} does not cross the supporting line of AE_1 before N_i induced $\Omega(k)$ flip events. The claim follows by choosing m, k roughly equal to $n/2$. □

One might feel that retriangulating at specific favorable moments could reduce the complexity. However, the polygon above seems to illustrate a counter example: Note that a retriangulation at any point in time does not save more than $O(n)$ flip events. Hence, to gain a runtime advantage one would have to perform $\Omega(n)$ retriangulations. Assume that we perform a retriangulation saving $O(n)$ flip events. Handling the flip events directly would have cost $O(n \log n)$ time. A single retriangulation invalidates $O(n)$ entries in the priority queue. This means that one would gain a runtime advantage only if $O(n)$ entries in the priority queue could be reset in $o(n \log n)$ time.

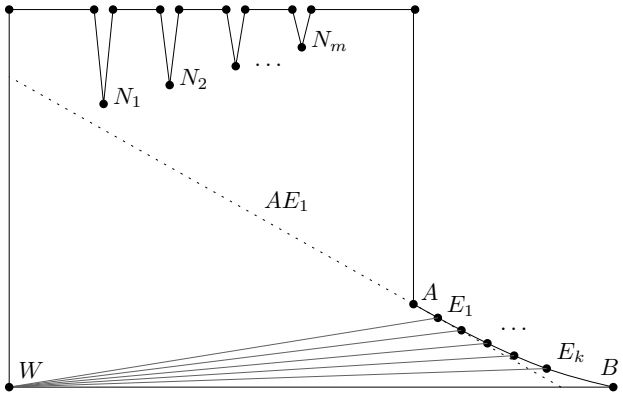


Figure 3: A polygon where every possible triangulation leads to $\Omega(n^2)$ flip events.

4 Steiner triangulations without flip events

In this section we investigate whether Steiner points can be used to obtain triangulations with a low number of flip events or, more precisely, with no flip events at all.

Lemma 3 *Every simple polygon P with n vertices admits a triangulation that employs at most $n-2$ Steiner points and which is free of flip events.*

Proof. We consider the straight skeleton S of P and add its at most $n-2$ inner nodes as Steiner points and its arcs as initial diagonals of the triangulation. It now remains to triangulate the faces of S.

Let the face f of S be induced by the segment s of P. Let p, q denote the endpoints of s. Note that f (as a polygon) is monotone with respect to s and that reflex vertices of f are only present in the corresponding monotone chain that does not contain s. Recall that split event nodes correspond to reflex vertices of f. If f contains no reflex vertices then we triangulate f arbitrarily. Otherwise we choose that reflex vertex v which has minimum orthogonal distance to s, and insert two diagonals vp and vq. Note that f contains the diagonals completely: otherwise we would have missed a split event node of f having smaller orthogonal distance to s. Then f is decomposed by the triangle pqv into two remaining parts A and B, where A contains the diagonal pv and B contains the diagonal qv, see Fig. 4. We proceed recursively within A and B. That is, if A has no reflex vertex then we triangulate arbitrarily. Otherwise we find a reflex vertex v' with minimum orthogonal distance to s and insert two diagonals $v'p$ and $v'v$. Then A is split by the triangle pvv' into two parts, and so on.

During the wave propagation the vertices of P move on the straight skeleton and hence do not cross any diagonal at any time. For every face f the corresponding segment s is moving in a self-parallel manner inwards and may be split when reaching reflex vertices of f. In contrast to that the Steiner points stay in place and wait until the corresponding segments of P reach them. The triangles in a face collapse only when s reaches a node of f and hence an edge or a split event occurs. However, no diagonal crosses a Steiner point such that a flip event needs to be handled. □

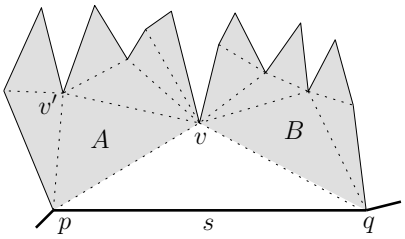


Figure 4: The triangulation scheme for a straight skeleton face of a segment s.

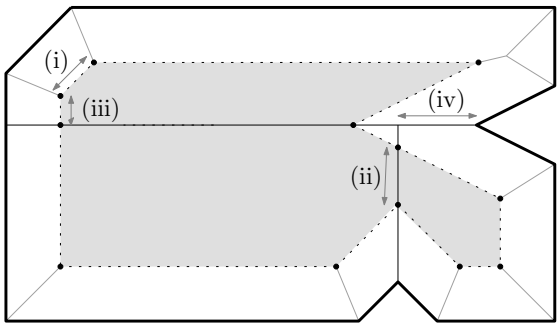


Figure 5: Different types of events for the motorcycle graph based straight skeleton algorithm.

5 Motorcycle graph based straight skeletons

Obviously, the proof of the previous lemma does not result in a new algorithm for computing straight skeletons. However, we recall again that convex vertices of P do not cause flip events. On the other hand, in the former construction, reflex vertices of P are barred from causing flip events since those vertices move along triangulation diagonals which are part of the straight skeleton. Fortunately, this property also holds if we replace the straight skeleton by the motorcycle graph M induced by the moving reflex vertices.

For the sake of simplicity we also adopt the assumption of Cheng and Vigneron [3] that no split event of higher degree exists, i.e., that no two or more reflex vertices meet simultaneously in a common point. Under this assumption Cheng and Vigneron [3] showed that a reflex arc of the straight skeleton is not longer than the trace of the corresponding motorcycle. (We assume that motorcycles crash at the boundary of the initial polygon.) Note that M always decomposes P into convex parts during the entire shrinking process.

This suggests that we can obtain an algorithm for computing straight skeletons by by employing the motorcycle graph during the wavefront propagation process. In contrast to the former sections we do not consider a triangulation but maintain the intersection points of M with the wave front and call them Steiner vertices. The following types of events occur, see Fig. 5: (i) edge event: two neighboring convex vertices in a convex part of P meet; (ii) split event: a reflex vertex meets its corresponding Steiner vertex; (iii) switch event: a convex vertex meets a Steiner event and hence the convex vertex migrates to a different convex part of P; (iv) start event: a reflex vertex or a (moving) Steiner vertex meets a (resting) Steiner vertex, which is the endpoint of a different trace and has to start moving. Note that only neighboring[2] vertices meet in the propagation process since the motorcycle graph decomposes the shrinking polygon P at any time into convex parts.

Hence, it suffices to check only for collisions among vertices which are neighbors. We put corresponding events into a priority queue and process them in chronological order. In the worst case there are up to $O(n^2)$ switch events, but all other events occur $\Theta(n)$ times. Every event can be handled in $O(\log n)$ time, since only a constant number of neighbors of the two vertices affected have to be modified in their propagation speed. Hence, the algorithm runs in $O(n^2 \log n)$ time in the worst case but still enjoys a simplicity that is comparable to the triangulation-based algorithm by Aichholzer and Aurenhammer [1].

We note that the $O(n^2)$ bound on the number of switch events seems overly pessimistic, and there is reason to assume that for many data sets, in particular for those from real-world applications, there might be only $O(n)$ switch events, resulting in $O(n \log n)$ runtime in practice (if the motorcycle graph is already given). Furthermore, our algorithm need not be restricted the interior of one polygon but could also be extended to planar straight-line graphs.

Acknowledgement We would like to thank Gerhard Mitterlechner for valuable discussions.

References

[1] O. Aichholzer and F. Aurenhammer. Straight Skeletons for General Polygonal Figures in the Plane. In A. Samoilenko, editor, *Voronoi's Impact on Modern Science, Book 2*, pages 7–21. Institute of Mathematics of the National Academy of Sciences of Ukraine, Kiev, Ukraine, 1998.

[2] O. Aichholzer, F. Aurenhammer, D. Alberts, and B. Gärtner. A novel type of skeleton for polygons. *Journal of Universal Computer Science*, 1(12):752–761, 1995.

[3] S.-W. Cheng and A. Vigneron. Motorcycle Graphs and Straight Skeletons. In *Proc. 13th ACM-SIAM Sympos. Discrete Algorithms*, pages 156–165, San Francisco, CA, USA, 2002.

[4] D. Eppstein and J. Erickson. Raising Roofs, Crashing Cycles, and Playing Pool: Applications of a Data Structure for Finding Pairwise Interactions. *Discrete Comput. Geom.*, 22(4):569–592, 1999.

[2] On the wave front or on the motorcycle graph.

Convex Hull Of Imprecise Points Modeled By Segments In The Plane

Ahmad Javad* Ali Mohades* Mansoor Davoodi* Farnaz Sheikhi*

Abstract

Let S be a set of n imprecise points in the plane that each imprecise point is modeled by a segment. In this paper, we study the problem of finding a minimum perimeter convex hull of S that segments are either inside this convex hull or intersected by it. We present the first polynomial time algorithm to solve this problem in $O(n^2 \log n)$ time where the segments are disjoint.

1 Introduction

Computational geometry is a vast area of research that mostly deals with designing algorithms that work with exact input data, but in real world problems, due to devices with limited accuracy, input data might be imprecise. Therefore, a new class of problems focuses on designing algorithms which are able to work with imprecise data. Imprecise data can be modeled by a region they lie on.

Suppose that S is a set of imprecise points where each of its points is modeled by a segment in the plane. We want to find the minimum perimeter convex polygon that has no segments of S outside. We assume there does not exist a line which intersects all segments of S. Goodrich and Snoeyink presented an algorithm that finds a convex polygon whose boundary stabs a set of parallel line segments, in $O(n \log n)$ time [1]. Meijer and Rappaport allowed the interior and the boundary of the polygon to stab the set S of parallel line segments, and found a stabbing polygon of the smallest perimeter, called a $minimumstabbingpolygon$ of S, in $O(nlogn)$ time [4]. Rappaport proposed an algorithm for the problem of computing convex hull of a set of disjoint segments, which is called $minimumpolygontransversals$, in $O(3^k n \log n)$ time [5]. Hassanzadeh showed that algorithm could not work correctly in some cases. So, he corrected it, resulting in an $O(4^k n \log n)$ time algorithm, and presented several approximation algorithms to solve that problem as well [2]. Löffler and van Kreveld studied minimum/maximum perimeter/area convex hull of imprecise points where each imprecise point was modeled by a segment or a square. They also proved the problem of finding maximum area/perimeter convex hull is NP-hard [3].

No polynomial time algorithms are known to solve the problem of finding the minimum perimeter convex hull of a set of segments. In this paper, we present the first polynomial time algorithm which solves this problem in $O(n^2 \log n)$ time.

*Laboratory of Algorithms and Computational Geometry, Department of Mathematics and Computer Science, Amirkabir University of Technology, {ahmadjavas,mohades,mdmonfared,f.sheikhi}@aut.ac.ir

2 Preliminaries

The algorithm that we present to solve the problem proposed, is similar to $QuickHull\ algorithm$ which computes the convex hull of a point set in the plane. Our algorithm is an iterative one that in each iteration computes the convex hull of some special segments of S by using an $unfolding\ method$ [2], and then updates the resultant convex hull regarding the segments that lie outside it.

Before concentrating on details of the algorithm, we present some useful concepts. Let P be a set which includes endpoints of all segments of S, and $CH(P)$ denote its convex hull.

Theorem 1 *Suppose at least one endpoint of each segment of S lies on the boundary of $CH(P)$. A tour MT which visits all segments has the minimum length, if it intersects the segments in their clockwise (or counter clockwise) traversal on the boundary of $CH(P)$.*

Proof. Let $s_1, s_2, , s_n$ be the ordered segments which are visited in clockwise traversal on the boundary of $CH(P)$, and w_i be the intersection point of s_i and MT. Assume MT does not visit segments in their clockwise order, so there exist some segments like s_i that MT visits it after s_j, $(i < j)$. Let MT be $\ldots, w_{i-1}, w_{i+1}, w_{i+2}, \ldots, w_{j-1}, w_j, w_i, w_{j+1} \ldots$ in clockwise order. If $w_i w_j$ intersects $w_{i-1}w_{i+1}$, and $w_{i-1}w_i w_{i+1} w_j$ is a convex quadrilateral, according to the triangle inequality, the tour $\ldots, w_{i-2}, w_{i-1}, w_i, w_{i+1}, \ldots, w_j, w_{j+1}, \ldots$ is a shorter tour, providing a contradiction. Otherwise, if $w_i w_j$ does not intersect $w_{i-1}w_{i+1}$, two cases will be arisen: either $w_{i-1}w_{i+1}$ intersects s_i or does not. If $w_{i-1}w_{i+1}$ intersects s_i, we can obtain a shorter tour by replacing the intersection point with w_i. On the other hand, if $w_{i-1}w_{i+1}$ does not intersect s_i, so s_i certainly lies inside the convex shape (5- or 6-gon) which is constructed with s_{i+1}, s_{i-1} and a part of the boundary

of $CH(P)$. Thus, w_iw_j intersects either s_{i+1} or s_{i-1}. Similarly, in both situations, we can obtain a shorter tour by replacing the intersection point with either w_{i+1} or w_{i-1}, providing a contradiction. □

Lemma 2 *Suppose that at least one point of each segment of S, lies on the boundary of $CH(P)$. The minimum length tour MT which visits all segments of S, is convex.*

Definition 1 *Given a set of ordered segments, a minimum perimeter tour that visits all such segments is denoted by $MTOS$.*

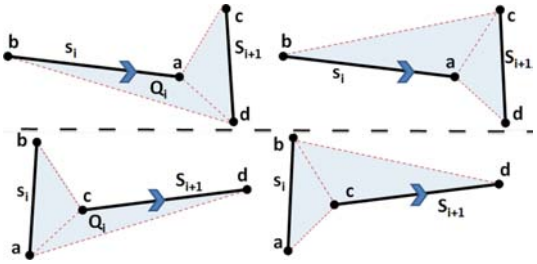


Figure 1: Illustration of two choices that exist to select Q_i.

3 Convex hull of segments algorithm

At first, we compute the convex hull of P and denote it by L_0. Then, by a clockwise traversal on L_0, we find all segments of S which intersect L_0, and insert them into a list called SL_0. Note that if there exist some segments that have more than one intersection point, we insert them only once. We set SL_0 to CS_1, and compute the convex hull of the ordered set CS_1 by using the following method. In this step, we prune segments which are not important in the final solution from CS_1. Based on Theorem 1, MT visits every segment s_i between s_{i-1} and s_{i+1}. Let a_i and b_i be the endpoints of s_i which a_i lies on the convex hull. We will remove s_i from CS_1 if it intersects with $b_{i-1}b_{i+1}$, $a_{i-1}b_{i+1}$ and $b_{i-1}a_{i+1}$. After each removal, we update CS_1 and repeat this process until there are no segments left to remove.

With respect to Theorem 1, for the ordered set CS_1, $MTOS$ should visit s_i before s_{i+1}. Let a and b be endpoints of s_i, and c and d be endpoints of s_{i+1}. Obviously, $MTOS$ crosses either the quadrilateral $abcd$ or $abdc$. Let Q_i be the quadrilateral which $MTOS$ crosses. If either $abcd$ or $abdc$ is a convex quadrilateral, we will select the convex one as Q_i. Otherwise, if both $abcd$ and $abdc$ are non-convex, we will select Q_i as follows. Suppose that the extension of s_i intersects s_{i+1}. Let b be the nearest endpoint of s_i to s_{i+1}. We select the quadrilateral that contains the triangle that lies on the right of the directed segment ab. On the other hand, if the extension of s_{i+1} intersects s_i, and c is its nearest endpoint to s_i, we select the quadrilateral that contains the triangle which lies on the right of the directed segment cd (see Fig. 1).

Each two consecutive quadrilaterals Q_i and Q_{i+1}, which are constructed by the way mentioned , share a segment s_{i+1}. If we start from a segment of CS_1 and cross its related quadrilateral to get the next quadrilateral, we can construct a tour which completely lies inside the union of all quadrilaterals, and also visits all segments of CS_1. This tour is a convex hull for the ordered set CS_1. The minimum tour, which is denoted by CHS_1, could be constructed in linear time by *unfolding method* [2]. See Fig. 2. The process mentioned in this section that computes $MTOS$ is called MTA.

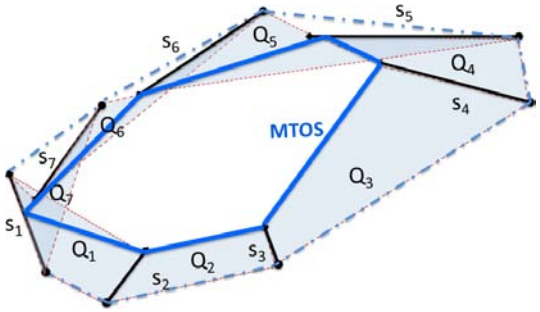


Figure 2: Illustration of consecutive quadrilaterals constructed and $MTOS$ corresponding to CS_1.

Lemma 3 *For a set of n ordered segments, MTA could compute $MTOS$ in $O(n)$ time.*

Suppose that CS_i and CHS_i have been computed up to the i-th iteration. Segments which are located inside CHS_i will be removed form S. Considering segments which are located outside of CHS_i, the convex hull of their endpoints will be computed, and denoted by L_i. With a clockwise traverse on the boundary of L_i, segments which fall outside of L_i and have at least one point on the boundary of L_i, will be added to SL_i in order; note that each segment will be added to SL_i only once. Regarding the location of L_i and CHS_i, CHS_{i+1} could be computed in three different cases as follows.

Case 1: if CHS_i is inside L_i, we will compute intersection points of segments of CS_i with L_i, and with a clockwise traverse on the boundary of L_i, we will add segments of CS_i among those of SL_i, and CS_{i+1} will be computed. Further, convex hull of segments of CS_{i+1} will be computed by using MTA, and denoted by CHS_{i+1}.

Case 2: if CHS_i is outside of L_i (see Fig. 3), there might exist some segments of CS_i which intersect with

L_i at two points (s_1 and s_2, in Fig. 3, are such segments). We will take their farthest intersection points from CHS_i as the place they intersect with L_i (points p_1 and p_2 in Fig. 3). Each segment of this kind will be added to SL_i between the segments of this set which their endpoints locate exactly before and after the specified intersection point of such segment with L_i (in Fig. 3, s_1 (resp. s_2) will be added between s_6 and s_{15} (resp. s_9 and s_8)). Let $s_j, \ldots, s_{j+m}$ be the segments of CS_i which intersect with L_i at two points. The segment of CS_i that is located before s_j (resp. after s_{j+m}) and does not intersect with L_i, is denoted by s_a (resp. s_b), (in Fig. 3, $s_a = s_3$ and $s_b = s_4$). It might be possible that $s_a = s_b$, but the fact that CHS_i is outside of L_i ensures that at least one of these segments exists.

By using an angular sweep line which is along s_a (resp s_b), and rotates around the intersection point of s_a (resp. s_b) and CHS_i, the plane is swept in counter-clockwise (resp. clockwise) direction; the first vertex of L_i that the sweep line intersects with it, is denoted by v_r (resp. v_l), see Fig. 3. The chain of L_i that is located between v_l and v_r in a clockwise traverse, is called *theupperchain*, and denoted by UC. Those segments of CS_i which intersect with L_i at two points, will be removed from CS_i, and the segments of SL_i that at least one of their endpoints is located on UC will be added to CS_i between s_b and s_a in the order that their endpoints are seen in a clockwise traverse on UC, (as an example, in Fig.3; at first, CS_i=$\{s_1, s_2, s_3, s_5, s_4\}$ and after adding the segments of SL_i which one of their endpoints is located on UC, CS_i will be changed into $\{s_4, s_{14}, s_{15}, s_1, s_6, s_7, s_8, s_2, s_9, s_{10}, s_3, s_5\}$). Now, we have an ordered set of segments. By using MTA, we could achieve an optimal convex polygon that intersects the segments of CS_i in order. CHS_{i+1} denotes this polygon, and we set CS_i to CS_{i+1}.

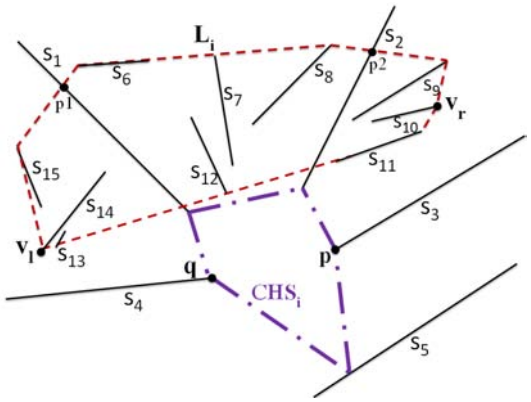


Figure 3: Illustration of case 2.

Theorem 4 *CHS_{i+1} is a convex polygon.*

Proof. Let p (resp. q) be the intersection point of s_a (resp. s_b) and CHS_i. By Theorem 1, we know that segments on UC should be visited in the order they locate on UC, and by Lemma 2, it is clear that the shortest path between p and q which visits those segments, is a convex chain. This chain is denoted by π. π and the chain which exists between p and q in a clockwise traverse on CHS_i, both contain p and q. Therefore, we could construct a polygon T by using them; see Fig. 4. T might have two concave vertices at points p and q. Let q be the concave vertex, and s (resp. r) be its previous (resp. next) vertex in a clockwise traverse on T. Since s_b has been visited before the segments on UC, segment rs intersects s_b. Thus, by substituting rq and qs with rs in T, T still intersects with all segments of CS_{i+1} and its concavity in q is also removed. The same approach could be taken for p, and T becomes a convex polygon which visits all segments of CS_{i+1}, as a result. According to the fact that there exists a convex tour for visiting ordered segments of CS_{i+1}; CHS_{i+1}, which is the output of MTA, will also be convex. □

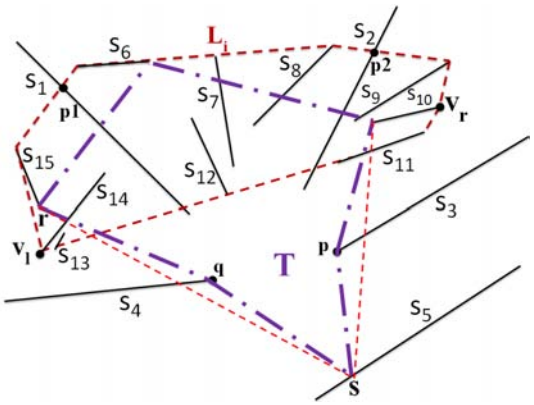


Figure 4: CHS_{i+1} that is computed by MTA is convex.

Case 3: if L_i intersects with CHS_i, then segments which are not completely inside CHS_i will become important in computing CHS_{i+1}; see Fig. 5. In Fig. 5, CS_i=$\{s_1, s_5, s_6, s_2, s_7, s_3, s_8, s_4, s_9\}$) and SL_i=$\{s_{10}, s_{11}, s_{12}, s_{13}, s_{14}, s_{15}\}$. We will find segments of CS_i that intersect with L_i (in Fig. 5, these segments are s_1, s_2 and s_8). Segments of this kind which their intersections with L_i are outside of CHS_i, will be added to SL_i with respect to their intersection points with L_i, and they will be removed from CS_i; in Fig. 5, only s_1 and s_2 are such segments, and after such addition and removal we will have CS_i=$\{s_5, s_6, s_7, s_3, s_8, s_4, s_9\}$ and SL_i=$\{s_{10}, s_1, s_2, s_{11}, s_{12}, s_{13}, s_{14}, s_{15}\}$. We find intersections of L_i and CHS_i. We call each part of L_i that is outside of CHS_i, an exterior chain (denoted by EC). It is clear that all exterior chains are convex

(in Fig. 5, two exterior chains are shown in ovals). Considering each EC, let s_a (resp. s_b) be the segment of CS_i which is visited immediately after (resp. before) the intersection point of EC and CHS_i by a clockwise traverse on the boundary of CHS_i (in Fig. 5, $s_a = s_5$ and $s_b = s_9$ for $EC1$, and $s_a = s_7$ and $s_b = s_6$ for $EC2$). Considering each EC, regarding its s_a and s_b we find a UC. Segments which have at least one of their endpoints on this UC, are added to CS_i between s_a and s_b corresponding to the specified EC. It could be easily proved that segments which are outside of CHS_i and have one of their endpoints on EC, should be visited by CHS_{i+1} between s_a and s_b corresponding to that specific EC (proof is similar to Theorem 1). So, they should be added to CS_i, between s_a and s_b. Adding these segments to CS_i in away that running MTA on CS_i results in a convex polygon, could be done via a similar approach as the one used in case 2; with the only difference that in this case instead of a polygon fallen outside of CHS_i, we have some ECs that segments of them which have one of their endpoints on ECs, should be added to CS_i to achieve CHS_{i+1}. Similar to the proof of Theorem 4, it could be proved that CHS_{i+1}, which is the output of MTA, is also convex in this case, and we set CS_i to CS_{i+1}.

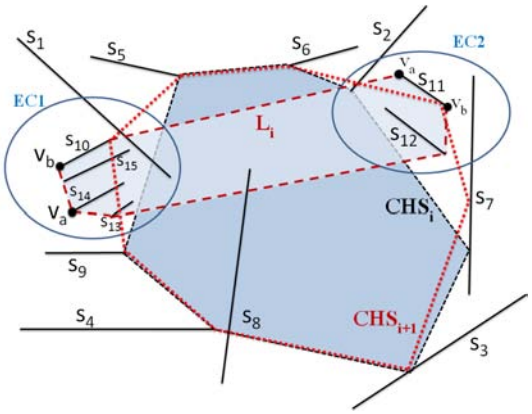


Figure 5: Illustration of case 3.

We will repeat this routine until there are not any segments of S outside of CHS_i at all, and in the end, we will report CHS_i as the final result of the problem.

4 Analysis of the algorithm

In each iteration of the algorithm, L_i should be computed. Finding segments which are outside of CHS_i as well as computing convex hull of their endpoints needs $O(n \log n)$ time in worse case. Regarding the location of L_i and CHS_i, determining the corresponding case could be done in $O(n)$ time. In case 1, According to the fact that L_i is convex and both SL_i and CS_i are ordered, finding intersection of L_i with the segments of CS_i and adding the segments to SL_i could be done in linear time. In case 2, Finding UC and adding the segments which have one of their endpoints on it, takes $O(n)$ time, and finally, in case3, since the total number of segments which have at least one of their endpoints on ECs, is the same as the total number of segments of SL_i which is $O(n)$, and they could be handled in $O(n)$ time in a similar way as case 2. In each iteration $CHS_i + 1$ should be computed by running MTA on CS_i. From the fact that CS_i is always a set of ordered segments, removal of unimportant segments from CSi could be done in $O(n)$ time. Constructing quadrilaterals and utilizing *unfolding method* take linear time [2]. So, MTA runs in $O(n)$ time. Therefore, each iteration of our algorithm takes $O(n \log n)$ time. Since in each iteration just one segments may involve in computing CHS_{i+1} in the worse case, the algorithm runs $O(n)$ times. Thus, the time complexity of the algorithm is $O(n^2 \log n)$ time.

In situations where two consecutive segments of CS_i are collinear, the quadrilateral could not be constructed, but handeling these situations could be done by *unfolding method*, and whenever CHS_i becomes a line segment, it will change into a convex polygon in next iterations. So, these special cases could be handled easily.

5 Conclusion

We have presented the first polynomial time algorithm to compute the convex hull of a set of imprecise points which are modeled by n disjoint segments in the plane. Our proposed algorithm runs in $O(n^2 \log n)$ time. The main idea of the algorithm is to find an order for visiting the segments. We believe that this idea can also be useful for computing the minimum perimeter convex hull of a set of imprecise points which are modeled by polygons instead of segments.

References

[1] M. T. Goodrich, J. S. Snoeyink, *Stabbing parallel segments with a convex polygon.* In Computer vision, Graphics and Image Processing 49, 152170 ,1990.

[2] F. Hassanzadeh. *Minimum Perimeter Convex Hull of a Set of Line Segments: An Approximation.* Master Thesis, Queen's University Kingston, Ontario, Canada, November 2008.

[3] M. Löffler, M. van Kreveld, *Largest and smallest convex hulls for imprecise points.* Algorithmica, 56(2), 235269, 2010.

[4] H. Meijer, D. Rappaport. *Minimum polygon covers of parallel line segments.* Department of Computing and Information Science Technical Report, 90-279, Queens University, 1990.

[5] D. Rappaport, *Minimum polygon transversals of line segments.* Journal of Computational Geometry and Applications, 5, 243256, 1995.

Hiding in the Crowd: Asymptotic Bounds on Blocking Sets

Nataša Jovanović* Jan Korst† Zharko Aleksovski‡ Radivoje Jovanović§

Abstract

We consider the problem of blocking all rays emanating from a unit disk U by a minimum number N_d of unit disks in the two-dimensional space, where each disk has at least a distance d to any other disk. We study the asymptotic behavior of N_d, as d tends to infinity. Using a regular ordering of disks on concentric circular rings we derive upper and lower bounds and prove that $\frac{\pi^2}{16} \leq \frac{N_d}{d^2} \leq \frac{\pi^2}{2}$, as d goes to infinity.

1 Introduction

Let U be a unit disk, i.e. a disk with radius 1, in the two-dimensional space and let $\mathcal{R}$ denote the set of all rays that emanate from U. A ray $r \in \mathcal{R}$ is said to be blocked by a disk δ if r and δ have a non-empty intersection. A set $\mathcal{D}$ of unit disks, with $U \notin \mathcal{D}$, is called a *blocking set* if every ray $r \in \mathcal{R}$ is blocked by a disk in $\mathcal{D}$. In addition, a blocking set $\mathcal{D}$ is called *d-apart* if the distance between each pair of disks in $\mathcal{D} \cup \{U\}$ is at least d, where distances are measured from center to center.

Minimum Cardinality Blocking Set Problem. *Given d, what is the minimum cardinality N_d of a d-apart blocking set?*

More specifically, we are interested in the asymptotic behavior of N_d as d tends to infinity. For reasons of convenience, we focus on the following problem, which is equivalent to the minimum cardinality blocking set problem.

Maximum Distance Blocking Set Problem. *Given N unit disks, what is the maximum distance d for which the disks can form a d-apart blocking set?*

These problems are related to occlusion problems in table-top interaction devices, where multiple sensors, for example, light sensors or cameras, scan the two-dimensional plane just above the table's surface for objects like game pieces or fingers. A disk in that plane is no longer visible if all rays emanating from it are blocked by other disks.

*Eindhoven University of Technology and Philips Research Europe, The Netherlands, n.jovanovic@tue.nl

†Philips Research Europe, The Netherlands, jan.korst@philips.com

‡Philips Research Europe, The Netherlands, zharko.aleksovski@philips.com

§Gimnazija u Lebanu, Serbia

1.1 Our Contributions

In this paper we prove that both upper and lower bounds on the minimum number N_d of disks are quadratic in d, i.e. we prove that $N_d = \Theta(d^2)$. In more detail, we first show that N disks can be positioned such that they form a 2-apart blocking set. The disks of that blocking set are placed on a circle concentric to U with neighboring disks being mutually tangent. We present a simple algorithm of pushing the disks towards the center of U such that the blocking of rays is preserved. The algorithm provides a regular ordering of disks on concentric circular rings such that the disks form a d-apart blocking set, where $d > 2$. Finally, upper and lower bounds are given showing that

$$\frac{\pi^2}{16} \leq \lim_{d\to\infty} \frac{N_d}{d^2} \leq \frac{\pi^2}{2}.$$

1.2 Related Work

Jovanović, Korst and Janssen [6] consider a variant of the above blocking set problem, where they consider blocking all lines intersecting a given unit disk, instead of blocking all rays emanating from a given unit disk. Jovanović et al. [7] show that the minimum number of unit disks needed to block all rays emanating from a single point is quadratic in d. In addition, we refer to Fulek, Holmsen and Pach [3], who focus on hitting a maximum number of disks with one ray from an arbitrary point, while we aim at blocking all rays emanating from a given disk with a minimum number of disks. The problem of our interest is also related to the work of [2] and [1], where the authors consider an illumination problem for maximal disk packings by proving the existence of points that are not visible from outside a disk packing. We are not aware of other work that is closely related, although there are many more remotely related visibility problems; see e.g. Chapter 28 on visibility by O'Rourke in [8]. For further details on object detection on related table-top devices we refer to [5, 4].

2 Blocking rays

In this section we propose an ordering of disks that enables blocking all rays from $\mathcal{R}$ for a given number N of disks. We assume for convenience that $N = 6n$. The N disks are placed on a circle c concentric to the given disk U, such that the centers of the disks are on

the circle c and there is no gap between neighboring disks; see Figure 1. The radius R_c of circle c is easily derived from $R_c = 1/\sin\frac{\pi}{6n}$. Given the mutual tangency of each pair of neighboring disks, one can easily see that any ray $r \in \mathcal{R}$ is blocked by at least one and at most two disks of the given set of $6n$ disks. Hence, these disks form a blocking set. The minimum of all pair-wise distances between the disks is 2. Therefore, the constructed blocking set is 2-apart and we denote it by $\mathcal{D}_2$.

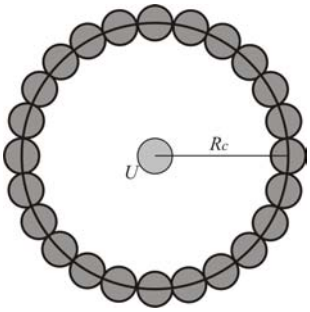


Figure 1: 24 disks positioned on the circle of radius R_c concentric to U.

For the maximum distance blocking set problem, we are interested in the maximum distance d for which the $6n$ disks form a d-apart blocking set for $\mathcal{R}$. As such, the problem appears to be hard: constructing a d-apart blocking set for an arbitrary d is certainly challenging. Therefore, we focus on *transforming* $\mathcal{D}_2$ into a d-apart blocking set. The transformation consists of separating the disks of $\mathcal{D}_2$ from each other, while the blocking of all rays is preserved.

One can easily prove that *the rays blocked by a given disk $D \in \mathcal{D}_2$ are still blocked by D after the disk is moved towards the center of U*, i.e. along the line segment that connects the two disks' centers. Consequently, a transformation of the blocking set $\mathcal{D}_2$ where some disks of $\mathcal{D}_2$ are shifted from their original position on circle c towards the center of U represents a transformation into a d-apart blocking set, where d is the minimum of all pair-wise distances between the disks; see Figure 2. The problem of our interest now is to determine the maximum d for which we can transform $\mathcal{D}_2$ into a d-apart blocking set.

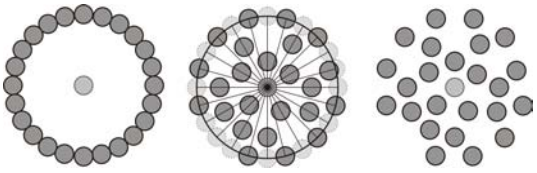


Figure 2: Transformation of $\mathcal{D}_2$ into a d-apart blocking set.

3 Ordering disks on circular rings

Let $\mathcal{D}_2$ be the 2-apart blocking set constructed as explained in Section 2, for a given integer n. In the interior of the circle c we can define a number of circles called *rings* and denoted as $c_1, c_2, \ldots, c_k$, where the radius of the ring c_1 is d_r, the radius of c_2 is $2d_r$, etc. The last ring c_k with the radius kd_r is assumed to be the given circle, which has radius R_c; see Figure 3. In the process of shifting the disks of $\mathcal{D}_2$ towards the center, we place each of them exactly on one of the rings. In this way, the distance between any two disks positioned on different rings is at least d_r.

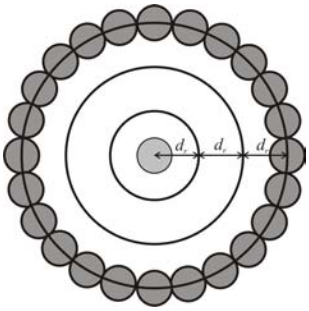


Figure 3: The definition of the circular rings.

The number k of rings determines the distance d_r for given n. Given that the radius of the largest ring is $R_c = 1/\sin\frac{\pi}{6n}$ and as we mentioned above $R_c = kd_r$, we have that

$$d_r = \frac{1}{k \sin\frac{\pi}{6n}}. \quad (1)$$

The line segment that connects the center of a disk in $\mathcal{D}_2$ and the center of U is called a *thread*. Thus, the disks of $\mathcal{D}_2$ define $6n$ threads. Since we chose to place the disks on the rings and the disks can be moved only along their threads, each disk can be placed in one of the k intersection points of its thread and the k rings.

Note that at most $6j$ disks can be placed on the j-th ring, equally spaced, such that the distance between two neighboring disks on this ring is at least d_r. In other words, at most 6 disks can be placed on the first ring, at most 12 disks on the second ring, at most 18 disks on the third ring, etc. However, the restriction of fixed positions for placing the disks, i.e. the positions defined as the intersections of the rings and threads, does not always allow placing this maximum number of disks on the rings. Therefore, we choose to place only $6n_j = 6 \cdot 2^{\lfloor \log_2 j \rfloor}$ disks on the j-th ring, such that the neighboring disks on this ring are equidistant. Since $6 \cdot 2^{\lfloor \log_2 j \rfloor} \leq 6j$, $6n_j$ disks can be placed on the j-th ring such that the distance between each pair of them is at least d_r.

First, we show that any set of n disks can be split into k subsets, where the j-th subset contains $2^{\lfloor \log_2 j \rfloor}$ disks or it is empty. The j-th subset is then placed on the j-th ring such that the distance between each

two disks is at least d_r. More precisely, we show that the given number n can be represented as

$$n = \bar{n}_1 + \bar{n}_2 + \cdots + \bar{n}_k, \tag{2}$$

where $\bar{n}_j \in \{0, n_j\}$, or simplified, any natural number n can be represented as

$$n = 1 + \underbrace{2+2}_{max\ 2} + \underbrace{4+\cdots+4}_{max\ 4} + \cdots + \underbrace{2^t+\cdots+2^t}_{max\ 2^t}, \tag{3}$$

for some $t \geq 0$. The conjecture is formally given as follows.

Lemma 1 *For any positive integer n a sequence $A_n = (a_0, a_1, \ldots, a_t)$ exists such that*

$$n = \sum_{i=0}^{t} a_i \cdot 2^i \tag{4}$$

where $a_i \in \{0, 1, \ldots, 2^i\}$ and $a_t > 0$.

Proof. The proof of the lemma follows from the binary number representation of n. □

For a given n, there are generally multiple sequences A_n. The number of rings k corresponds to the number of addends in representation (3) of n, including the zero-addends when we have less than maximum number of equal addends, for all addends except for the largest ones. Hence, the number of rings we define is

$$k = \sum_{i=0}^{t-1} 2^i + a_t = 2^t - 1 + a_t, \tag{5}$$

where a_t is the number of largest addends 2^t in (3). For example, $n = 15$ can be represented as $15 = 1 + 2 + 0 + 4 + 4 + 4$ and the number of rings needed is $k = 6$. Using a representation $15 = 1+2+0+0+0+0+4+8$, the number of rings needed is $k = 8$. From the definition of maximum distance blocking set problem and Equation (1), our interest is in the sequences A_n^* for which $2^t + a_t$ is minimal.

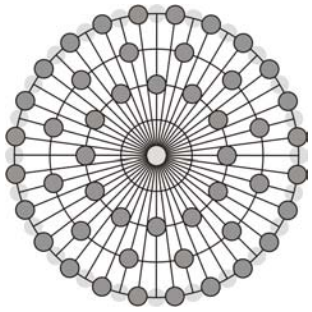


Figure 4: An example of a d-apart blocking set consisting of four rings.

Next, we present an algorithm that for each disk of $\mathcal{D}_2$ determines the ring on which it should be placed, such that the disks form a blocking set that is d_r-apart; see Figure 4. For symmetry reasons, we focus on one of the six sections of $\mathcal{D}_2$ with n disks. Furthermore, we restrict ourselves to finding the solutions for all n divisible by its largest addend 2^t in the representation (3).

The disk ordering algorithm considers only non-empty rings and it starts by placing n_k disks on the last ring. Without loss of generality, the first disk in the ordered set of n disks is placed on the last ring. As described in Section 3, $n_k = 2^t$ and since $2^t | n$, there is a subset of n_k equidistant positions on the last ring to place n_k disks. For each disk positioned on one ring, the intersection points of its thread and all other rings are tagged as "'unavailable"' positions. The algorithm continues by placing the disks of one ring in each iteration. More precisely, in iteration i, with $1 < i \leq k$, n_{k-i+1} disks are placed on equidistant positions, starting with the first available position on the ring $k - i + 1$. Figure 5 illustrates the iterations of the disk ordering algorithm for $n = 8$.

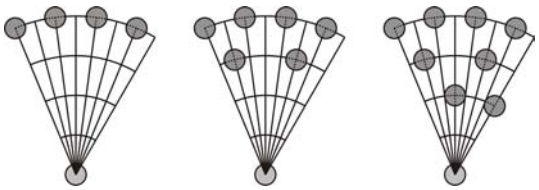


Figure 5: The three iterations of the disk ordering algorithm for $n = 8$.

4 Upper and lower bounds

In Sections 2 and 3, we showed that we can construct a d_r-apart blocking set for each n that is divisible by its largest addend in representation (3). In this section, we present upper and lower bounds on the cardinality N_d of such a blocking set, as a function of minimum distance d. We start by deriving an upper bound.

One can easily show that the ordering of disks presented in Section 3 implies that the minimum of all pair-wise distances between the disks is $d = d_r$. Hence, we have

$$d = \frac{1}{k \sin \frac{\pi}{6n}} \tag{6}$$

From the choice of sequence A_n^* in Lemma 1, for which $a_t + 2^t$ is minimal, we have that

$$\sum_{j=0}^{t-1} 2^{2j} \leq n \leq \sum_{j=0}^{t-1} 2^{2j} + a_t \cdot 2^t \tag{7}$$

where a_t is the number of largest addends 2^t in rep-

resentation (3). From

$$\sum_{j=0}^{t-1} 2^{2j} = \frac{1}{3}(4^t - 1) \tag{8}$$

it follows that

$$2^t \leq \sqrt{3n+1} \tag{9}$$

When bounding the number of rings k by a function in n, we can prove that $\sqrt{3n+1}-k$ has its local minima at $k = 2^t$, for $t = 0, 1, \ldots$. Hence, using (9), we have

$$k \leq \sqrt{3n+1} \tag{10}$$

We transform (6) into

$$\frac{1}{kd} \leq \sin\frac{\pi}{6n} \tag{11}$$

and multiply (10) by $\sqrt{n}$

$$k\sqrt{n} \leq \sqrt{3n^2+n} \tag{12}$$

Multiplication of (11) and (12) and expressing the limit for $d \to \infty$, results in

$$\lim_{d\to\infty} \frac{n}{d^2} \leq \frac{\pi^2}{12} \tag{13}$$

and since $N = 6n$, we derived an upper bound on N_d, i.e.

$$\lim_{d\to\infty} \frac{N_d}{d^2} \leq \frac{\pi^2}{2} \tag{14}$$

In [7], the authors proved that the lower bound on the minimum number of disks which form a d-apart blocking set for the set of all rays emanating from a single point is

$$\lim_{d\to\infty} \frac{N_d}{d^2} \geq \frac{\pi^2}{16} \tag{15}$$

To block the rays emanating from a given unit disk we need at least as many disks as to block the rays emanating from its center. Hence, a lower bound on the minimum number N_d of disks is given by (15).

Combining the results of (14) and (15), we proved the following theorem.

Theorem 2 *For the minimum cardinality N_d of a d-apart blocking set to block all rays emanating from a unit disk we have*

$$\frac{\pi^2}{16} \leq \lim_{d\to\infty} \frac{N_d}{d^2} \leq \frac{\pi^2}{2}.$$

5 Conclusion

We expect that both bounds, especially the upper bound, can be further improved. The following discussion provides some directions for potential improvements.

Constructing a d-apart blocking set from $\mathcal{D}_2$ through a sequence of transformation steps where a number of disks is pushed towards the center results in the rather large constant $\pi^2/2$. The disks pushed inside circle c block much larger sets of rays than the sets of rays they block from their original positions on c. Consequently, the sets of rays blocked by two disks on different rings may not be disjunct. This implies that constructing blocking sets for which the overlap of sets of blocked rays is minimized may potentially provide a better upper bound. In addition, the number of disks on one ring is less than the maximum possible number for the majority of rings. Placing the maximum number of disks on each of the rings may further improve the upper bound. The main challenge here is still the problem of proving that a set of disks, positioned following some constraints, is a blocking set for the set $\mathcal{R}$ of all rays.

References

[1] A. Dumitrescu, and M. Jiang. *The forest hiding problem.* Proceedings of the 21st ACM-SIAM Symposium on Discrete Algorithms, 2010, Austin, Texas, USA.

[2] J. Mitchell. *Dark points among disks.* Open Problems from the 2007 Fall Workshop in Computational Geometry, Hawthorne, New York, USA.

[3] R. Fulek, A.F. Holmsen, and J. Pach. *Intersecting convex sets by rays.* Proceedings of the 24th Annual ACM Symposium on Computational Geometry, 2008, College Park, MD, USA, 385–391.

[4] G. Hollemans, T. Bergman, V. Buil, K. van Gelder, M. Groten, J. Hoonhout, T. Lashina, E. van Loenen, and S. van de Wijdeven. *Entertaible: Multi-user multi-object concurrent input.* Adjunct Proceedings: Demonstrations, Proceedings 19th Annual ACM Symposium on User Interface Software and Technology, 2006, Montreux, Switzerland, 55–56.

[5] N. Jovanović, J. Korst, and V. Pronk. *Object Detection in Flatland.* Proceedings of the 3rd International Conference on Advanced Engineering Computing and Applications in Sciences, 2009, Sliema, Malta.

[6] N. Jovanović, J. Korst, and A.J.E.M. Janssen. *Minimum blocking sets of circles for a set of lines in the plane.* Proceedings of the 20th Canadian Conference on Computational Geometry, 2008, Montréal, Canada, 91–94.

[7] N. Jovanović, J. Korst, R. Clout, V. Pronk and L. Tolhuizen. *Candle in the Woods: Asymptotic Bounds on Minimum Blocking Sets.* Proceedings of the 25th ACM Symposium on Computational Geometry, 2009, Aarhus, Denmark, 148–152.

[8] J. O'Rourke. Visibility, chapter 28 in: J.E. Goodman & J. O'Rourke, *Handbook of Discrete and Computational Geometry*, 2nd Edition, 2004, Chapman & Hall/CRC, 643–664.

Largest Inscribed Rectangles in Convex Polygons (Extended Abstract) *

Christian Knauer † Lena Schlipf† Jens M. Schmidt† Hans Raj Tiwary ‡

Abstract

We consider approximation algorithms for the problem of computing an inscribed rectangle having largest area in a convex polygon on n vertices. If the order of the vertices of the polygon is given, we present a deterministic approximation algorithm that computes an inscribed rectangle of area at least $1-\epsilon$ times the optimum in running time $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\log n)$. Furthermore, a randomized approximation algorithm is given that works with high probability and achieves a running time of $O(\frac{1}{\epsilon}\log n)$.

1 Introduction

Much work has been devoted to compute inscribed objects of maximum area in a polygon in the past. Most contributions to this problem focus on objects that are again polygons, e. g., on largest axis-aligned rectangles in convex or non-convex polygons [1, 5], on largest squares and equilateral triangles in convex polygons [6], and on largest empty rectangles on point sets [4].

A rectangle is said to be *fat* if the aspect ratio of the rectangle is bounded by a constant, say c. Hall-Holt et al. [7] consider the problem of computing a *c-fat* rectangle with maximum area in a simple polygon and present a PTAS for this problem assuming that the largest inscribed rectangle is fat. In this paper, we consider the following problem: Given a convex polygon P with n vertices, compute a rectangle of largest area that is inscribed in P. Our results show that fatness is not required for approximating the largest inscribed rectangle if the input polygon is convex.

Furthermore, the dependence of the algorithm of Hall-Holt et al. is linear in the size of the input polygon. Since the algorithm is not described in detail in [7], it is not clear if a sublinear running time can be obtained when the ordering of vertices of the input polygon is known. Under the assumption that we have that ordering, we show that approximation algorithms exist, whose running times are only logarithmically dependent on n.

*Work by Schlipf, Schmidt and Tiwary was supported by the Deutsche Forschungsgemeinschaft within the research training group "Methods for Discrete Structures" (GRK 1408).

†Institut für Informatik, Freie Universität Berlin, Germany, {knauer,schlipf,jens.schmidt}@mi.fu-berlin.de

‡Institut für Mathematik, Technische Universität Berlin, Germany, tiwary@math.tu-berlin.de

Such assumptions on the input are common when handling polygons, in fact, Alt et. al show under that assumption that the largest axis-aligned inscribed rectangle inside a convex polygon can be computed in logarithmic time [1]. When the ordering in not known, it can be easily computed using standard convex hull algorithms in $\mathcal{O}(n\log n)$ time; throughout the paper we will assume that the ordering is given.

The main result of this paper can be stated as follows.

Theorem 1 *Let P be a convex polygon with n vertices. Suppose the vertices of the polygon are given in, say, clockwise order. Then, an inscribed rectangle in P with area of at least $(1-\epsilon)$ times the area of the largest inscribed rectangle can be computed*

- *in $\mathcal{O}(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\log n)$ deterministic time.*
- *with high probability in $O(\frac{1}{\epsilon}\log n)$ time.*

2 Preliminaries

We denote the area of a polygon P by $|P|$. A line segment through two points a and b is denoted by $\overline{ab}$ and its length by $|\overline{ab}|$. For a given convex polygon P, let R_{opt} be a largest inscribed rectangle. Note that in general the largest inscribed rectangle is not unique; we will sometimes use R_{opt} to denote any one of the largest inscribed rectangles.

We want to approximate the largest inscribed rectangle in a convex polygon. If we know the direction d_{opt} of one of the sides of R_{opt}, we can compute the largest rectangle R_{opt} itself in $O(\log n)$ time by applying the algorithm of Alt et al. [1]. The general idea of our algorithm is to approximate the direction of alignment of a largest inscribed rectangle and to prove that the area of the largest inscribed rectangle aligned along this direction also approximates $|R_{opt}|$. For the computation, we construct a set of candidate directions and find the largest inscribed rectangle along each of this directions using the algorithm of Alt et al.. The number of candidate directions will be $\mathcal{O}(\frac{1}{\epsilon}\log\frac{1}{\epsilon})$ for the deterministic version of our algorithm, and $\mathcal{O}(\frac{1}{\epsilon})$ for the randomized version.

3 Approximating the direction of R_{opt}

We want to find a direction close enough to the direction of any side of R_{opt}. To define what close enough

means, first suppose that we know R_{opt} and denote the intersection of its diagonals as its *center* s (see Figure 1). Let $\overline{ab}$ be one of the two shortest sides of R_{opt} and let d be the midpoint of the segment $\overline{ab}$. Then $\sphericalangle(asb) \leq \frac{\pi}{2}$ and we can define the triangles T_1 and T_2 as the two triangles with vertices s, d and the third vertex being either $f_1 := d + \epsilon(b - d)$ or $f_2 := d - \epsilon(b - d)$. Analogously, choosing the side of R_{opt} opposite of $\overline{ab}$ gives the two triangles T_3 and T_4 having the same area. The area for each triangle T_i is $\epsilon|\overline{db}||\overline{sd}|/2$ and therefore an $\epsilon/8$-fraction of $|R_{opt}| = 4|\overline{db}||\overline{sd}|$. We define a direction to be *ϵ-close* if the line through s with that direction intersects $\overline{f_1 f_2}$.

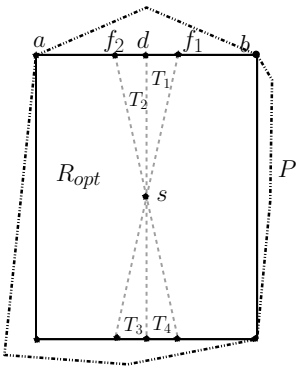


Figure 1: A largest rectangle R_{opt} in a polygon P. The area for each T_i, $1 \leq i \leq 4$, is an $\epsilon/8$-fraction of $|R_{opt}|$.

Now we show that an ϵ-close direction gives us a rectangle R_{apx} with $|R_{apx}| \geq (1 - c\epsilon)|R_{opt}|$ for a constant c. This can be reformulated to an $(1-\epsilon)$ approximation later by replacing $c\epsilon$ with ϵ' at the expense of an additional (but small) constant factor in the running time.

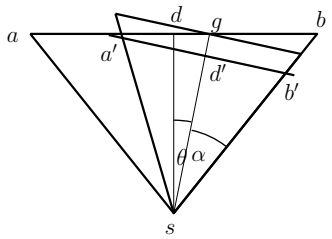


Figure 2: The triangle $T = asb$.

Let us consider the triangle $T = asb$ in R_{opt} (see Figure 2) and an ϵ-close direction d_{apx} that intersects w.l.o.g. $\overline{df_1}$ (the case for $\overline{df_2}$ is symmetric). We first rotate T around s until $\overline{sd}$ is aligned with d_{apx} and denote the angle between d_{opt} and d_{apx} by θ. Then we scale the triangle such that s and $\overline{sd}$ are preserved and the triangle is still isosceles, aligned with $\overline{sb}$ and fits completely into T. This rotation and scaling maps a to a point a', b to a point $b' \in \overline{sb}$ and we get a new, smaller triangle $T' = a'sb'$ that lies completely inside T. Consider the midpoint d' of the segment $\overline{a'b'}$. The segment $\overline{sd'}$ is aligned with the direction d_{apx}. Let α be the angle between d_{apx} and $\overline{sb}$.

Instead of comparing $|R_{apx}|$ with $|R_{opt}|$ directly, we now compare the triangles T and T'. If we can show that $|T'| \geq (1 - c\epsilon)|T|$ for some constant c, then the largest rectangle aligned to d_{apx} has at least an area of $(1 - c\epsilon)|R_{opt}|$. The reduction to triangles does not matter for the approximation, as $|R_{opt}| = 4|T|$ and $|R_{apx}| \geq 4|T'|$.

Theorem 2 *For the triangles T and T' holds that $|T'| \geq (1 - 6.2\epsilon)|T|$. Thus, $|R_{apx}| \geq (1 - 6.2\epsilon)|R_{opt}|$.*

Proof omitted.

4 How to get an ϵ-close direction

So far we have established that an ϵ-approximation algorithm only has to find a ϵ-close direction to d_{opt}. We get such an ϵ-close direction as follows: Assume first that we know the center s of R_{opt}. If we choose $O(\frac{1}{\epsilon})$ random points uniformly distributed inside P, with high probability (*w.h.p.*) at least one of them lies in the triangle T', which gives us immediately an ϵ-close direction. As we do not have the information where s is, assume that any other point p inside R_{opt} is given. Then there is a translated copy of T', where the translation maps s to p. Note that at least one of the triangles $T'_1, \ldots, T'_4$ in T' is inside P, as P is convex. Picking a point q' in this translated copy and taking the direction $\overline{pq}$ has the same effect as picking a point q in T' and taking the direction $\overline{sq}$. So we do not have to compute s explicitly, and we just have to find a point inside R_{opt}.

Even though we do not know R_{opt}, picking points from it essentially amounts to picking points from the input polygon because the area of the largest inscribed rectangle in a convex polygon is at least constant factor of the area of the polygon. More formally,

Lemma 3 ([9]) *Let P be a convex polygon and R_{opt} a largest inscribed rectangle in P, then $|R_{opt}| \geq |P|/2$.*

4.1 Randomized algorithm

It follows from Lemma 3 that if we pick k points sampled uniformly at random from a convex polygon P, then the expected number of points inside R_{opt} is $\frac{k}{2}$. Furthermore, these points are distributed uniformly at random inside R_{opt}. Thus, we have an algorithm for approximating the largest inscribed rectangle under the assumption that we can generate random points inside P efficiently. We summarize the steps in Algorithm 1.

Algorithm 1 Computes for a given convex polygon P and a source of random points in P an $(1-\epsilon)$ approximation R_{apx} for the largest inscribed rectangle in P with high probability.

1: Take $\mathcal{O}(1)$ points in P uniformly at random (*u.a.r.*) and store them in U.
2: Take $\mathcal{O}(1/\epsilon)$ points in P u.a.r. and store them in V.
3: $|R_{apx}| = 0$
4: **for all** $u \in U$ **do**
5: **for all** $v \in V$ **do**
6: Compute the largest inscribed rectangle S that is axis-aligned to $\overline{uv}$.
7: **if** $|S| \geq |R_{apx}|$ **then**
8: $R_{apx} = S$
return R_{apx}

It is easy to see that with a preprocessing of $\mathcal{O}(n \log n)$ we can create a data structure for a (not necessarily convex) polygon P that returns a point distributed uniformly at random inside P in $\mathcal{O}(\log n)$ time. This can be achieved by first computing a triangulation of the point set and then creating a binary tree with the triangles as leaves, where the weight of any node is the sum of areas of all triangles contained in the subtree rooted at that node. Sampling a random point from P then amounts to traversing this tree from root to a leaf and following the left or the right child at any node with the probability proportional to their weights.

Since the ordering of the vertices of P is given and we want to avoid any preprocessing for P, we will not sample points from P uniformly at random. Instead, we take a uniform distribution over a square and "fit" these points inside the polygon. Thus, the sampling from P will simulate the sampling of random points from a square. Let v_t, v_b be the topmost and the bottommost vertices of P. We pick a height h between the two vertices uniformly at random. We take the longest horizontal segment that fits inside P at this height and pick a point uniformly at random on this segment. This will be our sample point in P. We can repeat this process as many times as desired to get a large set of sample points that are in P.

We need to show that such a sampling works for our algorithm. Recall that we need two points p and q from P such that p lies in a largest inscribed rectangle R_{opt} and q lies in a triangle of area $\Omega(\epsilon)$ that is a translated copy of one of the T_i's (see Figure 1). We will show that with our sampling method, the probability of a sample point to lie in any convex region Q of area $\epsilon|P|$ is at least $\frac{\epsilon}{2}$.

Let L_h be the length of the largest horizontal segment inside P at height h, and l_h be the length of the largest horizontal segment inside Q at height h. Also, assume that the bottommost and topmost points in

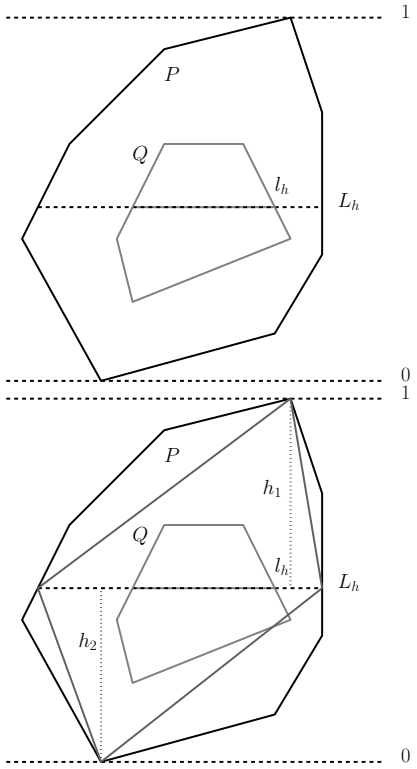


Figure 3: Sampling from a convex region Q in P.

P are at heights 0 and 1 respectively (see Figure 3). It holds $\frac{|Q|}{|P|} = \frac{\int_0^1 l_h\, dh}{\int_0^1 L_h\, dh}$. The probability that a sample point using the above sampling method lies in Q is $\int_0^1 \frac{l_h}{L_h}\, dh$. Since for any value of h we can find a quadrilateral that fits inside P and has area at least $\frac{L_h}{2}$, we have that $\frac{L_h}{2} \leq \int_0^1 L_h\, dh$. Therefore,

$$\int_0^1 \frac{l_h}{L_h}\, dh \geq 2\frac{\int_0^1 l_h\, dh}{\int_0^1 L_h\, dh}$$

Each of these sample points can be generated in logarithmic time assuming that the ordering of vertices of P is known in advance. Thus, the complexity of our algorithm is $\mathcal{O}(\frac{1}{\epsilon}\log n)$.

4.2 Deterministic algorithm

The algorithm for the deterministic case only needs to compute the sample points in a different way. First, observe that if we have a bounding box of P, whose area is proportional to that of P, then an $\mathcal{O}(\frac{1}{\epsilon} \times \frac{1}{\epsilon})$ grid on this bounding box allows us to have $\mathcal{O}(\frac{1}{\epsilon} \times \frac{1}{\epsilon})$ grid points inside P. For such a grid, the number of points in any convex region Qinside P that has an area of at least ϵ will contain at least $\mathcal{O}(\frac{1}{\epsilon})$ of the grid points.

We compute an enclosing rectangle R_e, such that $P \subset R_e$ and $|R_e| \leq 2|P|$. This can be done in logarithmic time as follows.

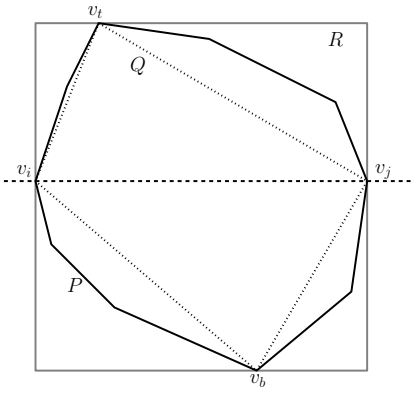


Figure 4: Polygon P and enclosing rectangle R, $|P| \geq |Q|$.

Lemma 4 *Let P be a convex polygon with n vertices, which are given in cyclic order. There is an algorithm that computes an enclosing rectangle R such that $P \subset R$ and $|R| \leq 2|P|$ in $O(\log n)$ time.*

Proof. We can compute an antipodal pair of points of the vertices of P in $O(\log n)$ time. Let this be the vertices v_i and v_j and let $l = |\overline{v_i v_j}|$ (see Figure 4). Take the direction of l as x-axis and compute a vertex v_t of P with highest y-value and a vertex v_b of P with lowest y-value. Let the vertical distance between v_b and v_t be k. We know that $|P| \geq \frac{kl}{2}$ and the rectangle R through vertices v_t, v_j, v_b and v_i has area $|R| = kl$. □

We compute a $8\sqrt{2} \times 8\sqrt{2}$ grid on R_e, so we know that at least 8×8 of the grid points lie in P and so at least one of this grid points lies in the largest inscribed rectangle of P. Also, a $\frac{\sqrt{2}}{\epsilon} \times \frac{\sqrt{2}}{\epsilon}$ grid on R_e is computed, and at least $\frac{1}{\epsilon} \times \frac{1}{\epsilon}$ of the grid points lie in P and at least one points lies in the right ϵ-fraction of P. Hence, we can take the direction of any pair of the grid points of both grids and compute tha largest inscribed rectangle aligned to that direction. So we get one rectangle that has at least an area of $(1-\epsilon)$ times the area of the largest inscribed rectangle. The running time is $\mathcal{O}(\frac{1}{\epsilon^2} \log n)$. We can further reduce the running time to $\mathcal{O}(\frac{1}{\epsilon} \log \frac{1}{\epsilon} \log n)$ by using the tools from the theory of ϵ-nets [8].

We omit further technical details due to the space limit.

5 Largest inscribed rectangles in simple polygons

The same ideas can be used to approximate the largest inscribed rectangle in a simple polygon with or without holes. We know that the largest inscribed rectangle in a simple polygon (with or without holes) on n vertices has an area of at least $\frac{1}{2(n-2)}$ times the area of the polygon. A largest axis-aligned rectangle in a simple polygon can be computed in $O(n \log n)$ time [2] and in a simple polygon with holes in $O(n \log^2 n)$ [5]. An inscribed rectangle that has at least $1-\epsilon$ times the area of the largest inscribed rectangle can be computed w.h.p. in $O(\frac{1}{\epsilon} n^3 \log n)$ time in simple polygons and in $O(\frac{1}{\epsilon} n^3 \log^2 n)$ time in polygons with holes, since we are dealing with an $O(\frac{\epsilon}{n})$-fraction for the triangle T and with an $O(\frac{1}{n})$-fraction for the rectangle.

6 Conclusion

We have considered the problem of how to approximate the largest inscribed rectangle in a convex polygon. If the order of the vertices of the polygon is given, we have presented a deterministic algorithm that computes in $O(\frac{1}{\epsilon} \log \frac{1}{\epsilon} \log n)$ time an inscribed rectangle of area at least $(1-\epsilon)$ times of the largest inscribed rectangle in the convex polygon. Also, a randomized algorithm is given, that achieves a running time of $O(\frac{1}{\epsilon} \log n)$ and computes w.h.p. an inscribed rectangle of area at least $(1-\epsilon)$ times of the largest inscribed rectangle in the convex polygon.

References

[1] H. Alt, D. Hsu, and J. Snoeyink. Computing the largest inscribed isothetic rectangle. In *Proc. 7th Canad. Conf. Comput. Geom.*, pages 67–72, 1995.

[2] R. P. Boland and J. Urrutia. Finding the largest axis-aligned rectangle in a polygon in O(n log n) time. In *In Proc. 13th Canad. Conf. Comput. Geom*, pages 41–44, 2001.

[3] J. E. Boyce, D. P. Dobkin, R. L. Drysdale, III, and L. J. Guibas. Finding extremal polygons. In *Proc. 14th Annu. ACM Sympos. Theory Comput.*, pages 282–289, 1982.

[4] J. Chaudhuri, S. C. Nandy, and S. Das. Largest empty rectangle among a point set. *J. Algorithms*, 46(1):54–78, 2003.

[5] K. Daniels, V. Milenkovic, and D. Roth. Finding the largest area axis-parallel rectangle in a polygon. *Comput. Geom. Theory Appl.*, 7:125–148, 1997.

[6] A. DePano, Y. Ke, and J. O'Rourke. Finding largest inscribed equilateral triangles and squares. In *Proc. 25th Allerton Conf. Commun. Control Comput.*, pages 869–878, Oct. 1987.

[7] O. Hall-Holt, M. J. Katz, P. Kumar, J. S. B. Mitchell, and A. Sityon. Finding large sticks and potatoes in polygons. In *SODA '06*, 2006.

[8] J. Matoušek. Approximations and optimal geometric divide-and-conquer. In *STOC '91: Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 505–511, 1991.

[9] K. Radziszewski. Sur une problème extrémal relatif aux figures inscrites et circonscrites aux figures convexes. *Ann. Univ. Mariae Curie-Sklodowska, Sect. A6*, pages 5–18, 1952.

2-Factor Approximation Algorithm for Computing Maximum Independent Set of a Unit Disk Graph

Sudeshna Kolay* Subhas C. Nandy† Susmita Sur-Kolay‡

Abstract

A 2-factor approximation algorithm for finding a maximum independent set of a given unit disk graph is proposed. The time complexity of our algorithm is $O(n^4)$. In particular, if it is a coin graph, then our approximation algorithm works in $O(n \log n)$ time. An $O(n^2)$ time heuristic is also reported that produces comparable results on randomly generated instances of unit disk graphs of fairly large size.

1 Introduction

Unit disk graphs play an important role in formulating several problems in mobile ad hoc networks. A *unit disk graph* $G = (V, E)$ is the intersection graph of a set of circular disks $\{c_1, c_2, \ldots, c_n\}$ of same radius placed in a two-dimensional (2D) plane. Each vertex $v_i \in V$ corresponds to a disk c_i, and an edge $(v_i, v_j) \in E$ between vertices v_i and v_j indicate that the corresponding pair of unit disks c_i and c_j intersect (overlap). In mobile networks, the base stations can be viewed as vertices of a unit disk graph, where the range of the base stations is the same. Various practical problems on this network in terms of unit disk graphs. In this paper, we consider the problem of computing the maximum independent set (MIS) [8] of a given unit disk graph, where the co-ordinates (x_i, y_i) of the center of each disk c_i are given.

The problems of computing a maximum independent set and a minimum vertex cover for unit disk graphs are known to be NP-complete [2]. Thus, research on this topic is concentrated on designing efficient approximation algorithms. Most of the related works assume that the geometric representation or the layout of the unit disks are given.

Earlier results: A dynamic programming based shifting strategy was used by Erlebach et al.[7] to design a polynomial time approximation scheme (PTAS) for finding a maximum weighted independent set (disjoint disks of maximum total weight) in a disk graph for a set of n disks with arbitrary radii. Their proposed algorithm achieves an approximation factor $(1+\epsilon)$ in time $O(n^{O(\frac{1}{\epsilon^2})})$, where $\frac{1}{\epsilon}$ is an integer greater than 1. Next, if all the disks have same radius (without loss of generality the diameter can be assumed to be 1) and their centers lie inside a region bounded by a pair of parallel lines at a distance at most k apart, then an algorithm by Matsui [10] computes an optimal MIS in $O(n^{4\lceil \frac{2k}{\sqrt{3}} \rceil})$ time. He also gave a $(1 + \frac{1}{(r-1)})$-factor approximation algorithm for the MIS problem of unit disk graphs that runs in $O(rn^{4\lceil \frac{2(r-1)}{\sqrt{3}} \rceil})$ time, for any positive integer r. Jan van Leeuwen [9] introduced the concept of thickness to propose a fixed parameter tractable algorithm for finding the MIS of a unit disk graph. An instance of unit disk layout is said to have thickness τ if the 2D region can be split into a set of strips of width 1 such that each strip contains at most τ disk centers. Further, he showed that an instance of the problem with thickness τ can be solved in $O(\tau^2 2^{2\tau} n)$ time. Although there are PTAS for finding an MIS of a given unit disk graph, the *best known constant factor approximation* available for this problem achieves an approximation factor 3 as follows with time complexity $O(n^2)$ [11]:

(i) sort the disks with respect to the x-coordinates of their centers in ascending order;
(ii) *repeat*
add the left-most disk c to the MIS; (There can be at most 3 disks that overlap c, and yet are mutually disjoint. If we choose c in the independent set, then we can lose at most 3 disks in the independent set.)
delete c and all the disks that overlap c;
until no disk remains.

It needs to be noted that Agarwal et al. [1] proposed a 2-factor approximation algorithm for the MIS problem of the rectangle intersection graph with a set of rectangles of fixed width. They also proposed a PTAS for the MIS problem which produces a $(1 + \frac{1}{k})$-factor approximation result in $O(n \log n + n^{2k-1})$ time for any integer k. Recently, Chan and Har-Peled [4] addressed the MIS problem for pseudo-disks in the plane. In the unweighted case of a set of n pseudo-disks, they provided a $(1+\epsilon)$-factor approximation in time $O(n^{O(\frac{1}{\epsilon^2})})$, similar to Erlebach's results [7] for n disks. For the weighted case, they proposed an $O(n^3)$ time algorithm to produce an independent set of total weight $\Omega(OPT)$ where OPT is the maximum weight over all independent sets, provided the set of pseudo-disks has linear union complexity.

Our main results: First we improve the approximation factor to 2 for computing the MIS of a unit

*Chennai Mathematical Institute, Chennai, India
†Indian Statistical Institute, Kolkata, India
‡Indian Statistical Institute, Kolkata, India

disk graph. In other words, if the size of the optimum solution for a given instance of the problem is OPT, then our algorithm produces a solution of size at least $\frac{1}{2}OPT$. The worst case time complexity of our proposed algorithm is $O(n^4)$, which is less than that of the approximation algorithm by Matsui [10] for any integer $r \geq 2$.

Second, we consider the MIS problem for the *coin graph*, which is also a unit disk graph, but proper overlap between a pair of disks is not allowed. Here an edge between a pair of vertices implies that the corresponding unit disks touch each other. The MIS problem for the coin graph is also known to be NP-complete [3]. Our algorithm produces 2-factor approximation result for the coin graph in $O(n \log n)$ time.

Finally, we propose a very easy-to-implement $O(n^2)$ time heuristic for MIS of unit disk graphs which has been observed to produce results comparable to our 2-approximation algorithm on randomly generated instances of fairly large size.

2 2-factor algorithm for MIS of unit disk graphs

A layout of unit disks and the corresponding unit disk graph are shown in Figures 1(a) and 1(b). An independent set in this unit disk graph consists of a set of disks $IS \subseteq C$ which are mutually non-intersecting. The objective of the MIS problem is to find the largest subset of C which are mutually non-intersecting disks.

As in the 2-factor approximation algorithm for the MIS problem of fixed width rectangle intersection graph [1], we split the region as shown in Figure 1(c) into a set of m ($\leq$ n) disjoint strips $\{H_1, H_2, \ldots, H_m\}$, separated by horizontal lines at y-coordinates $\{h_1, h_2, \ldots, h_{m+1}\}$, where $h_0 = \lceil \max_{1 \leq j \leq n}(y_j) \rceil$, $h_{m+1} = \lfloor \min_{1 \leq j \leq n}(y_j) \rfloor$ and $h_{i-1} - h_i = 1$ for all $i = 1, 2, \ldots, m+1$. In other words, each strip is of width 1. The j-th strip H_j contains the set of centers $P_j = \{c = (x, y) | h_{j-1} < y \leq h_j\}$. Note that each disk center c_i lies in exactly one strip (see Figure 1(c)). We study the problem under the general position assumption where no disk-center has integer y-coordinate, and each unit disk intersects exactly one horizontal line.

It is shown in [10] that if the width of a strip is $\frac{\sqrt{3}}{2}$, then the optimum solution for the MIS problem inside that strip can be computed in $O(n^2)$ time where n is the number of disk-centers inside that strip. We show that for a strip of width 1 also, the optimum solution of the MIS problem can be obtained in polynomial time. We propose an $O(n^4)$ time algorithm for this subproblem, and use it to design our 2-approximation algorithm for the MIS problem of unit disk graph.

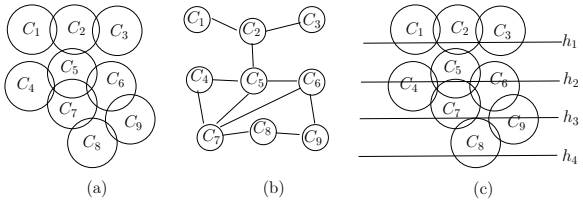


Figure 1: (a) A layout of unit disks, (b) corresponding unit disk graph, and (c) horizontal lines at distance 1 dividing the region into strips

Result 1 *If I_j denotes an independent set of disks in the strip H_j, then $I_j \cap I_k = \emptyset$ if $|j - k| > 1$. However, $I_j \cap I_{j+1}$ may be non-empty.*

Following the technique in the 2-factor approximation algorithm of equal width rectangle intersection graph [1], we compute the MIS for the disks in each strip separately. Finally we compute $IS_{odd} = I_1 \cup I_3 \cup \ldots$ and $IS_{even} = I_2 \cup I_4 \cup \ldots$. We report the one among IS_{odd} and IS_{even} whose cardinality is greater than the other one. It can be shown that this produces a 2-factor approximation algorithm [1].

We first describe the method of computing the MIS of the disks whose centers lie in a strip of width 1. Consider a strip H_j, and the set of centers of the unit disks P_j. We further split the horizontal strip into unit squares by drawing vertical line segments unit distance apart, and then delete all the vertical lines that do not intersect any disk inside H_j. Let us name the active vertical lines as $v_1, v_2, \ldots, v_k$. Let C^j_α be the set of disks whose centers lie inside H_j, and are intersected by the active vertical line v_α.

Now, we have the following two observations: (i) the size of the MIS among the set of disks C^j_α may be 1 or 2, and (ii) if we consider a pair of vertical lines v_α and v_β, then there exists no pair of intersecting disks $c \in C^j_\alpha$ and $c' \in C^j_\beta$ if $|\alpha - \beta| > 1$ (see Result 1).

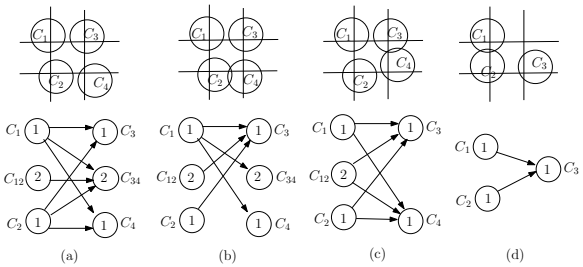


Figure 2: Assignment of edges in the graph G^j

Being motivated by the above two observations, we define a directed graph $G^j = (V^j, E^j)$ for P_j. The nodes $V^j = V^j_1 \cup V^j_2 \ldots$, where $V^j_\alpha = A^j_\alpha \cup B^j_\alpha$. Each node in A^j_α corresponds to a unit disk in C^j_α, and each node of B^j_α corresponds to a pair of non-intersecting

unit disks in C^j_α. While putting edges in E_j, we consider each pair of sets V^j_α and V^j_β ($\alpha < \beta$). Between a pair of nodes $u \in V^j_\alpha$ and $v \in V^j_\beta$, there is a directed edge (u, v) if each disk in node u does not intersect any disk in node v (Figure 2). Note that (i) there is no edge between any pair of nodes in a set V^j_α, and (ii) a pair of nodes $u \in V^j_\alpha$ and $v \in V^j_\beta$ is always connected if $|\alpha - \beta| > 1$.

G_j defined thus is a k-partite graph, where k is the number of active vertical lines for the strip H_j. Next, we add a source node s from which there is a directed edge to each of the nodes in V^j_1. The nodes in V^j_k are all connected to a sink node t by directed edges. Finally, we assign a weight equal to 1 or 2 to each node of G_j depending on the number of disks it represents. The nodes s and t are assigned with weight 0. If the number of centers of the disks in the strip H_j is τ, then $|V^j| = O(\tau^2)$ and $|E^j| = O(\tau^4)$.

We find the longest path in the directed acyclic graph G. Although the problem of finding the longest path in a directed graph is NP-complete, it can be solved in time proportional to the number of edges of the graph [6] if the graph is acyclic. Since the two sets of unit disks in the strips H_{2i+1} and H_{2j+1} are disjoint for $i \neq j$, the time required to compute the MIS for each of the odd indexed strips H_{2i+1} is $O(n^4)$, and the total time complexity for IS_{odd} is additive and hence $O(n^4)$. The same time complexity result holds for computing the MIS for all the even indexed strips H_{2i}. The final result reported by this algorithm is the larger of the two sets IS_{odd} and IS_{even}, leading us to the following theorem:

Theorem 1 *Given a set C of unit disks in a 2D plane, a set of at least $\frac{1}{2}OPT$ non-intersecting disks can be obtained in $O(n^4)$ time, where OPT is the maximum number of mutually non-intersecting disks in the set C.*

3 MIS problem for coin graphs

In the coin graph, since no two unit disks properly overlap, we can bound the number of non-intersecting unit disks on a vertical line inside a strip to at most 2. Thus, each vertical line inside a strip H_j contributes at most 3 nodes in V^j (see Figure 3). So, the total number of nodes and edges in G^j are $O(n)$ and $O(n^2)$ respectively. However, since our objective is to find the longest path in the resulting graph, we can reduce the number of edges to $O(n)$ by ignoring the edges (u, v) where $u \in V^j_\alpha$, $v \in V^j_\beta$ and $|\alpha - \beta| > 2$. Thus, the time complexity of the algorithm is dominated by the sorting time of the centers of the unit disks.

Theorem 2 *Given a set C of n coins in a 2D plane,*

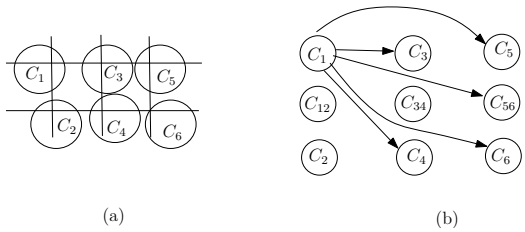


Figure 3: (a) A geometric layout of coin graph, and (b) the corresponding coin graph

a set of mutually non-touching coins of size at least $\frac{1}{2}OPT$ can be computed in $O(n \log n)$ time, where OPT is the maximum number of mutually non-touching coins present in the plane.

It needs to be mentioned that if the centers of the unit radius non-overlapping disks (coins) have integer coordinates then we can determine the *optimal* independent set of the coin graph in polynomial time based on the following facts:

(a) As the centers of the unit disks are placed on the vertices of a grid, the corresponding coin graph is planar bipartite $G = (V_1 \cup V_2, E)$; the partition V_1 (V_2) of vertices comprises the disks with centers on even (resp. odd) diagonals of the grid.

(b) A maximum independent set is obtained from a maximum bipartite matching of this coin graph, which can be computed in polynomial time [8].

4 A fast and efficient heuristic

Let C be a set of n unit disks in the plane, and $G = (V, E)$ be the corresponding unit disk graph. We repeatedly execute the following two steps until all the vertices in V are *deleted*.

Algorithm **MIS-Heuristic**
Set IS=$\emptyset$;
repeat
  choose an *unmarked* vertex v of V
    having maximum degree;
  if (v has no two neighbors that are adjacent)
    then add v to IS;
  delete v and all its incident edges from G;
  update the degrees of the remaining
    vertices accordingly;
until (all the vertices in V are deleted).

The creation of the graph G needs $O(n^2)$ time, and processing each vertex takes time equal to its degree, which may be $O(n)$. Hence the running time of the MIS-Heuristic algorithm above is $O(n^2)$.

We executed our MIS-Heuristic algorithm as well as the 2-factor approximation algorithm described in

Table 1: Comparison of the size of independent set obtained by our MIS-heuristic vs. our 2-approximation algorithm on random instances*

	# disks	Avg. $\frac{\lvert IS_H\rvert}{\lvert IS_A\rvert}$ for # strips = 1	Avg. $\frac{\lvert IS_H\rvert}{\lvert IS_A\rvert}$ for # strips = 10
1	50	0.91	1.62
2	100	0.87	1.60
3	500	0.72	1.37
4	1000	0.70	1.25
5	2000	0.67	1.06
6	4000	0.69	1.0
7	5000	0.73	0.86
8	10000	0.76	0.77

* H denotes the MIS-Heuristic presented in this section, and A the 2-approximation algorithm of Section 2.

Section 2 on random instances. For each n (number of disks), we have generated 5 random instances, and run both the algorithms. We use IS_H and IS_A to denote the size of the independent sets obtained by our MIS-Heuristic and and our 2-factor approximation algorithm respectively. We report the typical results obtained in Table 4 for two cases: (i) only one horizontal strip of width 1, and (ii) the general case with 10 horizontal strips of width 1. For the sake of comparison, we present the average of the ratio of the sizes of the independent sets obtained by the two proposed methods on the same instance.

For case (i), our 2-factor approximation algorithm is guaranteed to produce optimum solution by Theorem 1. The results obtained by MIS-Heuristic in this case is observed to be far better than $\frac{1}{2}OPT$. For other cases, it is noted that for values of n upto about 4000, our MIS-Heuristic algorithm performs better than the 2-factor approxmation. However, for large values of n, i.e., for the dense instances the 2-approximation algorithm produces marginally better result. Further, it is not essential in MIS-Heuristic that the vertices be processed in non-increasing order of their degrees. We strongly believe that a careful analysis may show that the solution produced by MIS-Heuristic will never produce a solution of size less than $\frac{1}{2}OPT$.

5 Conclusion

A 2-factor approximation algorithm for the MIS problem of a unit disk graph is proposed. Although the worst case time complexity of the algorithm is $O(n^4)$, it executes much faster for instances where the disks are uniformly distributed. Our proposed algorithm produces optimum solution if the disk centers are inside a strip of width 1, and it is faster than that of [10]. The worst case time complexity of our 2-factor approximation algorithm is less than that of [10] with $r = 2$ in their expression of time complexity. Finally, we conjecture that the proposed MIS-Heuristic algorithm, which is faster, is also a 2-factor approximation algorithm for the MIS problem. In fact, for instances with fewer than 5000 disks, it is likely to be beneficial to employ MIS-Heuristic.

References

[1] P. K. Agarwal, M. van Kreveld, S. Suri, *Label placement by maximum independent set in rectangles*, Computational Geometry, vol 11, 1998, pp. 209-218.

[2] B. N. Clark, C. J. Colbourn, D. S. Johnson, *Unit disk graph*, Discrete Mathematics, vol. 86, 1990, pp. 165-177.

[3] M. R. Cerioli, L. Faria, T. O. Ferreira and F. Protti, *On minimum clique partition and maximum independent set on unit disk graphs and penny graphs: complexity and approximation*, Electronic Notes in Discrete Mathematics, vol. 18, 2004, pp. 73-79.

[4] T. M. Chan and S. Har-Peled, *Approximation algorithms for maximum independent set of pseudo-disks*, Proceedings of the 25th Annual Symposium on Computational Geometry, 2009, pp. 333-340.

[5] P. Carmi, M. J. Katz and N. Lev-Tov, *Covering points by unit disks of fixed location*, Proceedings of the International Symposium on Algorithms and Computation (ISAAC), LNCS-4835, 2007, pp. 644-655.

[6] T. H. Cormen, C. E. Lieserson and R. L. Rivest, Introduction to Algorithms, Prentice-Hall, 2007.

[7] T. Erlebach, K. Jansen, E. Seidel, *Polynomial time approximation schemes for geometric intersection graphs*, SIAM J. Comput., vol. 34, 2005, pp. 1302-1323.

[8] M. R. Garey and D. S. Johnson, *Computers and Intractability: A guide to the theory of NP-Completeness*, Freeman, 1979.

[9] E. Jan van Leeuwen, *Approximation algorithms for unit disk graphs*, Technical Report UU-CS-2004-066, Institute of Information and Computing Sciences, Utrecht University, 2004.

[10] T. Matsui, *Approximation algorithms for maximum independent set problems and fractional coloring problems on unit disk graphs*, Proceedings of the Japan Conference on Discrete and Computational Geometry (JCDCG), LNCS-1763, 1998, pp. 194-200.

[11] M. Marathe, H. Breu, H. B. Hunt III, S. S. Ravi and D. J. Rosenkrantz, *Simple heuristics for unit disk graphs*, Networks, vol. 25, 1995, pp. 59-68.

[12] T. Nieberg, J. L. Hurink and W. Kern, *A new PTAS for maximum independent sets in unit disk graphs*, Memorandum No. 1688, University of Twente, 2003.

Visibility Polygons in the Presence of a Mirror Edge

Bahram Kouhestani* Mohammad Asgaripour* Salma S. Mahdavi* Arash Nouri*
Ali Mohades*

Abstract

Suppose P is a polygon which one of its edges is a mirror. We consider the problem of computing the visibility polygons inside P. Generally, the visibility polygon of a point or an edge or a segment, inside a simple polygon, can be computed in linear time. In this paper we will show that, even when an edge of the polygon is a mirror, the visibility polygon can still be computed in $O(n)$. Also, we will prove that computing the visibility polygon of a query point, can be done in the same time and space complexity, when an edge is a mirror.
To prove these, we will propose an optimum linear time algorithm for computing the union of two visibility polygons inside a polygon.

1 Introduction

The visibility problem arises in many computational geometry subjects, and so far various variations of this problem have been studied. For example, in Art Gallery, the goal is to find the minimum stationary guards that can see all points in a polygon. In Visibility Polygon problem we have a polygon, called P, and a viewer as a point or an edge or a segment inside P. The goal is to find the maximal subpolygon of P, in which all the points are visible to the viewer. There are linear time algorithms for computing the visibility polygon for a point [6] or a segment [4] inside a simple polygon. Also, using preprocessing, an output sensitive algorithm can compute the latter problem in time proportional to the size of the visibility polygon. Bose et al. [3] presented an algorithm with preprocessing time of $O(n^3 logn)$ and space complexity of $O(n^3)$ that can answer queries in $O(logn + k)$ time, where k is the size of the visibility polygon. Aronov et al. [2] found another solution for the same problem with a preprocessing time of $O(n^2 logn)$ and space of $O(n^2)$ in $O(log^2 n + k)$.
Visibility in the presence of mirrors was introduced by Klee, for the first time. He asked whether every polygon whose edges are mirror, is illuminable from every interior point [5]. Tokarsky in [7] constructed a polygon inside which, exists a dark point by putting the light source at a particular point.

Works similar to ours, are the papers about visibility with reflection such as [1]. In this paper, authors modeled two different reflections; specular and diffuse. In specular reflection, light rays reflect along a specific direction, abide the standard law of reflection. In diffuse reflection, light rays reflect in all directions. They show that the resulting visibility polygon may not be simple, and prove a tight $\theta(n)$ bound on the worst case combinatorial complexity, based on the number of visibility polygon vertices. They also present an algorithm with $O(n^2 log^2 n)$ time for computing it.
We consider the same problem when only one of the edges is mirror. Our algorithm runs in an $O(n)$ time. Also, we show that the visibility polygon for a query point in a polygon with only one mirror, can be computed in the same order as [2] or [3] as desired.

This paper is organized as follows: In section 2, notations are described. In section 3.1 an algorithm for computing the union of two visibility polygons inside a polygon in $O(n)$ time is presented. Sections 3.2 and 3.3, deal with the computation of visibility polygon of a point or a segment in a simple polygon with a mirror edge. In section 4, it is discussed that visibility polygon of a query point inside a polygon with a mirror edge, is the same as simple polygons. And finally, section 5 contains discussions and future works.

2 Notation

Suppose P is a simple polygon. Let $int(P)$ and $bd(P)$ denote interior and boundary of P, respectively. Also, $[a, b]$ means counterclockwise polygonal chain from a to b, where a and b are two points on $bd(P)$. Two points, x and y, are visible to each other (or can see each other), if and only if the open line segment, $\overline{xy}$, lies completely in $int(P)$. The visibility polygon of the point q in P, is denoted by $V_P(q)$ and consists of all points of P, which are visible to q. Among the edges of $V_P(q)$, those which are not edges of P, are called windows. Weak visibility polygon of a segment, s in P, is denoted $WV_P(s)$, is the maximal subpolygon of P which all points of it, are visible to at least one point of s which is not an endpoint.

*Laboratory of Algorithm and Computational Geometry, Department of Mathematics and Computer Science, Amirkabir University of Technology. b_kouhestani@alum.sharif.edu; asgaripour@aut.ac.ir; ss.mahdavi@aut.ac.ir; arash1718@aut.ac.ir; mohades@aut.ac.ir

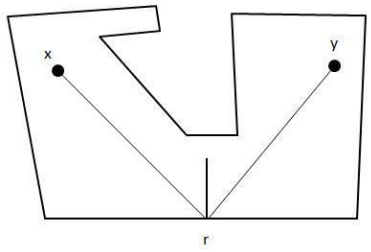


Figure 1: visible using a mirror

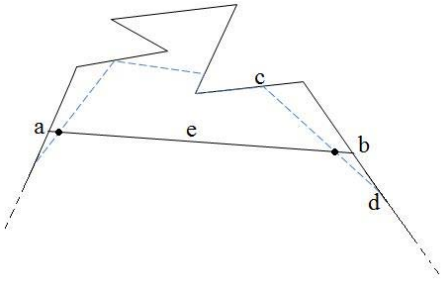


Figure 2: e can at most intersect two edges of W

$V_P(q)$ has an important property which is used in this paper several times. This property says that the order of visibility polygon's vertices and edges are exactly the same as their order on the boundary of P. We use this property, called *Order Property*, to obtain an $O(n)$ time algorithm to compute the union of two visibility polygons.
Combinatorial representation of $V_P(q)$, is a circular list of vertices and edges of P, in the order which they appear on the boundary of $V_P(q)$. Given combinatorial representation, actual coordination of each vertex can be computed in constant time.
Suppose edge e of P is a mirror. Two points x and y inside P can see each other using the mirror e, if and only if there exists a point r on e, which is visible to both x and y such as $\overline{xr}$ and $\overline{yr}$ lie on the opposite sides of the inward normal of e at r, and make the same angle with it as is depicted in figure 1 (this definition is the same as the definition of directly visible with one specular reflection in [1]).

Now visibility polygon of a point q inside P with mirror edge e, denoted by $V_{P,e}(q)$, is the maximal subpolygon of P that all of its points, are visible to q either directly or by using the mirror, e.

3 Computing visibility polygon in a polygon with one mirror edge

3.1 Union of two visibility polygons

Most algorithms in sections 3 and 4, require computing the union of two visibility polygons. Here, we present an algorithm that can do this computation in $O(n)$ time.

Lemma 1 *Suppose Q and W are two visibility polygons inside P. The Union of P and Q can be computed in $O(n)$ time.*

Proof. Suppose $e = \overline{ab}$ is an edge of Q and $e' = \overline{cd}$ is an edge of W. If e' intersects e, then either c belongs to $[a, b]$ and d belongs to $[b, a]$, or d belongs to $[a, b]$ and c belongs to $[b, a]$. Therefore, whereas both Q and W have the *Order Property*, e can at most intersect two edges of W, and the intersection of Q and W have $O(n)$ vertices (see figure 2). Vertices of the union polygon consist of vertices of Q and W (which are located at the boundary of P), and some internal vertices appeared on the intersection of Q and W. We traverse the boundary of P, if we see two consecutive vertices which belong to different polygons, for example q_i and w_j (which q_i belongs to Q and w_j belongs to W), we may need to compute the intersection point between these two vertices. This intersection point is indeed, the intersection of edge $\overline{q_i q_{i+1}}$ and $\overline{w_{j-1} w_j}$ (other cases when one of these two vertices lies on the edge of the other polygon can be handled easily).
By continuing this process the union of the two polygons can be computed in $O(n)$ time.

□

Note that, our algorithm actually works for any two polygons inside P, with the *Order Property*.

3.2 visibility polygon of a point

In this subsection an algorithm is proposed to compute $V_{P,e}(q)$ (as depicted in figure 3).
Our algorithm is composed of the following steps:

1. Compute $V_P(q)$ and determine the portion of e that can be seen by q, $\overline{v_1 v_2}$.
2. If q can not see any interior points of e, then $V_P(q) = V_{P,e}(q)$ and we are done.
3. Compute q', the projection of q using e (as a mirror).
4. Compute the first intersection of the rays $\overline{q'v_1}$ and $\overline{q'v_2}$ with the boundary of P (other than e) which is shown by t_1 and t_2 in figure 3.
5. Construct a polygon by attaching the chain $[t1, t2]$ with the chain $t_2 v_2 q' v_1 t_1$. this polygon is called P'.
6. Compute $V_{P'}(q')$.
7. Omit the triangle $q' v_1 v_2$ from $V_{P'}(q')$ and call the resulting polygon T (T is a portion of P, which q can see using the mirror, e).

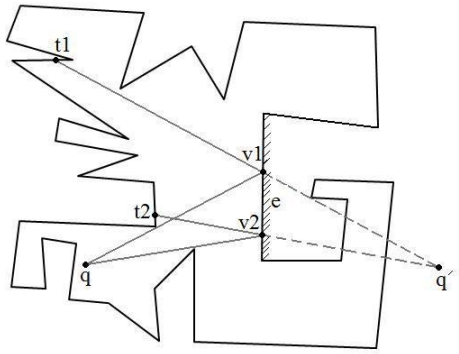


Figure 3: visibility polygon of a point inside P

8. Compute the union of T and $V_P(q)$.

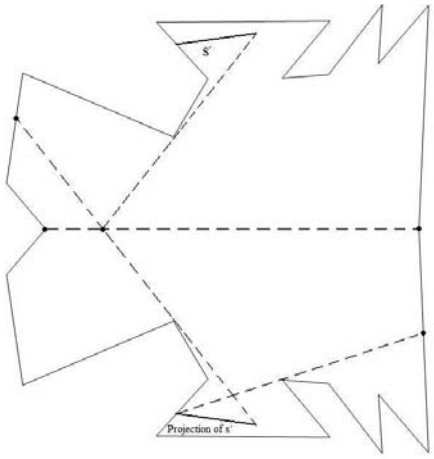


Figure 4: Construction of P'

Theorem 2 *The visibility polygon of a point inside a polygon with a mirror edge can be computed in $O(n)$ time.*

Proof. $V_{P,e}(q)$ consists of points that q can see either directly or by using mirror, e. In step 1, we compute the points that q can see directly. Polygon T which is computed in step 7, is the portion of P that q can see using the mirror, e. Therefore, the union of these two polygons is indeed, $V_{P,e}(q)$.
Obviously steps 1 to 5 can be computed in $O(n)$ time, so constructing P' can be done in linear time. Also, step 6 and 7 take $O(n)$ time, so the computation of polygon T can be done linearly. Because T and $V_P(q)$ have the *Order Property*, step 8 can be done in $O(n)$ time by using Lemma 1.

□

3.3 Weak visibility polygon of a segment

Suppose s is a segment which completely lies in P and we want to compute $WV_{P,e}(s)$. The idea is similar to the computation of $V_{P,e}(q)$. First, we compute $WV_P(s)$. If $WV_P(s)$ contains no interior points of e, the work is done. Otherwise $WV_P(e)$ is computed (e assumed to be open, not containing its endpoints). Let s' be the portion of s lies in $WV_P(e)$ (after computing $WV_P(e)$, s' can be computed in constant time). We like to compute all the points that are visible to s', using e. Let P' be the polygon obtained by adding the projection of $WV_P(e)$ using e, to $WV_P(e)$ and omitting e, as depicted in figure 4. Note that e is open, therefore, this polygon is simple.

We compute the weak visibility polygon of the projection of s' using e in P', and call it Q. $Q \cap P$ is the portion of P that is visible to s using the mirror. The union of this polygon and $WV_P(s)$ is the set of all points which are visible to s, directly or by using the mirror, e.
$Q \cap P$ can be computed in $O(n)$ time by simply cutting Q along e, using lemma 1 computation of the union needs only $O(n)$ time, therefore, all above steps can be done in $O(n)$ time. Hence, we have the following theorem.

Theorem 3 *Weak Visibility polygon of a segment inside a polygon with a mirror edge can be computed in $O(n)$ time.*

4 Visibility polygon of a query point

In some applications, one may need to compute the visibility polygon of a big number of points. Here using an $O(n)$ time algorithm is not efficient. Therefore, methods were presented using preprocessing to compute visibility polygons in a more efficient time. The time complexity of these methods (after preprocessing) are output sensitive. As we mentioned earlier two of such methods can be found in [2] and [3]. Suppose in a polygon, an edge is fixed to be a mirror. The question is: Can previous methods be extended so that they work for a polygon with a (fixed) mirror? In the following we show that the answer is yes.

First, we compute $WV_P(e)$ and construct polygon P', the same as in section 3.3.
Then we preprocess P and P' by one of the methods either presented in [2] or [3]. We also do a point planar location preprocess on a copy of P in $O(nlogn)$ time, so that we are able to answer in $O(logn)$ time whether the query point, belongs to $WV_P(e)$ or not.
Now for answering visibility polygon of a query point q, we check whether q is in $WV_P(e)$ or not. If q is not in $WV_P(e)$, q can not see any portion of the mirror

and compute the answer of query by the chosen method.
If q is in $WV_P(e)$, q', the projection of q using e, is computed. Now we can compute $V_{P'}(q')$, by querying q' in P'. We select the portion of $V_{P'}(q')$ inside $WV_P(e)$. This selection can be done in $O(|V_{P'}(q')|)$ easily, by finding intersection of $V_{P'}(q')$ and e, and then cutting $V_{P'}(q')$ in the intersection points. Call the resulting polygon Q. Q is indeed, the portion of P, that is visible to q using the mirror.
Now $V_P(q)$ is computed by querying q in P. Then the union of $V_P(q)$ and Q can be computed in $O(|Q + V_P(q)|)$ time using the method mentioned in lemma 1. The resulting polygon is obviously $V_{P,e}(q)$. The time complexity of the added preprocesses is equal to the time complexity of the chosen method. Also, for answering the query, added steps need $O(logn + |V_{P,e}(q)|)$ time. Therefore, the time and space complexity of answering visibility polygons of a query point in the presence of a fix mirror, are equal to the situation without mirror.

5 Discussion

In this paper, we showed that adding a mirror edge will not change the time and space complexity of computing visibility polygons. In [1] it is shown that having two mirrors, the resulting visibility polygon -even with one reflection- may not be a simple polygon. Also, having h mirrors, number of vertices of the resulting visibility polygon, can be $O(n + h^2)$. For h mirrors, each projection, and its relative visibility polygon can be computed in $O(n)$ time, which is leading to overall time complexity of $O(hn)$. The challenging part is the efficient computation of the union of these polygons. We proposed an efficient algorithm in the case of computing the union of two such polygons. We hope to improve this algorithm to reach an efficient algorithm for polygons in which the number of mirrors is more than one.
In section 4, we fixed an edge as a mirror. One interesting problem is to find a method for answering visibility polygon of a query point, when the mirror edge, is also queried.

Acknowledgments

We would like to thank members of our research group, especially Amin Gheibi, for helping us with this paper.

References

[1] B. Aronov, A. Davis, T. Dey, S. P. Pal, D. Prasad. *Visibility with one reflection.* Discrete & Computational Geometry, (19), pp. 553-574, 1998.

[2] B. Aronov, L. J. Guibas, M. Teichmann, and L. Zhang. *Visibility queries and maintenance in simple polygons.* Discrete & Computational Geometry, (27), pp. 461-483, 2002.

[3] P. Bose, A. Lubiw, and J. Munro. *Efficient visibility queries in simple polygons.* Computational Geometry: Theory and Applications, (23), pp. 313-335, 1992.

[4] L. J. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. E. Tarjan. *Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons.* Algorithmica, (2), pp. 209-233, 1987.

[5] V. Klee. *Is every polygonal region illuminable from some point?* Computational Geometry: Amer. Math. Monthly (76) p. 180, 1969.

[6] D. T. Lee. *Visibility of a simple polygon.* Computer Vision, Graphics, and Image Processing, (22), pp. 207-221, 1983.

[7] G. T. Tokarsky. *Polygonal rooms not illuminable from every point.* American Mathematical Monthly, (102), pp. 867-879, 1995.

Snap Rounding on the Sphere*

Boris Kozorovitzky† Dan Halperin†

Abstract

Snap rounding (SR for short) is a well known method for transforming a planar arrangement of segments given in some arbitrary-precision coordinates into a fixed-precision representation. We extend the method to transforming an arrangement of geodesics on the sphere. We present two approaches for solving the problem. A simple approach of enclosing the sphere in an isocube and projecting the arrangement onto its faces and a more complex approach that makes use of tools from *Discrete Global Grid Systems* (DGGS) to create a better approximation to the sphere. We also generalize the Guibas-Marimont proof of the topological properties preserved by the standard SR for segments in the plane; the generalization is needed for the DGGS approach. Finally, we give rounding results for both methods, obtained with our CGAL (Computational Geometry Algorithms Library) based implementation.

1 Introduction

In computational geometry geometric objects and algorithms are often described assuming infinite precision and exact calculations and predicates. In practice, using infinite precision and exact arithmetic is possible, yet it is often too slow and space consuming to run on real world inputs. Snap rounding is a method for finite-precision approximation for arrangements of segments in the plane: It transforms an arrangement whose segment endpoints' coordinates are given in some arbitrary-precision into a low precision representation.

Given a finite set of segments $\mathcal{S}$, the arrangement $\mathcal{A}(\mathcal{S})$ is the subdivision of the plane into vertices, edges, and faces induced by $\mathcal{S}$. The vertices of $\mathcal{A}(\mathcal{S})$ are either endpoints or intersection points of segments in $\mathcal{S}$. For a given arrangement whose vertices are specified in arbitrary precision snap rounding is the following process. Tile the plane with a grid of unit squares centered at integer coordinates. We refer to each square as a *pixel*. Define a pixel to be *hot* if it contains a vertex of the given arrangement. Replace each vertex by the center of the hot pixel containing it. Replace each original input segment e by a polygonal chain e' going through the centers of the hot pixels intersected by e in the same order the pixels are met by e. Note that in the process, vertices, edges, and faces of the original arrangement may collapse. At the end of the process all the vertices in the snap rounded arrangement are at integer coordinates.

We refer to an original (unrounded) segment and to a resulting polygonal chain as *ursegment* and *polysegment*, respectively [5]. The snap rounded arrangement preserves topological and geometric properties with respect to the original arrangement. **Geometric similarity**— the rounded polysegment e' is within the Minkowski sum of the original ursegment e and a unit square centered at the origin. **Topological similarity**— there is a continuous deformation of the segments in $\mathcal{S}$ to their snap-rounded counterparts such that no segment ever crosses over a vertex of the arrangement.

In this paper we consider a variant of the rounding problem. In our case the input is an arrangement of *geodesics* (arcs of great circles) on a sphere with vertices given in arbitrary precision. Our *spherical snap rounding* (SSR for short) transforms this arrangement into low precision representation, while preserving the following properties: the rounded arcs drift such that the directed Hausdorff distance on the sphere (see formal definition in Section 3.4) between the rounded polysegment and the original arc is no larger than the diameter of the circumcircle of the largest spherical pixel (s-pixel) in the defined grid. The original and the rounded arrangements are topologically equivalent up to the collapsing of features [5]. The point to which we snap the arcs inside the spherical pixels can be represented using small (with respect to bit length) rational values. The spherical pixels have similar shape and area (though not the same) and are fairly regular. Finally, the grid is refinable to allow for increasing the approximation quality.

Related work SR was independently introduced by Hobby [7] and by Greene (unpublished manuscript). The scheme was then generalized by Guibas and Marimont [5] to a dynamic SR algorithm where the authors also introduce some elementary proofs regarding the topological and geometric properties of the snap rounded arrangement. Two additional algorithms by

*This work has been supported in part by the Israel Science Foundation (grant no. 236/06), by the German-Israeli Foundation (grant no. 969/07), and the Hermann Minkowski–Minerva Center for Geometry at Tel Aviv University.

†School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel., boris.kozorovitzky@gmail.com, danha@post.tau.ac.il

Goodrich *et al.* [4] and by de Berg *et al.* [3] give two versions of the SR algorithm, each of which performs better in some cases and worse in others. Later, Hershberger [6] presented an improvement that makes the SR algorithm perform well in all the cases. The subdivision of the sphere that we use is based on known methods in the field of Geodesy and Cartography; Sahr *et al.* [8] show various favorable properties of such subdivisions.

2 Grid Types for which SR Preserves Topology

First, we determine which types of planar pixels (p-pixels) would preserve the topological properties of planar SR; the geometric properties are easier to maintain and prove. Furthermore, we show that for certain grids the choice of *reference point*, namely the point through which the polysegments will snap to in each pixel is immaterial. It has to be, however, the same point inside each pixel. These properties are needed for our usage of the DGGS scheme. For example, Ben-Moshe *et al.* [1] show why the standard SR scheme would not work in the case of triangular tessellations of the plane.

The Deformation Guibas and Marimont [5] look at the SR process as a continuous deformation of the segments from the ursegments to the polysegments. First, the ursegments are divided into *fragments* by adding vertices (nodes) at their intersection points with the boundaries of the hot pixels. A fragment is *external* if it does not intersect the interior of a hot pixel and *internal* otherwise. During the first stage, the hot pixels are contracted in the x-direction toward the center dragging the nodes with the boundary of the pixel until the pixel becomes a hot "stick". In the second stage, the stick is contracted in the y-direction to the center of the pixel. During the deformation, no external fragment crosses into a contracting hot pixel. We extend this observation to a more general setting as we aim to use DGGS grids. Consider a single pixel of a general grid as a convex polygon P. We shrink the hot pixels in a single stage from time $t = 0$ to $t = 1$ by moving the vertices of the pixel at a constant speed (per vertex) toward the reference point. During this deformation process, the pixel is always a homothetic copy of the original pixel and at time $t = 1$ it is reduced to the reference point. We start the discussion with two auxiliary lemmas.

Lemma 1 *Let Q be a convex polygon in the plane. If p is a point in the plane not contained in Q then for every $t \in [0,1]$ it holds that $(p \oplus -tQ) \cap (1-t)Q = \emptyset$.*

The proof of the lemma is a fairly straightforward extension of the well known observation that for two polygons A, B in the plane, $A \cap B \neq \emptyset$ if and only if the Minkowski sum $B \oplus -A$ contains the origin.

Lemma 2 *Let h_1 and h_2 be two shrinking hot pixels and let $s_{\text{frag}}(t)$ be an external fragment with endpoints on the boundaries of h_1 and h_2 at time $0 \leq t \leq 1$. The external fragment $s_{\text{frag}}(t) \subset s_{\text{frag}}(0) \oplus (t \cdot (-P))$, where P is the pixel-polygon with its reference point at the origin.*

We sketch the main idea of the proof. Each fragment endpoint moves with a constant speed toward the reference point. When we take the union of the vectors that represent the movement of all the points on the boundary of P at time t, and move them such that they begin at the origin, we get the polygon $(t \cdot (-P))$ with a reference point that coincides with the origin. The Minkowski sum of $s_{\text{frag}}(0)$ with this polygon is the union of all the segments $s_{\text{frag}}(x), x \in [0,t]$ regardless of where on the boundary of h_1 or h_2 the endpoints of $s_{\text{frag}}(0)$ are.

Theorem 3 *Snap rounding applied to an arrangement of segments with a grid which is a tiling of the plane with identical (in terms of shape and orientation) convex polygonal pixels, maintains the topology preserving property of the SR process for any choice of a fixed reference point inside the pixel.*

Proof. To prove that the topology is maintained we show that during our deformation no external fragment crosses over into a shrinking hot pixel. Without loss of generality, let $s_{\text{frag}}(t) = \overline{s_1 s_2}$ be an external fragment with endpoints on the boundary of the shrinking hot pixels $s_1 \in h_1$ and $s_2 \in h_2$, and let h_3 be another (different) shrinking hot pixel. We know that $s_{\text{frag}}(0)$ does not intersect h_3. Assume to the contrary that $s_{\text{frag}}(t')$ crosses into h_3 and let $t' > 0$ be the first time where $s_{\text{frag}}(t')$ touches its boundary. Assume now, w.l.o.g., that $P := h_3$ contains the origin and that the origin coincides with the reference point. Since $s_{\text{frag}}(0)$ did not cross h_3, it follows from Lemmas 1 and 2 that $s_{\text{frag}}(0) \oplus (t' \cdot (-P))$ is disjoint from $(1-t') \cdot P$, and therefore it is impossible for $s_{\text{frag}}(t')$ to touch the boundary of the shrunk h_3. □

Corollary 4 *Every tiling of the plane with either (i) identical parallelograms or (ii) identical hexagons having parallel opposite edges preserves the topological property of SR , for any selection of a reference point within the boundary of the tile.*

3 Two Approaches to SSR

3.1 The Isocube Approach

Our first approach is quite simple. We enclose the unit sphere in an isocube and project the arrangement of

arcs onto the faces of the cube using Gnomonic projection (G_P). In G_P a point on the sphere is projected to a plane by extending a ray from the center of the sphere through the point. The projected point is the intersection of the ray and the plane (when such exists). The result is six arrangements of segments, one on each face of the isocube. We use the planar SR scheme to round the resulting arrangements to a given pixel size and project the result back onto the sphere using the inverse Gnomonic projection (G_P^{-1}). Now we connect the resulting arrangements by adding *connection arcs* between neighboring hot pixels on different faces of the cube. Some care needs to be exercised when making these connections; we give more details on this step in the next section which explains our second approach.

3.2 The DGGS Approach

In the DGGS approach we inscribe an octahedron in the unit sphere and follow Corollary 4 to define parallelogram pixels on each triangular face. On every face of the octahedron we perform ρ (given as an input for the algorithm) steps of Class I Aperture 4 subdivision [8] creating $4^{\rho-1}$ triangles with two possible orientations o_s— similar to the orientation of the face and o_o— opposite orientation. We select one of the face edges to be inclusive and starting from that edge connect each triangular pixel with orientation o_s to its neighbor (with orientation o_o) thus creating a parallelogram pixel (see Figure 1). For triangles in the last column that do not have a neighbor we create a *phantom* triangle (outside the boundary of the face) with orientation o_o and connect the two triangles. We define the reference point of each parallelogram pixel to be the circumcenter of the triangle with orientation o_s. This selection guarantees that all the reference points of the parallelogram pixels are in the interior of the face of the octahedron and not in the phantom regions. The inclusive region of such a pixel is the part of the parallelogram boundary that was the boundary of the triangle with orientation o_s (right and bottom sides and their joint vertex). We use the parallelograms as p-pixels and create s-pixels by projecting the non phantom parts onto the sphere using the inverse Gnomonic projection.

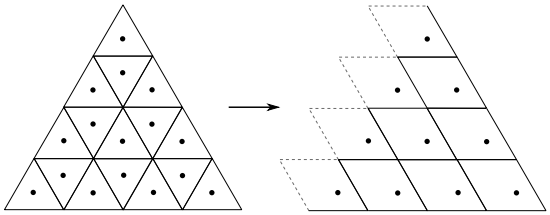


Figure 1: Merging the triangular pixels into parallelogram pixels on a face.

3.3 The DGGS Spherical SR Process

As before, the input is an arrangement $\mathcal{A}(\mathcal{C})$ of geodesic arcs (urarcs) on a unit sphere centered at the origin and an integer ρ. We inscribe an octahedron with vertices at $(\pm1, 0, 0), (0, \pm1, 0), (0, 0, \pm1)$ inside the unit sphere and subdivide each face into parallelogram pixels. We project the arrangement $\mathcal{A}(\mathcal{C})$ onto the faces of the octahedron using Gnomonic projection. A single arc may create a segment over several faces, in this case we consider the segment to be *broken* and the point of the intersection between a segment and an edge of the octahedron as the *break-point* of the segment (a segment can have more than one break-point). We define a p-pixel to be hot if it contains a vertex in the projected arrangement or a break-point. For every break-point we create two vertices infinitesimally close to it, one in each p-pixel, and register the two p-pixels as *boundary-connected* and **hot**. Each vertex of the projected arrangement is replaced by a reference point of the hot p-pixel containing it and each projected segment s is replaced by a polygonal chain going through the hot p-pixels in the same order they are met by s on a single face. After the rounding process, all the rounded segments are projected back onto the sphere using inverse Gnomonic projection. For each registered pair of connected p-pixels, their corresponding s-pixels reference points are connected with a geodesic arc.

3.4 Topological and Geometric Properties of DGGS SSR

We define the distance $d(a, b)$ between the points a and b on the unit sphere as the length of the geodesic between these points and use it in the directed Hausdorff distance on the sphere, $d_H(X, Y) = \max_{x \in X} \left(\min_{y \in Y} (d(x, y)) \right)$.

Lemma 5 *Let f_i be a face of the octahedron, let $\phi_i = G_P^{-1}(f_i)$ and let σ be an unrounded geodesic arc that is contained entirely within ϕ_i. Let ξ_i be the s-pixel with the largest circumcircle crossed by σ and let $\hat{\sigma}$ be the spherically snap rounded polysegment induced by σ. The Hausdorff distance $d_H(\hat{\sigma}, \sigma)$ is no larger than the diameter of the circumcircle of ξ_i.*

Lemma 6 *Let ξ_k and ξ_j be two registered boundary-connected s-pixels and let $p_k = G_P(\xi_k)$ and $p_j = G_P(\xi_j)$ be their corresponding p-pixels. Then, the small geodesic arc connecting the reference points of ξ_k and ξ_j is contained in $G_P^{-1}(p_k) \cup G_P^{-1}(p_j)$.*

Lemma 7 *Let σ be an urarc and $\hat{\sigma} = SSR(\sigma)$ (the spherically snap rounded version of σ) and let ξ_i be*

the s-pixel with the largest circumcircle crossed by σ. The Hausdorff distance $d_H(\hat{\sigma}, \sigma)$ is no larger than the diameter of the circumcircle of ξ_i.

It remains to show that this scheme preserves the topological property as it is defined in the Introduction.

Theorem 8 *Let f_i be a triangular face of the octahedron tiled with parallelograms and let $\mathcal{A}(\mathcal{S})$ be an arrangement of segments contained completely in f_i. Then during the SR process on f_i no vertex crosses over an edge and no new vertices are created.*

Proof. We define the pixels on the face f_i as parallelograms and their reference points as described above. Because the arrangement $\mathcal{A}(\mathcal{S})$ is contained within f_i we can apply Theorem 3 directly to the planar SR scheme. □

Let f_i be a triangular face of the octahedron and g_i the triangle whose corners are the reference points of the corner pixels of that face. We define the *forbidden region* of the face as $f_i^{\text{forb}} = f_i \setminus g_i$.

Observation 1 *After SR on the octahedron face f_i the forbidden region f_i^{forb} does not contain vertices or segments in the rounded arrangement on f_i.*

Lemma 9 *No two connection arcs intersect except at their endpoints.*

Theorem 10 *The original and the rounded arrangement are topologically equivalent up to the collapsing of features.*

Proof. From the properties of planar SR the theorem holds inside a face of the octahedron. The inverse Gnomonic projection does not change topology of features on a single face thus the projected arrangements of arcs conform as well. The projected arrangements are disjoint. We show in Lemma 9 that the connection arcs are disjoint except at the endpoints and we observe that they are disjoint from the projected arrangements because the connection arcs are within the forbidden region. Since the connection arcs do not introduce new intersections the assertion follows. □

4 Implementation

We have implemented both the Isocube and the DGGS algorithms using CGAL[1] and the Arrangement on Surfaces package [2]. We construct a specialized arrangement from the input arcs and split them into smaller arcs such that each arc projects onto a single face of the underlying polytope. During this stage we also record the connection points. Next, we project all the split arcs onto the faces of the polytope and run planar SR on each face. Interestingly, in the DGGS case we can transform the segments and the parallelograms such that the parallelograms become squares. We project the rounded arrangements from each face back onto the unit sphere and add the connection arcs using the previously recorded connection points. Figure 2 shows two examples obtained with our implementation.

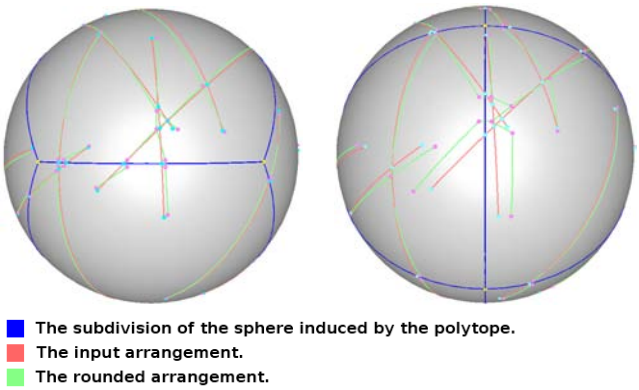


Figure 2: Two examples of a snap rounded arrangement of geodesic arcs. On the left we demonstrate the Isorcube approach and on the right the DGGS approach.

[1] http://www.cgal.org

References

[1] B. Ben-Moshe, B. K. Bhattacharya, and J. Sember. Efficient snap rounding in square and hexagonal grids using integer arithmetic. Technical Report TR-2009-04, The University of British Columbia, 2009.

[2] E. Berberich, E. Fogel, D. Halperin, K. Melhorn, and R. Wein. Sweeping and Maintaining Two-Dimensional Arrangements on Surfaces: A First Step In *Proc. 15th Annu. Eur. Symp. Alg.*, Vol. 4698 of *LNCS*, pages 645–656. Springer-Verlag, 2007.

[3] M. de Berg, D. Halperin, and M. Overmars. An intersection-sensitive algorithm for snap rounding. *Comput. Geom. Theory Appl.*, 36(3):159–165, 2007.

[4] M. T. Goodrich, L. J. Guibas, J. Hershberger, and P. J. Tanenbaum. Snap rounding line segments efficiently in two and three dimensions. In *The 13th ACM Symp. Comput. Geom.*, pages 284–293, 1997.

[5] L. J. Guibas and D. H. Marimont. Rounding arrangements dynamically. In *The 11th ACM Symp. Comput. Geom.*, pages 190–199, 1995.

[6] J. Hershberger. Improved output-sensitive snap rounding. *Discrete & Computational Geometry*, Vol. 39, Num. 1-3 pages 298–318, 2008.

[7] J. D. Hobby. Practical segment intersection with finite precision output. *Comput. Geom. Theory Appl.*, Vol. 13, Num. 4, pages 199–214, 1999.

[8] K. Sahr, D. White, and A. Kimmerling. Geodesic discrete global grid systems. *Cartography and geographic information science.*, Vol. 30, Num. 2, pages 121–134, 2003.

Locating an Obnoxious Line Through a Set of Weighted Points

M. Al-Bow * C. Durso † M.A. Lopez ‡ Y. Mayster §

Abstract

Let $S = \{p_1, \ldots, p_n\}$ be a set of points in the plane with positive weights $w_1, \ldots, w_n$. We find a line ℓ, with at least one point of S on each side, maximizing the weighted Euclidean distance to the nearest point. Our algorithm runs in $O(n^3 \log n)$ time, uses $O(n)$ space, and extends easily to a much broader class of metrics. We also show that the previously proposed $O(n^3)$ solution to this problem is incorrect. Finally, when ℓ is restricted to a small set of possible orientations, an $O(n \log n)$ algorithm is possible.

1 Introduction

The problem of finding a farthest separating line, also known as an obnoxious route, through a set $S = \{p_1, \ldots, p_n\}$ of points in the plane has resulted in algorithms for both the unweighted [4] and the weighted cases [2, 3]. Given S and a set of associated positive weights w_i, we search for a line ℓ, with at least one point from S on each side, that maximizes the distance to S, $\min_{p_i \in S} w_i d(p_i, \ell)$. In the sequel we shall refer to the weighted distance simply as distance and the optimal ℓ as the obnoxious line.

A recent paper [2] considers a more general version of this problem, that of separating polygonal regions, yet the algorithms proposed work for the case of points as well. The ideas in the paper use duality and require $O(n^2)$ space. Furthermore, in order to complete the algorithm for the weighted case parametric search is utilized. In an earlier paper by Drezner and Wesolowsky [3], the authors provide an $O(n^3)$-time and $O(n)$-space algorithm for computing the obnoxious line through a set of weighted points. However, upon careful examination of the proposed algorithm, we found a subtle error that renders the algorithm incorrect. Additionally, the problem cannot be easily corrected, without radically changing the algorithm.

In [3] the authors introduce formulas to compute various quantities, such as the distance to the weighted midpoint of the segment joining two points ($f_{ij}(\Theta)$) and the distance from a point to the line R_L equidistant either from one point on each side or from two points on one side and one on the other ($f_k(\Theta)$). For the actual formulas as well as the specifics of their derivations, the reader is referred to the original paper. Here, $0 \leq \Theta < \pi$ is the angle made by T, the vector perpendicular to R_L, with the positive x-axis. (Other interpretations of Θ lead to inconsistencies in the distance formulas given in the paper.)

The authors outline an algorithm, consisting of two parts, *Procedure 1* and *Procedure 2*. The first computes for any two points i, j the perpendicular route ℓ through the weighted midpoint and checks that no other point is closer to ℓ. It remembers the pair i, j that yields the best such "empty corridor". *Procedure 2* then computes the distance $f_k(\Theta)$ from any other point k to the line R_L equidistant from i, j on one side of it and k on the other. It then classifies all such points k as "below" or "above" (our terminology) depending on whether the quantity $S_k(\Theta)$, the signed Euclidean distance from R_L to k, is positive or negative, respectively, and picks the one closest to its corresponding line for each set (see [3] for details). For each i, j, the algorithm then picks as its candidate for *Procedure 2* the point k which yields the larger of these two minimum distances. The intent is to choose the wider of the two weighted "corridors" on each side of R_L. The algorithm terminates by returning the overall maximum among all candidates picked by both procedures. In doing so, it appears that the authors are using, for the weighted case, properties that only hold for the unweighted case. Their assumption seems to be that, in *Procedure 2*, they don't need to check for "emptiness" of the candidate regions, as they are supposedly ensuring that by picking the k that gives the smaller of the weighted distances on each side. This, while perfectly reasonable for unweighted corridors, fails in the weighted case. Here, for the same pair i, j, we have routes that have different slopes for different points k (unlike the unweighted case!). This means that for different k the weighted corridors may intersect even if they belong to two different sets (i.e., one "above" and one "below") and, as a result, a point k_1 may be inside of another point k_2's "corridor".

This situation is illustrated in Figure 1 and Table 1. We start with points $p_1 = (5.92, 4.92)$, $p_2 = (5.92, -0.44)$, $p_3 = (2.25, 11.03)$, $p_4 = (12.64, 7.96)$, and weights 0.598802, 0.241546, 0.3709, 0.176678, respectively. *Procedure 1* investigates 6 pairs and finds that none of their weighted perpendicular bisectors yield proper corridors. *Procedure 2* then considers each of the 6 pairs in turn on the same side of ℓ

*Computer Science, University of Denver, malbow@du.edu
†Computer Science, University of Denver, cdurso@du.edu
‡Mathematics, University of Denver, mlopez@du.edu
§Computer Science, University of Denver, ymayster@du.edu

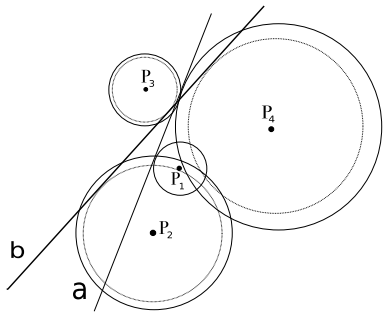


Figure 1: Counterexample to algorithm in [3].

and each of the remaining two points on the other side. This gives a total of 12 candidate separating lines, which are then placed for each pair into either the "above" or the "below" group. These results are tabulated in Table 1 which exposes the flaw in the algorithm: the last column shows the weighted distance f_h from a point h to the line determined by the other three. One can observe that sometimes p_h gets closer to the candidate separator than any of the other three points. This violates the "emptiness" of the candidate, yet if f_k is large enough, it may be picked by the algorithm while the true best separator is skipped as inferior. In our example, the line determined by p_1, p_4 and p_3 is reported as the final answer, yet p_2 is closer to it than the other three points. At the same time, p_2, p_4 and p_3 determine the line, whose f_k is only slightly smaller, but turns out to actually be empty (since p_1 is further away from it than any of these three points) and thus is the true best separator.

Pair	k	sign of $S_k(\theta)$	$f_k(\theta)$	$f_h(\theta)$
$(1,2)$	3	negative	1.59954	0.878394
	4	negative	1.00462	0.779681
$(1,3)$	2	positive	0.534473	1.43659
	4	negative	1.00579	0.904625
$(1,4)$	2	positive	0.918379	2.69556
	3	negative	*1.29187*‡	0.991894
$(2,3)$	1	negative	0.314024	1.39185
	4	negative	0.979299	1.09746
$(2,4)$	1	negative	0.521644	2.72721
	3	negative	*1.26881*†	1.62224
$(3,4)$	1	positive	0.809943	1.45618
	2	positive	1.09385	0.0468722

Table 1: Optimal (†) and Procedure 2 (‡) solutions.

2 An $O(n^3 \log n)$ general algorithm

In this section we describe an algorithm for locating the obnoxious line. It is simple to show that a maximal separating line (in the sense that any sufficiently small perturbation, rotation or translation, leads to worse solution) must be equidistant from at least two points, one in each half-plane. We call these the "witness" points of the separating line. It is perhaps less obvious that the best line, regardless of its orientation, must pass through the weighted midpoint of its witnesses. This property, described in [3], is exploited differently here to synthesize a different algorithm.

For each pair of candidate witnesses p_i, p_j, we find the maximal separating line ℓ_{ij} as the line that passes through the weighted midpoint of p_i and p_j making the least angle with the weighted perpendicular bisector of the segment $\overline{p_i p_j}$ (which, without loss of generality, we assume to coincide with the x-axis) and such that there is no other point p_k closer to ℓ_{ij} than p_i or p_j. We look for a solution to the problem below.

Problem 1 MSTWP: Maximal Separator of Two Weighted Points. *Given a set of weighted points $P = \{p_1 = (x_1, y_1, w_1), \ldots, p_n = (x_n, y_n, w_n)\}$ and two witness points p_i, p_j aligned on a vertical line, find the separator line ℓ_{ij} that maximizes the least distance to P and passes through the weighted midpoint of p_i and p_j.*

Let ϕ be the counterclockwise angle measured from the positive x-axis to a given candidate orientation $\ell_{ij}(\phi)$ of ℓ_{ij}. Clearly, unless emptiness is violated, we would like to keep ℓ_{ij} as the weighted perpendicular bisector itself. Otherwise, we have to exclude all orientations from $[0, \frac{\pi}{2}) \cup (\frac{\pi}{2}, \pi)$ that would result in non-empty "corridors" (i.e., some point of P being closer to $\ell_{ij}(\phi)$ than the witnesses). We then have to find the smallest angle $0 < \phi_1 < \frac{\pi}{2}$ and the largest angle $\frac{\pi}{2} < \phi_2 < \pi$ from those orientations that remained available. The best separator line has the orientation ϕ_1 if $\phi_1 < \pi - \phi_2$ and ϕ_2 otherwise.

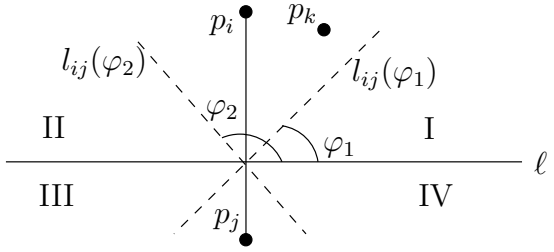


Figure 2: Finding the orientation of the maximal separator. For a pair of witness points p_i, p_j, the angle range $[0, \frac{\pi}{2}) \cup (\frac{\pi}{2}, \pi)$ of possible orientations ϕ is considered and all angles resulting in $\ell_{ij}(\phi)$ closer to some p_k are excluded. Each p_k results in at most two subintervals of angle values removed from further inspection. For p_k in quadrants I and III, we have $0 < \phi_k^s < \phi_k^e < \frac{\pi}{2}$ or $0 < \phi_k^e < \frac{\pi}{2} < \phi_k^s < \pi$, and for p_k in quadrants II and IV, we have $\frac{\pi}{2} < \phi_k^s < \phi_k^e < \pi$ or $0 < \phi_k^e < \frac{\pi}{2} < \phi_k^s < \pi$.

We solve this problem by converting each point $p_k \in P, k \neq i, j$ to at most two intervals of angles ϕ_k that result in p_k being closer to $\ell_{ij}(\phi_k)$ than the two witnesses. In many cases, there is only one interval (ϕ_k^s, ϕ_k^e) of values of ϕ that correspond to such "bad"

separators $\ell_{ij}(\phi)$. However, in the case when $\ell_{ij}(0)$ is closer to p_k than to the witness points, we get two intervals $[0, \phi_k^e), (\phi_k^s, \pi)$ to be excluded from further consideration. In either case, and for a fixed k, all of these angle intervals lie entirely either in $[0, \frac{\pi}{2})$ or in $(\frac{\pi}{2}, \pi)$, since $\ell_{ij}(\frac{\pi}{2})$ passes through both p_i and p_j and is not a separator line at all. The figure above illustrates the process of investigating various candidate orientations ϕ.

Naturally, we have reduced MSTWP to the problem below (since it is the same as finding the best $\phi = \min\{\phi_1, \pi - \phi_2\}$).

Problem 2 LRE: Least Right Endpoint. *Given intervals (a_1, b_1), ..., (a_n, b_n) in $[0, M)$, find the leftmost right endpoint b_i such that $b_i \notin (a_j, b_j) \forall j$.*

We now argue that MSWTP can be solved optimally in $O(n \log n)$ time. (This does not necessarily mean that the original problem has a lower bound of $\Omega(n^3 \log n)$ even though we can solve it with $O(n^2)$ instances of MSWTP.)

Claim 1 *MSWTP can be solved in $\Theta(n \log n)$ time. This is optimal.*

Proof. (By reduction from Connected Union (CU) [1]). Let $(c_1, \ldots, c_n, \epsilon)$ be an instance of CU. We assume that all c_i's and ϵ have already been shifted and scaled to fit inside $[0, \frac{\pi}{2})$, i.e. that $\min_i(c_i - \frac{\epsilon}{2} = 0)$ and $\max_i(c_i + \frac{\epsilon}{2}) < \frac{\pi}{2}$. Let $p_1 = (0, 1)$ and $p_2 = (0, -1)$ be the two "witness" points for the separator line. We set the weights of p_1 and p_2 to be 1 (we shall see that an unweighted version of this problem has the same complexity). Then, any separator line ℓ_{12} passes through the origin. Now, construct the points $p_3, \ldots, p_{n+2}$ one by one based on $c_1, \ldots, c_n$ and ϵ as follows. Let $\phi_{i+2}^s = c_i - \frac{\epsilon}{2}$ and $\phi_{i+2}^e = c_i + \frac{\epsilon}{2}$ be two angles in $[0, \frac{\pi}{2})$. Then, since both $\phi_{i+2}^s < \frac{\pi}{2}$ and $\phi_{i+2}^e < \frac{\pi}{2}$, p_{i+2} is in the first quadrant of Fig. 2 and above the separator line with slope $\tan(\phi_{i+2}^s)$ and below the separator line with slope $\tan(\phi_{i+2}^e)$. So, we can compute the coordinates x_{i+2}, y_{i+2} of p_{i+2} by solving the two equations given by the latter equalities in each of the following

$$\tan(\phi_{i+2}^s) = \frac{w_{i+2}y_{i+2} - w_1 y_1}{w_{i+2}x_{i+2} - w_1 x_1} = \frac{y_{i+2} - 1}{x_{i+2}},$$

$$\tan(\phi_{i+2}^e) = \frac{w_1 y_1 + w_{i+2}y_{i+2}}{w_{i+2}x_{i+2} + w_1 x_1} = \frac{y_{i+2} + 1}{x_{i+2}}.$$

Since the choice of w_i's is immaterial, we fix $w_i = 1$ for all $1 \leq i \leq n + 2$. Then, we end up with two equations in two unknowns (x_{i+2}, y_{i+2}) and can find the coordinates of each $p_i, 3 \leq i \leq n + 2$. We can then find the best separator line ℓ_{12} and if one exists with non-horizontal orientation we know that its slope is not properly inside of the slope ranges for any of the points p_i and, therefore, the union of the intervals from which these points came must be disconnected. Otherwise, we conclude that all angle ranges overlap and, therefore, the union of the open intervals is connected. Hence, since the transformation of the intervals into points takes linear time, we have shown that the lower bound of $\Omega(n \log n)$ carries over from CU to MSWTP.

MSWTP can be solved in $O(n \log n)$ time using a simple counting algorithm. First, sort the set of all angle range endpoints $\{\phi_k^s, \phi_k^e | 1 \leq k \leq n\}$. If $\phi = 0$ is a proper solution (i.e., if no interval has the form $[0, \phi_k^e)$) this is the answer. Otherwise, we process the endpoints, one at a time. Whenever we process a starting (resp. ending) endpoint, we increment (resp. decrement) a counter. (because of the way we have created our intervals, we never encounter an ending endpoint before the corresponding starting endpoint). Now, the first time after processing some $\phi_k^e < \frac{\pi}{2}$ that the counter became 0 we remember that angle as ϕ_1 and the last time that the counter became 0 after processing some $\frac{\pi}{2} < \phi_k^e < \pi$, we remember the solution as ϕ_2. Afterwards, we pick the smaller of ϕ_1 and $\pi - \phi_2$ as the overall best solution. □

3 An $O(kn \log n)$ restricted orientation algorithm

We now turn to the problem of finding the furthest separating line with a prespecified orientation, which we assume to be horizontal. We begin by looking at where in the vertical range of the point set S the best separating line should be placed. Just as in the general case, the goal of maximizing the width of the "corridor" dictates that the line must be centered with respect to the closest points on each side. This means that only two input points $(x_i, y_i), (x_j, y_j)$ with weights w_i, w_j (the witnesses of the previous section) are responsible for determining the location $y = y_s$ of the best separating line s, which must satisfy

$$(y_i - y_s)w_i = (y_s - y_j)w_j \Rightarrow y_s = \frac{y_i w_i + y_j w_j}{w_i + w_j}.$$

Therefore, the solution to the problem is the intersection of the two lines $d^w = -w_i y_s + y_i w_i$ and $d^w = w_j y_s - y_j w_j$, where d^w stands for the weighted distance to the horizontal line at y_s. This leads us to consider a "distance-location" space composed of such lines, each point in S giving rise to one upward and one downward sloping line with the absolute values of the slopes equal to the weight of the point. Let us suppose that all points in S are located in the first quadrant, i.e., $x_i, y_i > 0, \forall 1 \leq i \leq n$ (we can, in linear time, find the right translation to move the origin of the coordinate system). We map each point p_i with the weight w_i to the pair of lines in the "distance-location" plane $\ell_{i0} = w_i y_i - w_i y$ and $\ell_{i1} = -w_i y_i + w_i y$ and restrict the domain to the first quadrant. Thus, for each point we have a linear transformation ℓ_i of the

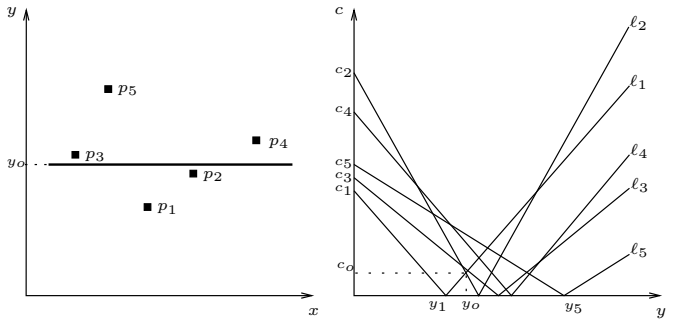


Figure 3: (a) A set of points $p_i = (x_i, y_i)$, sorted by y-coordinate, having weights w_i such that $w_2 > w_4 > w_1 > w_3 > w_5$, and the best fit segment. (b) The corresponding lines in the distance-location plane with the slopes w_i and the vertical axis intercepts c_i. The highest point on the lower boundary (envelope) gives the y-coordinate of the furthest separating line.

absolute value metric function on the nonnegative domain. Each such wedge ℓ_i computes the distance from p_i to the separating line as we hypothetically sweep it upward from $y = 0$, and consists of a finite downsloping segment (recording the distance for $y < y_i$) and an infinite up-sloping ray (for $y > y_i$). In this arrangement of $2n$ lines we are interested in the greatest achievable minimum distance, i.e., in the point with the highest vertical coordinate on the lower envelope of the wedge lines in the distance-location plane. Note that the optimal vertex cannot be found by solving a linear program. Next, we need the following result.

Claim 2 *The lower envelope of a set of n wedges in the distance-location plane has complexity $O(n)$.*

Proof. If we relabel the wedges and consider them sorted by slope, $w_1, \ldots, w_n$, then the wedge with slope w_i, when added to the set of wedges with slopes $w_1, \ldots, w_{i-1}$, only modifies the envelope locally, between the two immediate neighbors of its vertex on the x-axis. Thus, its branches contribute two new edges and three new vertices to the envelope. When the newly added wedge intersects a single edge of the old envelope, it may split the old edge in two, creating one more edge. Therefore, there can be at most $3n-1$ edges (the first wedge adds only two) and $3n-2$ vertices on the lower envelope of n wedges. This is somewhat more tight than what could be obtained from the theory of Davenport-Schinzel sequences.

□

Finding the optimal vertex requires $\Omega(n \log n)$ time. This is true because any solution to the problem of finding the best horizontal separating line takes $\Omega(n \log n)$ time, as can be easily seen by reduction from Max-Gap (see [1]). The argument, of course, carries over to arbitrary fixed orientations of the separating line.

In order to find the highest vertex on the lower envelope of the wedges we employ a simple divide-and-conquer "skyline" merge algorithm. It is optimal since it takes $O(n \log n)$ time. Consequently, given k possible orientations of the separating line, we can find the best separator in $O(kn \log n)$ time.

4 An $O(n^3 \log n)$ algorithm for general norms

The algorithm of Section 2 can be applied to a much broader class of metrics. Provided the metric is derived from a norm, and, as a practical matter, provided the boundaries of the excluded angle intervals in MSTWP can be calculated in constant time, the algorithm can be applied as written. Simply substitute distances in the new metric for Euclidean distances. Among others, the L_p norms, including L_∞, satisfy these conditions.

Below, we provide two claims demonstrating that the properties of the Euclidean metric enabling the algorithm to perform correctly are shared by this broader class of metrics. First, we establish some notation. Let $d(p, q)$ denote the distance between the points p and q in the norm-derived metric under consideration. Denote the distance between a point p and a line ℓ in this metric, $\min_{x \in \ell}\{d(p, x)\}$, by $d(p, \ell)$.

Claim 3 *Let p_1 and p_2 be points in the plane with positive weights w_1 and w_2. Denote by s the weighted midpoint of the segment $\overline{p_1 p_2}$, i.e., the point satisfying $w_1 d(p_1, s) = w_2 d(p_2, s)$. A line ℓ separating $p_1$and p_2 has equal weighted distance to p_1 and p_2, $w_1 d(p_1, \ell) = w_2 d(p_2, \ell)$, if and only if ℓ passes through s.*

Claim 4 *Given a point p and a distance r, there is an interval $[\theta_1, \theta_2]$ such that the line ℓ_θ through $(0, 0)$ and $(\cos\theta, \sin\theta)$ passes within r of p if and only if $\theta + n\pi$ is in $[\theta_1, \theta_2]$ for some integer n.*

These claims, together with the stipulation that the θ_1 and θ_2 mentioned above be available in constant time, allow the obnoxious line problem for $d(\cdot, \cdot)$ to be reduced to MSTWP and then to LRE.

References

[1] Arkin, E., Hurtado, F., Mitchell, J.S.B., Seara, C., Skiena, S.: Some Lower Bounds on Geometric Separability Problems. Int. J. Comp. Geom. and App. **16(1)** (2006) 1-26

[2] Díaz-Báñez, J.M., Ramos, P.A., Sabariego, P.: The Maximin Line Problem with Regional Demand. European J. of Op. Res. **181** (2007) 20-29

[3] Drezner, Z., Wesolowsky, G.O.: Location of an Obnoxious Route. J. of the Op. Res. Soc. **40(11)** (1989) 1011-1018

[4] Houle, M., Maciel, A.: Finding the widest empty corridor through a set of points. In Toussaint, G.T. (ed.): Snapshots of Computational and Discrete Geometry. (1988) 201-213

On Widest Empty Wedges

R. Cardona * M.A. Lopez † Y. Mayster ‡

Abstract

Let S be a set of points in the plane. We solve the problem of finding the widest empty wedge through the interior of the convex hull of S. We consider two cases: when the apex is required to be an extreme point of S and when it can lie anywhere (with some minor restrictions to make the problem well-defined). The first case is solved in $O(n^2)$ time and the second in $O(n^2 \log n)$ time. Both cases require linear space.

1 Introduction

The problem of locating an object of a certain type so as to minimize the distance to a given set of points has been widely studied in the literature. This problem is often called "facility location" or "facility routing," for the cases when the target object is a point or a path, respectively. The types of paths considered for this "minimax" problem include lines, polygonal paths with or without restriction on the number of segments, and many others.

The problem of finding a point or a path that lies as far away as possible from the given set of points is the "maximin" alternative, and is often called the "obnoxious" facility location/routing problem.

The obnoxious location problem has been tackled in several papers [10, 14], with variants involving various error metrics [13] and constraints on the location of the obnoxious facility ([9, 12]), such as restricting it to lie in a given polygonal region [15]. In [11], Cappanera et al. tackle the problem of simultaneous facility location and routing. Different types of obnoxious facilities such as planes [18] or annuli [8] have also been considered. The first paper to solve the widest empty corridor problem (when the obnoxious facility is a line) was [3]. Using duality, Houle and Maciel solved the problem in $O(n^2)$ time. Subsequent papers addressed other variants, such as allowing the corridor to contain up to some k input points [5, 7], weighted distances [2, 1, 4], L-shaped [6] and 1-corner corridors [16, 17]. We take up the problem of finding the widest (obnoxious) empty cone (wedge) through a set S of points. Let S be a set of points and $CH(S)$ its convex hull. We shall describe an algorithm to compute the widest empty wedge anchored anywhere on the boundary of or outside $CH(S)$ and analyze its running time. The wedge (or cone) $\mathcal{W}$ is a convex open polygon, formed by the intersection of two-halfplanes. Therefore, the boundary of the wedge is formed by two rays that meet in a point q, the apex of $\mathcal{W}$. In our problem, it always makes sense to widen the wedge until each of its boundary rays comes to rest on one or more input points. Hence, we shall refer to the boundary rays of $\mathcal{W}$ by the identities of the "supporting" points, the "supports" of the wedge. Further, since the apex of $\mathcal{W}$, for the reasons explained below, is to be fixed on the boundary of $CH(S)$, we shall distinguish between the two supports of $\mathcal{W}$ based on the order in which the two supported edges appear when the wedge is encountered in the counterclockwise traversal of $\partial CH(S)$.

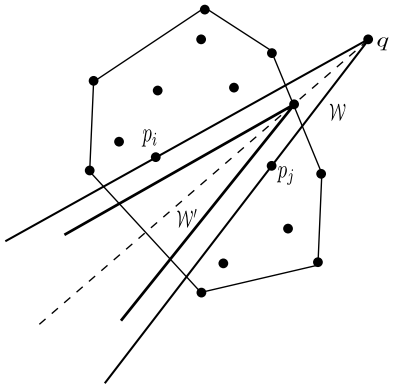


Figure 1: An illustration of the widest empty cone problem.

2 Finding the widest wedge with an apex on the boundary or outside of $CH(S)$

We first describe the algorithm to solve the general problem with the location for the apex unrestricted.

Before we proceed, we observe that it suffices to consider cones with apices on the boundary of $\partial CH(S)$. If one assumes that the best wedge $\mathcal{W}$ rests on an apex outside $CH(S)$, then a quick argument shows that there is a wedge $\mathcal{W}'$ that has a superior width and an apex on $\partial CH(S)$, namely rests on the point q where an edge e of $\partial CH(S)$ intersects the bisector of $\mathcal{W}$. This is shown in Figure 1. Even though attempting to fix the supports of $\mathcal{W}'$ on the same pair of points that support $\mathcal{W}$ may not produce an empty wedge, any wedge with an apex at q that is maximally empty is wider than $\mathcal{W}$ (we can always find support

*Mathematics, University of Denver, rcardon3@du.edu
†Mathematics, University of Denver, mlopez@du.edu
‡Computer Science, University of Denver, ymayster@du.edu

points radially closest from q to the supports for $\mathcal{W}$). Alternatively, keeping the supports of $\mathcal{W}'$ parallel to the supports of $\mathcal{W}$ preserves both the angular width and the emptiness of the wedge. Finally, the case for the apex of $\mathcal{W}$ situated inside of $CH(S)$ does not merit consideration either, because it makes the problem ill-defined. $\mathcal{W}$ can be made arbitrarily wide by moving the apex closer and closer to an edge of $CH(S)$. Hence, we can restrict our attention exclusively to the wedges that have apices on $\partial CH(S)$.

First, we tackle the purely geometric problem of finding the widest wedge with supports on any two points p_i, p_j of S with the anchor q allowed to slide anywhere on a line ℓ. We divide the analysis into two cases.

Case 1: The line ℓ and the line that passes through the segment $\overline{p_ip_j}$ are parallel. In this case we claim that the widest wedge is achieved at the intersection W_{ij} of the line ℓ and the perpendicular bisector of the segment $\overline{p_ip_j}$. To show that this point produces the widest wedge, let's consider the circle Γ that passes through p_i, p_j, W_{ij}. The line ℓ is tangent to Γ because the perpendicular bisector of $\overline{p_ip_j}$ passes through its center and is perpendicular to ℓ. Now consider any point q on ℓ different from W_{ij}. Let q' be the intersection of the segment $\overline{qp_j}$ and the circle Γ. Then we have that $\angle p_iqp_j + \angle qp_iq' = \angle p_iq'p_j = \angle p_iW_{ij}p_j$. The last equation and the fact that $\angle qp_iq' \geq 0$ imply that $\angle p_iW_{ij}p_j \geq \angle p_iqp_j$ for any q on ℓ.

Case 2: The line that passes through p_i and p_j intersects ℓ in the point m_{ij}. We claim that the points on ℓ such that the circle defined by p_i, p_j and one of these points is tangent to ℓ produce the widest possible wedges. The problem of finding these points is known as one of the special cases of the problem of Apollonius. The solution to this problem (refer to Figure 2) is well known and shows that there exist two points that fulfill the conditions. One of the points will be lexicographically to the left of m_{ij} and we will refer to it as $W_{ij}^L(\ell)$, while the point lying to the right of m_{ij} we shall call $W_{ij}^R(\ell)$ (we shall often drop ℓ from these names for brevity when it is implied or irrelevant). Now we prove that for every q on ℓ that is to the left of m_{ij}, $W_{ij}^L(\ell)$ produces the widest wedge.

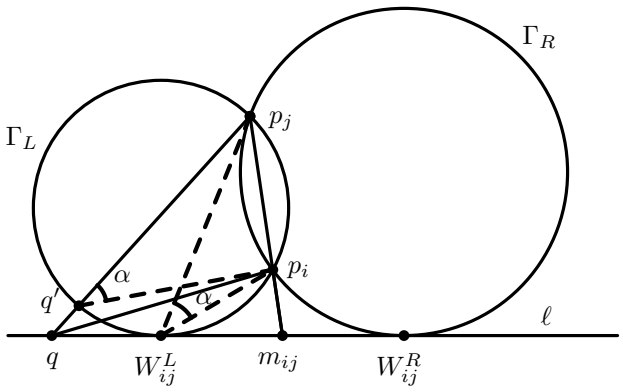


Figure 2: An illustration of the problem of Apollonius.

Consider q to the left of m_{ij}. Let Γ_L be the circle that passes through p_i, p_j and W_{ij}^L. Let q' be the intersection of the segment $\overline{qp_j}$ and the circle Γ_L. Then, we have that $\angle p_iqp_j + \angle qp_iq' = \angle p_iq'p_j = \angle p_iW_{ij}^Lp_j$. Again the result is that $\angle p_iW_{ij}^Lp_j \geq \angle p_iqp_j$ for any q on ℓ to the left of m_{ij}. The argument for W_{ij}^R and q being to the right of m_{ij} is similar.

To determine the position of points W_{ij}^L and W_{ij}^R we use the notion of the power of a point. Let's consider Γ_L as defined previously. The power of m_{ij} with respect to Γ_L can be calculated using points p_i, p_j or using the point of tangency W_{ij}^L. In this case the relation obtained is $m_{ij}p_i \cdot m_{ij}p_j = (m_{ij}W_{ij}^L)^2$ which results in $m_{ij}W_{ij}^L = \sqrt{m_{ij}p_i \cdot m_{ij}p_j}$. Since the coordinates of m_{ij} and the product $m_{ij}p_i \cdot m_{ij}p_j$ are easy to calculate, so is the distance between m_{ij} and W_{ij}^L. The distance from m_{ij} to W_{ij}^R has the same magnitude but opposite sign.

Now, let's define the function $W_{ij}(q) = \angle p_iqp_j, \forall q \in \ell$. We will consider the point q as it is moved from left to right. For the interval $q \in [-\infty, W_{ij}^L]$ the function is increasing. Consider the points q_1, q_2 both of them to the left of W_{ij}^L with q_2 being the closer to it. Let Γ_{L1} be the circle that passes through p_i, p_j and q_2. Γ_{L1} intersects the line ℓ in two points: q_2 and another point to the right of W_{ij}^L. So, the point q_1 is outside of Γ_{L1}. Repeating this geometrical argument, we conclude that $W_{ij}(q_1) = \angle p_iq_1p_j \leq \angle p_iq_2p_j = W_{ij}(q_2)$. Therefore, the function increases as we approach W_{ij}^L from the left. Similar analysis show that $W_{ij}(q)$ decreases on $[W_{ij}^L, m_{ij}] \cup [W_{ij}^R, \infty]$ and increases on $[-\infty, W_{ij}^L] \cup [m_{ij}, W_{ij}^R]$, with W_{ij}^R and W_{ij}^L being the local maxima and m_{ij} being the absolute minimum.

Now, that we have seen how the problem of finding the widest wedge can be solved for a single pair of points and a single line, we are ready to put forth a description of an algorithm for S and restrict the anchors to lie anywhere on $\partial CH(S)$. First, we can see that since $\partial CH(S)$ can have $O(n)$ edges in the worst case, the number of points $W_{ij}^L(\ell), W_{ij}^R(\ell)$ can be $O(n^3)$ (or more precisely, $O(kn^2)$, if $\partial CH(S)$ has k edges). In this setting, we now define $W_{ij}^L(\ell)$ as the local maximum of $W_{ij}(q), q \in \partial CH(S)$ that is encountered first along the supporting line ℓ of an edge e of $\partial CH(S)$ in the counterclockwise traversal of $\partial CH(S)$ (thus, $W_{ij}^R(\ell)$ is the relative maximum that occurs later in such a traversal, and if any of the two points W_{ij}^L, W_{ij}^R are outside of e, it is not considered, yet the naming is still consistent since the direction of traversal along ℓ is well defined).

We shall now see that we do not need to analyze

more than $O(n^2)$ of these points. We are interested only in those wedges that are empty and thus it makes sense to start with a valid configuration of $O(n)$ empty wedges that can be produced with a radial sort on S at some vertex v_1 of $\partial CH(S)$. For p_i, p_j non-parallel to ℓ, we then compute the points $W_{ij}^M(\ell) = W_{ij}^L(\ell)$ (or $W_{ij}^M(\ell) = W_{ij}^R(\ell)$ when $W_{ij}^L(\ell)$ is outside of e, or null if both values are outside) for all pairs of points p_i, p_j adjacent in the radial sort, which provide supports to this initial set of wedges, for the line ℓ through the edge $e = (v_1, v_2)$. Hence, we retain only those $W_{ij}^M(\ell)$'s that lie on the edge e itself, giving initial priority to W_{ij}^L since it is the first to be encountered. For p_i, p_j that are parallel to ℓ, we simply set $W_{ij}^M(\ell)$ to be the point where the perpendicular bisector of p_ip_j intersects e or if it falls on either side of e, the vertex of e closest to that point. We further compute the points $m_{ij}(\ell)$ on e where the changes to the radial order for currently adjacent pairs p_i, p_j take place (for all p_i, p_j such that p_ip_j is not parallel to ℓ). Below we shall describe where this information will be stored for every adjacent pair p_i, p_j.

We observe that the radial order changes when the moving apex q becomes collinear with $K \geq 2$ input points, thereby changing the identities of the supports of the neighboring two wedges (the wedge supported by p_i, p_j does not really change except that the order of the supports is flipped). We make another observation that for each pair p_i, p_j, their relative order in the radial sort as computed from points q along $\partial CH(S)$ can change exactly twice - when q becomes collinear with p_i, p_j at $q = m_{ij}(\ell)$, which happens for exactly two edges e_r, e_s of $\partial CH(S)$, with their supporting lines ℓ_r, ℓ_s (q may become collinear with $K > 2$ points simultaneously with the outcome being the reversal of the order in which the wedges appeared prior to this event, which can be viewed as a sequence of pairwise wedge order reversals). Hence, in total, there are no more than $O(n^2)$ such points $m_{ij}(\ell_r), m_{ij}(\ell_s)$. We, therefore, for each adjacent pair of points in the radial order compute when, if at all, they are "scheduled" to flip along e and put that flip priority (we can represent $m_{ij}(\ell)$ as a single parameter value along (v_1, v_2) if it falls inside, ∞ for those wedges not scheduled to flip along e) in a heap organized in counterclockwise order from v_1. The locations and values for W_{ij}^M can be stored in these nodes as additional data.

The crucial observation here is that the only points where this structure needs to be updated are the points where the radial order changes and the only information that needs to be updated at those points is the information for the flipped wedge and the two neighboring wedges. Since their identities have changed, we need to recompute their W^M's, as well as determine $m_{ij}(\ell)$ for these newly adjacent pairs (or put ∞ as their flip priority if it is to the left of q) and re-heapify on each of these two nodes (we also need to set the flip priority of the wedge causing the change to ∞ and compute $W_{ij}^M = W_{ij}^R$, if W_{ij}^R is inside of e). Finally, the way we keep track of the maximum width $\mathcal{W}$ is by investigating W_{ij}^M (which we precomputed and stored with each wedge as it comes into existence) only those wedges that have been involved in an update at a particular $m_{i'j'}(\ell)$ where the radial order has changed. Since the wedge width function $\mathcal{W}_{ij}(q)$ is semi-monotone between any pair of points on the same side of $m_{ij}(\ell)$ (it contains the single extremum W^L or W^R on such an interval), the wedges that did not participate in the update need not be accounted for at the point of update. Their widths are getting either uniformly wider or narrower anywhere in the vicinity of that point, or they could have achieved a single relative maximum, which has been recorded for these wedges and will be investigated either at some point of update when such an edge is affected or at v_2 if it survives till the end of e. Hence, at the point of update we need to consider W_{ij}^M's for the three wedges in question, if these have been recorded for them, or evaluate them at the point of update itself and maintain the best width so far and the identity of that wedge. We then proceed to move along the edge until v_2, at which point we recompute $W_{ij}^M(\ell')$ for each wedge with respect to the new line ℓ' through the edge (v_2, v_3) and produce a new structure. The cost of this algorithm is $O(kn \log n)$ for the initial constructions at each of the k vertices of $\partial CH(S)$ and $O(n^2 \log n)$ amortized time for the $O(n^2)$ points of update that occur along the entire boundary.

This algorithm always finds the optimal wedge. This is a straightforward consequence of the fact that every wedge that can exist with an apex on $\partial CH(S)$ (and we have already shown that only these wedges need be examined) is, in fact, processed when it comes into existence and no wedge is ever destroyed without the algorithm performing the correct procedure for determining the absolute maximum of $W_{ij}(q)$ by investigating its local maximum W_{ij}^M and evaluting $W_{ij}(q)$ at the endpoints of the interval of existence (which are exactly the points of update to the structure). Hence, the optimal wedge cannot evade detection.

3 Finding the widest wedge with an apex at an extreme point of S

We now consider the case where the apex of the widest empty cone is constrained to coincide with a vertex of $CH(S)$. For clarity, we assume that S is in general position (no two input points share the same x coordinate). We solve this case through recourse to duality, where a point (a, b) becomes the line $ax - b$. Of particular importance is the fact that duality preserves incidence and topological relationships: point P is below line ℓ in primal space iff the dual of ℓ, the point $D(\ell)$, is below the line $D(P)$ in dual space.

We, therefore, convert S to its dual representation, and look at the resulting arrangement of lines $A(S)$. The key to the solution is to describe what an arbitrary empty cone in the primal space that is anchored at a vertex v of $CH(S)$ and with supports at p_i and p_j looks like in the dual. The apex of the cone, v, as well as p_i and p_j, become lines $D(v)$, $D(p_i)$, and $D(p_j)$, respectively. The two boundary rays (or really lines through them) become two points on $D(v)$, call them $D(\ell_i)$ and $D(\ell_j)$. Hence, since the cone contains all rays through v with slopes between those of the rays through p_i and p_j, it becomes a segment on $D(v)$ in the dual, except in the case when it contains a vertical ray (in that case, actually, it's the complement of the interval of slopes between the supports). In fact, since we assumed general position for S, the dual of every point of S intersects $D(v)$. Therefore, what does it mean for the cone at v to be empty? If there is a point p inside of the cone apexed at v, then in the case of that cone not containing a vertical ray (we shall look at that case later), p ends up being above the ray through p_i and below the one through p_j, or vice versa. That means that in the dual space the points $D(\ell_i)$ and $D(\ell_j)$ lie on opposite sides of the line $D(p)$. Therefore, $D(p)$ must intersect the segment connecting $D(\ell_i)$ with $D(\ell_j)$, which we know lies on $D(v)$. Hence, an empty cone in primal space becomes an edge of $A(S)$ in the dual.

For the remaining case, when the empty cone contains a vertical ray, a point inside of the cone is either below or above *both* rays. Furthermore, because the apex is on $CH(S)$ there can only be either points that are below both rays, or above both rays, but never points of each kind simultaneously. Therefore, if such a cone is empty, then all other points of S not lying on the rays of that cone must be above $D(p_i)$ and below $D(p_j)$ or vice versa. This means that the supports of this cone correspond in the dual to the lines that produce the intersections on $D(v)$ that are furthest apart, i.e. the "extreme pair" of points on $D(v)$.

In order to compute all such empty cones, we build $A(S)$ in $O(n^2)$ time with a topological sweep (in order to use $O(n)$ memory). For every edge encountered, we can go back to the primal space and in constant time compare its angular width with the best found so far. The time to examine the candidate cones is $O(n^2)$, i.e., proportional to the size of the arrangement. Finding the extreme pairs also takes at most $O(n^2)$, since we can in linear time find the extreme pair for each candidate apex. (This approach actually allows us to compute the widest empty cone with an apex in S, which may or may not be extreme.)

References

[1] J.M. Díaz-Báñez, P.A. Ramos, P. Sabariego. *The Maximin Line Problem with Regional Demand.* European J. of Operational Research. **181** (2007) 20-29.

[2] Z. Drezner, G.O. Wesolowsky. *Location of an Obnoxious Route.* Journal of the Operational Research Society. **40(11)** (1989) 1011-1018.

[3] M. Houle, A. Maciel. *Finding the Widest Empty Corridor through a Set of Points.* In Toussaint, G.T. (ed.): Snapshots of Computational and Discrete Geometry. (1988) 201-213.

[4] M. Al-bow, C. Durso, M.A. Lopez, Y. Mayster. *Locating an Obnoxious Line Through a Set of Weighted Points in the Plane.* Submitted.

[5] R. Janardan, F. Preparata. *Widest Corridor Problems.* Nordic Journal of Computing. (1994) 231-245.

[6] S.-W. Cheng. *Widest Empty L-Shaped Corridor.* Information Processing Letters. **58(6)** (1996) 277-283.

[7] C.S. Shin, S.Y. Shin, K.-Y. Chwa. *The Widest k-dense Corridor Problems.* Information Processing Letters. **68(1)** (1998) 25-31.

[8] J.M. Díaz-Báñez, F. Hurtado, H. Meijer, D. Rappaport, J. Sellares. *The Largest Empty Annulus Problem.* Proceedings of the International Conference on Computational Science, Part III. Lecture Notes in Computer Science. **2331** (2002) 46-54.

[9] F. Hurtado, V. Sacristan, G. Toussaint. *Some Constrained Minimax and Maximin Location Problems.* Studies in Locational Analysis: Special Issue on Computational Geometry in Locational Analysis. (2000) 17-35.

[10] M. Katz, K. Kedem, M. Segal. *Improved Algorithms for Placing Undesirable Facilities.* Computers and Operations Research. **29(13)** (2002) 1859-1872.

[11] P. Cappanera, G. Gallo, F. Maffioli. *Discrete Facility Location and Routing of Obnoxious Activities.* Discrete Applied Mathematics. **133(1-3)** (2003) 3-28.

[12] B. Ben-Moshe, M. Katz, M. Segal. *Obnoxious Facility Location: Complete Service with Minimal Harm.* International Journal of Computational Geometry and Applications. **10(6)** 2000 581-592.

[13] S. Bespamyatnikh, K. Kedem, M. Segal. *Optimal Facility Location under Various Distance Functions.* In: Proc. 6th International Workshop on Algorithms and Data Structures (WADS'99). (1999) 318329

[14] P. Cappanera. *A Survey on Obnoxious Facility Location Problems.* Tech. Report. Uni. of Pisa. (1999).

[15] P. Bose, Q. Wang. *Facility Location Constrained to a Polygonal Domain.* In: 5th Latin American Symposium on Theoretical Informatics. Lecture Notes in Computer Science. **2286** (2002) 153-164.

[16] J.M. Díaz-Báñez, F. Hurtado. *Computing Obnoxious 1-Corner Polygonal Chains.* Computers and Operations Research. **33(4)** (2006) 1117-1128.

[17] J.M. Díaz-Báñez, M.A. Lopez, J. Sellares. *On Finding a Widest Empty 1-Corner Corridor.* Information Processing Letters. **98(5)** (2006) 199-205.

[18] J.M. Díaz-Báñez, M.A. Lopez, J. Sellares. *Locating an Obnoxious Plane.* European Journal of Operational Research. **173(2)** (2006) 556-564.

Certifying curve-reconstruction algorithms

Asish Mukhopadhyay Harshit Rathod Chong Wang Bryan St. Amour

Abstract

In this paper we propose a novel method for certifying the quality of a curve-reconstruction algorithm. We run any reconstruction algorithm, smoothen the resulting polygonal graph, resample and compare the "distance" betwen this sample and the input sample. We compute this distance using the Hausdorff metric as well as a point pattern-matching algorithm. Experimental results included lend support to our approach.

1 Introduction

The curve reconstruction problem is to reconstruct an unknown curve $\mathcal{C}$ in a given class (for example smooth and closed, or smooth and open etc.) from a sample S of n points $\{p_1, p_2, p_3, \ldots, p_n\}$ (Fig.1). Given the practical importance of the problem, it has been thoroughly researched. The main thrust of some of the recent work has been on reconstructing curves whose correctness are guaranteed provided the sample S satisfies some sampling conditions. We will briefly review some of this work in the next section.

Figure 1: The original Curve, Sample Points and the reconstructed Curve

Unfortunately, we cannot verify that a given sample is an ϵ-sample. In practice we are just given a sample from some unknown curve on which we must carry out a reconstruction. In this paper we propose a novel approach to certifying curve-reconstruction algorithms.

2 Prior work

Many of the ideas surrounding some of the recent Delaunay triangulation-based reconstruction algorithm originate in the work by Brandt and Algazi [3], who showed how to obtain the skeleton of an r-regular shape from the Voronoi diagram of a set of points sampled along the boundary of that shape. The parameter r controls two aspects of such a shape: the curvature at a boundary point never exceeds the reciprocal of r, and the radius of a maximal disk contained in this shape or its complement never exceeds r. Moreover, they were able to show that for shapes in this class the computed skeleton converges to the exact skeleton as the sampling density increases.

Dominique Atali [2] borrowed this notion of an r-regular shape and provided a correct reconstruction for shapes in this class under some guarantee on the sample. Her ideas were further refined by Amenta, Bern and Epstein [1], Dey and Kumar [4], among others.

Guha and Tran [6] proposed a non-Delaunay-based technique that reconstructs a curve in 2 or 3 dimensions from a sample S. Their proposed method determines monotone pieces of the curve, using the idea of bounding curvature.

We are not aware of any work exploiting the ideas we are about to outline.

3 Certification Algorithm

Let $\mathcal{A}$ be any curve-reconstruction algorithm. Our certification algorithm has the following four steps.

Algorithm **CERTIFICATION**

1. Run a reconstruction algorithm $\mathcal{A}$ on the sample S.

2. Smoothen the resulting polygonal reconstruction into a set of curves $\mathcal{C}$.

3. Resample the curves in $\mathcal{C}$ so that we have a sample point from each segment of a curve in $\mathcal{C}$, that corresponds to an edge of the polygonal reconstruction.

   Let S' be the resampled point set.

4. Match the point sets S and S'.

The closeness of the match in the last step is an indication of the accuracy of the curve-reconstruction algorithm.

3.1 Polygonal reconstruction

There are a number of reconstruction algorithms [4, 1] that take an ϵ-sample as input and produce a provably correct reconstruction under suitable restrictions on the parameter ϵ. In this paper, we use our RNG-based reconstruction algorithm as $\mathcal{A}$ [7].

3.2 Smooth the reconstruction

Let us assume that the polygonal reconstruction, $\mathcal{P}$, consists of chains and cycles of varying sizes and isolated vertices. We smoothen $\mathcal{P}$, based on ideas suggested in [5].

The direction of the tangent to the smooth curve that passes through a vertex, p, of degree 2 is set to the direction of the tangent at p to the circumcircle, defined by p and its two neighbours. We fix the smooth curve piecewise for each edge $\overline{pq}$ thus.

1. If both p and q are of degree 1, we retain this edge as part of our smooth curve
2. If p is of degree 1 and q is of degree 2, then the piece of the smooth curve for this part is the part of the circumcircle that is used to define the tangent at q (see Fig 2).

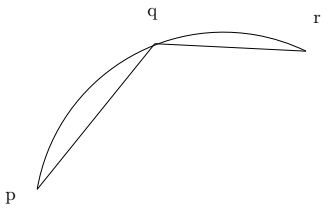


Figure 2: *p is of degree 1 and q is of degree 2*

3. If both p and q are of degree 2, we do this. Let $\overline{up}$ and $\overline{qv}$ be incident on p and q respectively. Two cases arise:
   - If u and v are on opposite sides of the supporting line of $\overline{pq}$ as in Fig. 3 below, the smooth curve through $\overline{pq}$ consists of 4 subpieces that are joined together to form a single piece. In the subpiece px, the circle section satisfies the tangent constraint at p and the tangent at x (1/4 location of $\overline{pq}$) is parallel to $\overline{pq}$. Similarly, in the subpiece qy, the circle section satisfies the tangent constraint at q and the tangent at y (3/4 location of $\overline{pq}$) is parallel to $\overline{pq}$. For the two middle subpieces, w is the midpoint of $\overline{xy}$. $\overline{zc2}$ is the perpendicular bisector of $\overline{xw}$, while $\overline{sc3}$ is the perpendicular bisector of $\overline{wy}$.
   - If u and v are on the same side of the supporting line of $\overline{pq}$ as in Fig. 4 below, the smooth curve through $\overline{pq}$ consists of 2 subpieces that are joined together to form a single piece. $\overline{Kp}$ is the angle bisector of $\angle mpq$ and $\overline{Kq}$ is the angle bisector of $\angle nqp$. The tangent b at K is perpendicular to line d. $\overline{pc1}$ is perpendicular to $\overline{um}$ and $\overline{qc2}$ is perpendicular to $\overline{vn}$.

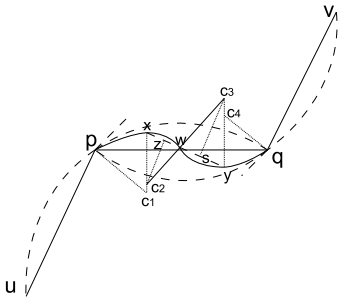


Figure 3: *p and q are of degree 2 and the neighbors are on opposite sides*

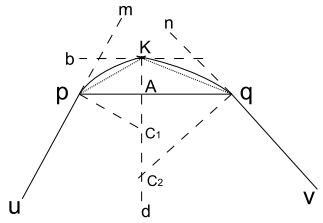


Figure 4: *p and q are of degree 2 and the neighbors are on the same side*

3.3 Sampling the smoothened curve

We sample the smooth curves in $\mathcal{C}$ constructed in the last section by choosing a random point from each section of a curve in $\mathcal{C}$ that corresponds to an edge in the polygonal reconstruction.

In the experiments that we report in a subsequent section, when we consider the output of our RNG-reconstruction before removal of the non-curve adjacent edges, we add to this set of sample points a randomly chosen point from each non-curve adjacent edge.

3.4 Matching the two samples

We discuss two measures for quantifying the "distance" between S and the sample obtained from $\mathcal{C}$. We contend that the accuracy of the reconstruction

algorithm $\mathcal{A}$ is reflected by this "distance".

The first measure is the Hausdorff distance between the two samples; the second is obtained by using a point matching algorithm due to Murtagh [8]. The details are in the next two subsections.

3.5 Hausdorff distance

The Hausdorff distance between two non-empty subsets X and Y of a metric space (M, d) is defined thus:

$$D_H(X,Y) = \max \{sup_{x\in X}\ inf_{y\in Y} d(x,y), \\ sup_{y\in Y}\ inf_{x\in X} d(x,y)\}$$

We measured $D_H(X,Y)$ for 10 different samples, in pixel units, to test our RNG-algorithm [7]. The entries in the second and third columns of Table 1, are respectively the values of $D_H(X,Y)$ prior to and after the removal of non-curve adjacent edges.

Sample	without Remove-edges	Remove-edges
S1	18.00	13.99
S2	32.38	11.24
S3	27.31	10.62
S4	42.30	11.06
S5	35.60	10.95
S6	20.30	13.97
S7	39.82	11.72
S8	39.61	12.32
S9	29.15	16.28
S10	119.05	13.09

Table 1: *Comparing Hausdorff distances*

Since the reconstruction can be construed as being poor prior to the removal of the non curve-adjacent edges the entries in the third column are smaller than those in the second.

3.6 Point matching

Our second method of estimating the closeness of the two samples is based on a a slightly modified version of Murtagh's algorithm [8].

We first compute the centroid (x_0, y_0) of $S \cup S'$. We then sort the two point sets about (x_0, y_0) by increasing polar angle with respect to the x-axis.

Next, we join the points of S, consecutive in sorted order to create a star-polygon and compute the intersections of this star-polygon with 360 rays, spaced 1^0 apart, anchored at the centroid. The distances of these intersection points from the centroid, normalized with respect to the maximum distance, are used to create a 360-element vector, call it VS. We follow exactly the same steps on the set S' to obtain the vector VS'.

In the third step we compute the maximum angle θ_{max} between two points of S, successive in sorted order and compute the Euclidean distance between VS and θ_{max} left and right rotations of VS', returning the minimum of these distances.

We ran the above algorithm on the same set of 10 samples we used for the Hausdorff metric. The results are summarized in Table 2 below.

Sample	without Remove-edges	Remove-edges
S1	4.39	4.46
S2	0.69	0.30
S3	1.061	1.060
S4	2.92	3.03
S5	3.05	3.02
S6	2.25	2.23
S7	1.47	0.74
S8	0.94	0.86
S9	1.63	0.85
S10	0.76	0.17

Table 2: *Comparing outputs of Murtagh's Algorithm*

4 Conclusions

The Hausdorff distance metric seems to work quite well. However, the same can't be said of the distance metric based on Murtagh's point matching algorithm. We have not quite understood why the entries in the third column of Table 2 are not consistently smaller than the entries in the second column, unlike what we had expected. We are looking into this problem.

Figure 5: Sample 1, without Remove-edges, with Remove-edges

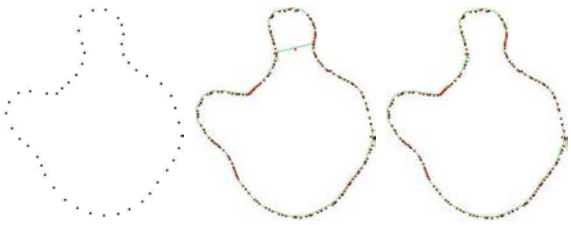


Figure 6: Sample 2, without Remove-edges, with Remove-edges

Figure 7: Sample 3, without Remove-edges, with Remove-edges

Figure 8: Sample 4, without Remove-edges, with Remove-edges

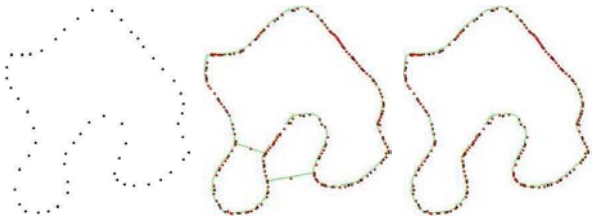


Figure 9: Sample 5, without Remove-edges, with Remove-edges

Figure 10: Sample 6, without Remove-edges, with Remove-edges

Figure 11: Sample 7, without Remove-edges, with Remove-edges

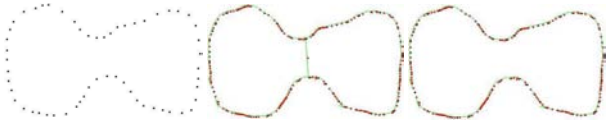


Figure 12: Sample 8, without Remove-edges, with Remove-edges

Figure 13: Sample 9, without Remove-edges, with Remove-edges

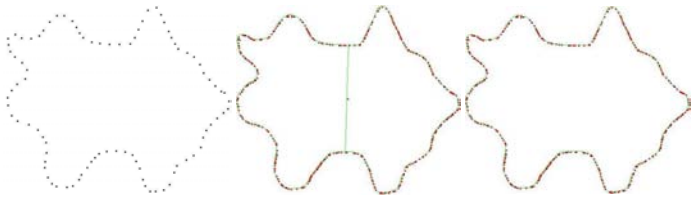


Figure 14: Sample 10, without Remove-edges, with Remove-edges

References

[1] N. Amenta, M. Bern, and D. Eppstein. The crust and the β-skeleton: Combinatorial curve reconst ruction. In *Graphical Models and Image processing*, volume 60, pages 125–135, 1998.

[2] D. Attali. r-regular shape reconstruction from unorganized points. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 248–253, 1997.

[3] J. W. Brandt and V. R. Algazi. Continuous skeleton computation by Voronoi diagram. *CVGIP: Image Understanding*, 55(3):329–338, 1992.

[4] T. Dey and P. Kumar. A simple provable algorithm for curve reconstruction. In *ACM-SIAM Symposium Discr. Algorithms*, pages 893–894, 1999.

[5] T. Dey, K. Mehlhorn, and E. A. Ramos. Curve reconstruction : connecting dots with good reason. In *Proc. 15th Annual ACM Sympos. Comput. Geom.*, pages 197–206, 1999.

[6] S. Guha and Son Dinh Tran. Reconstructing curves without delaunay computation. *Algorithmica*, 42:75–94, 2005.

[7] Asish Mukhopadhyay and Augustus Das. An RNG-based heuristic for curve reconstruction. *International Symposium on Voronoi Diagrams in Science and Engineering*, 0:246–251, 2006.

[8] F. Murtagh. A feature-based $O(n^2)$ approach to point pattern matching. In *International Conference on Pattern Recognition*, pages II:174–177, 1992.

Circles with Independent and Dependent Uncertainties

Yonatan Myers and Leo Joskowicz *

Abstract

Classical computational geometry algorithms handle geometric constructs whose shapes and locations are exact. However, many real-world applications require modeling and computing with geometric uncertainties, which are often coupled and mutually dependent. We have developed the Linear Parametric Geometric Uncertainty Model (LPGUM), a general and computationally efficient worst-case first-order linear approximation of geometric uncertainty that supports dependencies among uncertainties. Previous papers describe LPGUM models for points and lines, and algorithms for efficiently solving relative positioning and point distance problems. In this paper we present two LPGUM models of a circle defined by an uncertain center point and an uncertain vector, and by three uncertain points. We describe the geometric properties of their uncertainty zone envelopes and present efficient algorithms to compute them.

1 Introduction

Geometric uncertainty is ubiquitous in mechanical CAD/CAM, robotics, and many other fields. Sensing, measuring, and manufacturing processes are intrinsically imprecise, thereby introducing error and uncertainty. In contrast, the corresponding geometric models are usually exact and do not account for these inaccuracies. Modeling and computing geometric variability is thus of great practical importance.

Numerous frameworks have been proposed for modeling and computing with geometric imprecision [1, 3, 8, 9]. A common approach is to use simple geometric entities, such as rectangles [10, 11], circles [2, 5], or convex polygons [4] to bound point coordinates variations. Efficient algorithms for common problems, such as finding the smallest/largest enclosing circle/convex hull of a set of independent uncertain points, have been developed [9]. The key drawback of these models is that they cannot model mutually dependent uncertainties, which are very common in practice [14]. Assuming independent variations or errors often overestimates the actual geometric uncertainty.

Geometric uncertainty has also been studied in an algebraic framework. Interval arithmetic shows how to represent and propagate real-valued numbers uncertainty intervals [15]. Since dependencies cannot be modeled, this often yields overestimated uncertainty intervals. Affine arithmetic [6] improves interval estimation by tracking round-off and truncation errors, and allows for quantities interdependence. However, it does not provide an explicit description of the geometric uncertainty. Other techniques such as robust, finite precision, and epsilon geometry are only applicable for very small variations [7, 17].

We have introduced the Linear Parametric Geometric Uncertainty Model (LPGUM) [8], a general, expressive, and computationally efficient worst-case first-order linear approximation of geometric uncertainty that allows for coupling between uncertainties. Geometric objects are defined by joint parameters with uncertainty intervals. The uncertainty zones around nominal objects are defined by uncertainty sensitivity matrices whose entries indicate their sensitivity to parameters variations. We have developed efficient algorithms for computing uncertainty zones of points and lines [8], for relative positioning queries [12], and for point distance problems [13].

In this paper we introduce two LPGUM models of a circle defined by an uncertain center point and an uncertain vector, and by three uncertain points. We describe the geometric properties of their uncertainty zone envelopes and present efficient algorithms to compute them for the independent/dependent cases.

2 The Linear Parametric Geometric Uncertainty Model

A parametric uncertainty model $(q, \bar{q}, \Delta)$ is defined as follows. Let $q = [q_1, q_2, ..., q_k]^{\mathrm{T}}$ be a vector of k parameters over an *uncertainty domain* Δ. Each parameter q_j can take a value from an *uncertainty interval* $\Delta_j = [q_j^-, q_j^+]$, $(\Delta_j \subset \mathbb{R})$ and is associated with a nominal value $\bar{q}_i \in \mathbb{R}$. The parameters' uncertainty domain $\Delta = \Delta_1 \times \Delta_2 \ldots \times \Delta_k$ is the product of the parameters uncertainty intervals. The *nominal parameters vector* $\bar{q} = (\bar{q}_1, ..., \bar{q}_k)$ is the parameters vector values with no uncertainty. WLOG, we assume that the uncertainty intervals are zero-centered symmetric, i.e., $-q_j^- = q_j^+$ (asymmetric domains are transformed by adjusting the nominal parameter value and interval).

An uncertain dimension $d(q)$ is defined by a nominal value $\bar{d}$ and a k-dimensional *uncertainty sensitiv-*

*School of Engineering and Computer Science, The Hebrew University of Jerusalem, ISRAEL. Emails: yoni_m@cs.huji.ac.il, josko@cs.huji.ac.il

ity vector A_d. Entry $(A_d)_i$ is a constant that quantifies the sensitivity of the dimension to parameter q_i. It is zero when the dimension is independent of parameter q_i. The LPGUM of dimension $d(q)$ is $d(q) = \bar{d} + A_d q$. Its uncertainty zone is the set of all dimension values for instances of parameters vector q, $\mathcal{Z}(d) = \{d \mid d = \bar{d} + A_d q,\ q \in \Delta\}$. It is a closed interval whose bounds are computed in optimal $O(k)$.

An uncertain point $v(q)$ is defined by a nominal location $\bar{v}$ and a $2 \times k$ *uncertainty sensitivity matrix* A_v. Entry $(A_v)_{i,j}$ is a constant that quantifies the sensitivity of coordinate i to parameter q_j ($i = 1$ for x, $i = 2$ for y). It is zero when coordinate i is independent of parameter q_j; When the entire column $(A_v)_j$ is zero, the point $v(q)$ is independent of parameter q_j.

The LPGUM of point $v(q)$ is $v(q) = \bar{v} + A_v q$. Its uncertainty zone is the set of all point locations for instances of parameters vector q, $\mathcal{Z}(v) = \{v \mid v = \bar{v} + A_v q,\ q \in \Delta\}$. It is a zonotope (a centrally symmetric convex polygon) with at most $2k$ vertices (e.g., the center point in Fig. 1) [14]. LPGUM vectors are defined identically.

The uncertainty zone of point $v(q)$ is the feasible region of the linear programming problem:

$$\max_q \langle A_v^\top b, q \rangle \quad \text{subject to } q \in \Delta \tag{1}$$

where $\langle \cdot, \cdot \rangle$ is the vector inner product. The k-dimensional vector that maximizes Eq. (1) in direction b, called the *sign vector*, is formed by the maximum of each parameter q_i: "+" for q_i^+, "−" for q_i^-.

The point uncertainty envelope vertices are computed from its *cone diagram*. Each sensitivity matrix column vector induces a line normal to it. The lines all intersect at the origin and are sorted in increasing angle order. They form a planar subdivision whose cells are cones, each bound by the two lines defined by their corresponding column vectors. For every cone, we find the parameter value that changes when crossing the line to the adjacent cone, from q_i^+ to q_i^- or vice-versa, and compute the corresponding envelope vertex. The resulting zonotope is thus computed in optimal $O(k \log k)$ time and $O(k)$ space [14].

3 LPGUM circle definitions

A circle can be defined in one of four ways:

1. *Center-radius*: a center point $o(q)$ and radius $r(q)$
2. *Center-point*: a center point $o(q)$ and a circumference point $u(q)$
3. *Antipodal-points*: two antipodal circumference points $u(q)$, $v(q)$
4. *Three-points*: three circumference points $u(q)$, $v(q)$, $w(q)$

where $r(q)$ is an LPGUM dimension and $o(q)$,$u(q)$, $v(q)$ and $w(q)$ are LPGUM points over parametric uncertainty model $(q, \bar{q}, \Delta)$. Since they are defined with the same parametric uncertainty model, the parameters uncertainties can be independent or dependent, based on the sensitivity matrices entries.

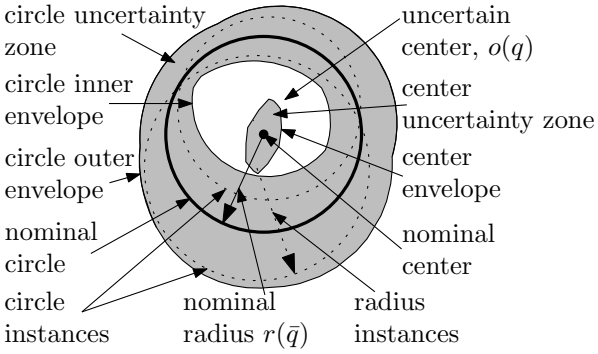


Figure 1: Illustration of an uncertain circle defined by an uncertain center point and an uncertain radius. Their uncertainty zones are shown in grey.

In the nominal case ($q = 0$), all the above representations are equivalent. In the LPGUM model, it can be shown that the first three representations are equivalent, while the last one is different.

Definition 1: An *uncertain center-vector circle* $c(q)$ is defined by a nominal center point $\bar{o}$, a nominal radius vector $\bar{v}$, and their uncertainty sensitivity matrices, A_o and A_v over parametric uncertainty model $(q, \bar{q}, \Delta)$. The circle center is $o(q) = \bar{v} + A_o q$ and the radius vector is $v(q) = \bar{v} + A_v q$. The center-vector LPGUM circle is $c(q) = o(q) + R(\theta) v(q)$, where

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

and θ is an auxiliary parameter. Its uncertainty zone is the set of all circles $c(q)$ for instances of q:

$$\mathcal{Z}(c) = \{c \mid c = \bar{o} + A_o q + R(\theta)(\bar{v} + A_v q), \quad q \in \Delta,\ \theta \in [0, 2\pi]\} \tag{2}$$

Definition 2: An *uncertain three-point circle* $c(q)$ is defined by three nominal points $\bar{u}, \bar{v}, \bar{w}$ and their respective uncertainty sensitivity matrices A_u, A_v, A_w over parametric uncertainty model $(q, \bar{q}, \Delta)$. The points are $u(q) = \bar{u} + A_u q$, $v(q) = \bar{v} + A_v q$, and $w(q) = \bar{w} + A_w q$. The three-point LPGUM circle $c(q)$ is computed by the well known three-point circle method $circle(u(q), v(q), w(q))$ for a given instance of parameter vector q. Its uncertainty zone is the set of all circles $c(q)$ for instances of parameter vector q:

$$\mathcal{Z}(c) = \{c \mid c = circle(u(q), v(q), w(q)),\ q \in \Delta\} \tag{3}$$

4 Circle uncertainty zone computation

We distinguish four cases for the LPGUM circle uncertainty zone computation: center-vector and three-

point LPGUM circle, with independent or dependent uncertainties. Fig. 1 shows a circle uncertainty zone.

4.1 Center-vector LPGUM circle

The center-vector LPGUM circle uncertainty zone is an annulus-like area enclosed by an outer envelope and a possibly empty inner envelope, each formed by straight line segments and circular arcs. The outer envelope is the union of all circle instances; the inner envelope is the intersection of all circle instances, and is thus convex. The inner and outer envelopes have at most $O(k^2)$ segments (possibly not a tight bound).

1. Independent uncertainties

When the center point and vector uncertainties are independent, the circle uncertainty zone is computed from the center point zonotope and the annulus whose center is the circle origin. The annulus a is defined by the minimum and maximum circle radiuses, $r_{\min} = \min_q \|v(q)\|$ and $r_{\max} = \max_q \|v(q)\|$ for $q \in \Delta$. The outer and inner envelopes are the Minkowski sum of the zonotope $o(q)$, computed in $O(k \log k)$, and the annulus a, computed in $O(k)$. The outer envelope is computed in $O(k)$ time by offseting the edges of $o(q)$ outwards by r_{max} and connecting the resulting edge segments with arcs of radius r_{max} centered at vertices of $o(q)$. The inner envelope is the boundary of the intersection of k discs of radius r_{min} centered at the vertices of $o(q)$. When the diameter of $o(q)$ is greater than $2r_{min}$ the inner envelope is empty. Otherwise, it contains $\bar{o}$ and is computed by intersecting the circles in $O(k \log^2 k)$ [16], which dominates the computation time complexity.

2. Dependent uncertainties

When the center point and vector are dependent, we use a sweep algorithm similar to that of the LPGUM line uncertainty envelope computation [12]. We perform an angular sweep of θ starting at $\theta = 0$ in Eq. (2). Each θ value yields a zonotope $c_\theta(q)$. As θ changes, $c_\theta(q)$ traces out the circle envelope. The values of θ for which the topology of the cone diagram of $c_\theta(q)$ changes correspond to *events*. Starting from the initial zonotope $c_0(q)$, we sweep the zonotope from its previous location to the current one and compute new events. When the event queue is empty, we sweep the last zonotope to the initial one, $c_0(q)$. and combine all the swept parts to obtain the envelope.

There are two types of events:

1. *Switch events* occur at values of θ where two lines of the cone diagram of $c_\theta(q)$ coincide. These events correspond to the the values of θ where two columns of $c_\theta(q)$ are linearly dependent:

$$\begin{vmatrix} (A_o)_{ix} + (R(\theta)A_v)_{ix} & (A_o)_{iy} + (R(\theta)A_v)_{iy} \\ (A_o)_{jx} + (R(\theta)A_v)_{jx} & (A_o)_{jy} + (R(\theta)A_v)_{jy} \end{vmatrix} = 0$$

that is, when $a\cos(\theta) + b\sin(\theta) + c = 0$ where

$$\begin{aligned} a &= \begin{vmatrix} (A_v)_{ix} & (A_v)_{iy} \\ (A_o)_{jx} & (A_o)_{jy} \end{vmatrix} + \begin{vmatrix} (A_o)_{ix} & (A_o)_{iy} \\ (A_v)_{jx} & (A_v)_{jy} \end{vmatrix} \\ b &= (A_o)_i \cdot (A_v)_j - (A_o)_j \cdot (A_v)_i \\ c &= \begin{vmatrix} (A_o)_{ix} & (A_o)_{iy} \\ (A_o)_{jx} & (A_o)_{jy} \end{vmatrix} + \begin{vmatrix} (A_v)_{ix} & (A_v)_{iy} \\ (A_v)_{jx} & (A_v)_{jy} \end{vmatrix} \end{aligned}$$

This trigonometric equation in one unknown can have zero, one, or two solutions. When $a = b = c = 0$, there are infinitely many solutions, but no event can occur, as the is no change in the cone diagram topology.

2. *Flip events* occur at values of θ for which column vector $(A_\theta)_i$ is zero, that is, when $\|(A_o)_i\| = \|(A_v)_i\|$. In this case, $(\theta = \pi - \alpha)$ where $\alpha \in (-\pi, \pi)$ is the angle between $(A_o)_i$ and $(A_v)_i$.

To move the zonotope from one event to the next, we compute the circle $c(q)$ for every point on the boundary of the first zonotope and connect the resulting zonotopes with arc segments on the circle between the event angles. To compute this zonotope edge sweep, we compute the circle segments only for edge vertices. When the tangent to the two circles in the interval is valid, it is added.

Next, we define the inner and outer boundary of the swept area. If the swept area contains a tangent to the circular segments, the outer part consists of the tangent segment and the two arcs leading from the end of the tangent to the end of the zonotopes edges. The rest of the boundary is the inner part. If there is no tangent, the outer part is the arc that bounds the area so that the swept area is on the inner side of the arc and the inner part is the other arc. The outer and inner parts of the swept areas are collected in separate sets. When no more events are left, the outer and inner envelopes are computed from their corresponding sets.

We show that the circle uncertainty zone envelope complexity is $O(k^2)$ as follows. The outer envelope is the unbounded cell of an arrangement of arcs and line segments (elements). Each event can add at most three elements to the arrangement. Since there are at most k flip events and $2k^2$ switch events, the arrangement has $O(k^2)$ elements, and any pair of elements can intersect at most twice. The cell complexity in this arrangement can be shown to be $O(\lambda_4(n)) = O(n \cdot 2^{\alpha(n)})$ where $\lambda_s(n)$ is the length of the Davenport-Schinzel sequence $DS(n, s)$ [16]. Thus, the envelopes complexity is $O(k^2 \cdot 2^{\alpha(k^2)}) \approx O(k^2)$.

The algorithm time complexity is dominated by the computation of the outer and inner cells of two arrangements. As there are $O(k^2)$ events and every zonotope edge is swept at every event, each of the sets has at most $O(k^3)$ arcs. The arrangement can be computed with a standard sweepline algorithm in $O(k^6 \log k)$. When a point in the interior of each cell to be computed is known, the time is reduced to $O(\lambda_4(n) \log^2(n))$. Since $n = k^3$, $O(\lambda_4(k^3) \log^2(k^3)) \approx O(k^3 \log^2(k^3)) = O(k^3 \log^2 k)$.

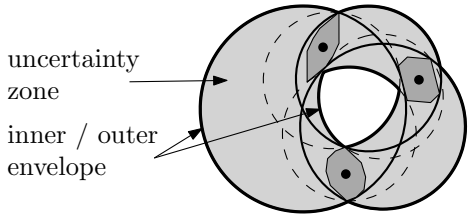


Figure 2: Uncertainty zone of a three-point LPGUM circle for independent uncertainties. Circles supporting the outer envelope are solid; circles supporting the inner envelope are dashed. LPGUM points are dark gray; the circle's uncertainty zone is light gray.

4.2 Three-point LPGUM circle

The three-point LPGUM circle uncertainty zone has the same annulus-like structure of the center-vector circle: the outer envelope is the union of all circle instances; the inner envelope is the intersection of all circle instances, and is thus convex. However, they are different, as the center point of the three-point circle is not necessarily an LPGUM point.

We describe next the properties of the three-point LPGUM circle uncertainty zone and its computation for independent uncertainties. Let $u(q), v(q), w(q)$ be three independent LPGUM points such that for every $q \in \Delta$, $w(q)$ lies to the right of the directed line through $u(q), v(q)$. The circle's outer and inner envelopes each consist of three arcs (Figure 2). To compute the envelopes, we find the circles supporting the arcs. For the outer envelope, we find three circles each tangent to the three LPGUM points. Every circle has two of the points in its interior and one point to its exterior. The outer envelope is the outer envelope of the arrangement of the three circles. To compute the inner envelope, we find three circles tangent to the LPGUM points but, now two points are exterior to the circle and one is interior. The inner envelope is the boundary of the area at the intersection of the three circles. Both envelopes can be computed in $O(k)$ optimal time.

The three-point LPGUM circle with dependent uncertainties is currently an open problem.

5 Conclusion

We have presented a parametric, first-order model of the geometric uncertainty of a circle defined by a point and a vector, and by three points in the plane. We derived the properties of the circle uncertainty zones and developed efficient algorithms to compute them for independent and dependent uncertainties. These new circle results add to the expressiveness and generality of LPGUM and provide a model for the accurate estimation of circle uncertainties.

Future work includes addressing the open problems in this paper, tightening the time and geometric complexity bounds, and exploring other problems such as LPGUM convex hulls and Voronoi diagrams.

References

[1] M. Abellanas, F. Hurtado, and P. A. Ramos. Structural tolerance and Delaunay triangulation. *Information Processing Letters* 71(5-6):221–227, 1999.

[2] S. Akella and M. Mason. Orienting toleranced polygonal parts. *Int. J. Robotics Research* 19(12), 2000.

[3] I. Averbakh, S. Bereg. Facility location problems with uncertainty on the plane. *Discrete Optim.* 2(1), 2005.

[4] R. Brost, and R.R. Peters, Automatic design of 3D fixtures and assembly pallets. *Proc. IEEE Int. Conf. Robotics and Automation*, Minneapolis, USA, 1996.

[5] J. Chen, K. Goldberg, M. Overmars, D. Halperin, K-F. Böhringer, and Y. Zhuang. Computing tolerance parameters for fixturing and feeding. *The Assembly Automation Journal* 22:163–172, 2002.

[6] L.H. de Figueiredo and J. Stolfi, Affine Arithmetic: concepts and applications. *Numerical Algorithms* 37:147–158, 2004.

[7] D. Salesin, J. Stolfi, and L. Guibas. Epsilon Geometry: building robust algorithms from imprecise computations. *Proc. ACM Symp. on Comp. Geom.*, 1989.

[8] L. Joskowicz, Y. Ostrovsky-Berman, and Y. Myers. Efficient representation and computation of geometric uncertainty: the linear parametric model. *Precision Engineering*, May 2009.

[9] M. Löffler and M. van Kreveld. Largest bounding box, smallest diameter, and related problems on imprecise points. *Algorithms and Data Structures*, *Lecture Notes in Computer Science*, Springer Berlin, Vol 4619, 2007.

[10] M. Löffler and M. van Kreveld. Largest and smallest convex hulls for imprecise points. *Algorithmica*, Springer, New York, 2009.

[11] A. A. Khanban. Basic algorithms of computational geometry with imprecise input.
PhD dissertation, Univ. of London, UK, 2005.

[12] Y. Myers and L. Joskowicz. The linear parametric geometric uncertainty model: Points, lines and their relative positioning. *24th Europ. Workshop on Computational Geometry*, pp 137–140, France, March 2008.

[13] Y. Myers, and L. Joskowicz. Point distance problems with dependent uncertainties. *25th Europ. Workshop on Computational Geometry*, Belgium, March 2009.

[14] Y. Ostrovsky-Berman and L. Joskowicz. Tolerance envelopes of planar mechanical parts with parametric tolerances. *Comp. Aided Design*, 37(5):531–544, 2005.

[15] M. Segal. Using tolerances to guarantee valid polyhedral modeling results. *SIGGRAPH Computer Graphics* 24(4):105–114, 1990.

[16] M. Sharir and P. K. Agarwal. *Davenport-Schinzel sequences and their geometric applications.* Cambridge University Press, New York, USA, 1995.

[17] C.K. Yap. Robust geometric computation. *Handbook of Discrete and Comp. Geometry.* CRC Press, 1997.

Partial Visibility Polygon with Semi-Transparent Objects

Mostafa Nouri Baygi* Mohammad Ghodsi†

Abstract

In this paper, we study partial visibility (or p-visibility) of a scene containing semi-transparent (convex) obstacles through which the light can pass partly. We define the p-visibility polygon and give algorithms to compute it for any query point. Then, we use the same technique to update the p-visibility polygon of a moving point, and to compute the maximum number of intersected objects by a ray emanating from a query point.

1 Introduction

We study the concept of partial visibility (or *p-visibility*) and present an algorithm to compute p-visibility polygon for a query point. Roughly speaking, the p-visibility polygon is the set of visible regions around a viewer when there are some semi-transparent objects in the scene.

P-Visibility can be applicable in many areas. For example, as Fulek *et al.* [2] mentioned, according to a model of wireless positioning service patented by Liu and Hung [3], the signal sent by a sensor can penetrate only at most a certain number, k, of obstacles and will not be received by the base station if there are more than k obstacles between the sensor and the base station.

Fulek *et al.* [2] studied a problem related to p-visibility. For a set S of n objects in the plane, and a point p, they defined $\tau(p, S)$ as the maximum number of objects that are intersected by all the rays emanated from p. Likewise, they defined $\tau(S)$ as the minimum value of $\tau(p, S)$ over all points p. Their problem is to provide an upper and lower bound for the value of $\tau(n)$, which is the maximum of $\tau(S)$ over all sets S of n objects. They also showed how $\tau(S)$ can be computed in $O(n^4 \log n)$.

In addition to algorithms for computing and updating the p-visibility polygon, we show how to compute $\tau(q)$ for a query point q. Briefly, we give the following results:

1. In the presence of some semi-transparent objects with total complexity of n, we compute any desired p-visibility polygon of a query point, in $O(n^2 \log(\sqrt{m}/n)/\sqrt{m} + |PVP(q)|)$ query time, using $O(m)$ space. Here $|PVP(q)|$ is the total size of the p-visibility polygons.
2. For a moving point, we maintain the p-visibility polygon of the point, in $O(n^2 \log(\sqrt{m}/n)/\sqrt{m})$ time for each change, and detect the first place that a change occurred in the same time.
3. For a query point q, we compute $\tau(q)$ in $O(n^2 \log(\sqrt{m}/n)/\sqrt{m})$.

In the above formulae, $n^2 \leq m \leq n^4$

2 Preliminaries

Assume that a set S includes l disjoint semi-transparent convex polygons, called objects, in the plane with total complexity of n. We need to compute the visible portion of the plane from an observer point. This problem is similar to computing the visibility polygon of a point.

Since the objects are semi-transparent, the light can partially pass through them, i.e., its intensity degrades when the light passes through an object. Assume that the light intensity only decreases when it enters an object. This way, we have different areas in the plane, each is visible with a different intensity. The problem is to compute these regions.

With the notion of p-visibility, for a point q and a parameter k, we define k-$PVP(q)$ as the set of points in the plane whose connecting line segments to q intersects at most k objects. Obviously, k-$PVP(q)$ contains j-$PVP(q)$ for $j < k$. For $k \geq \tau(S)$, k-$PVP(q)$ consists of all the plane and 0-$PVP(q)$ is the well-known visibility polygon of q. The main problem in this paper is to compute k-$PVP(q)$.

Let r_q be a ray emanating from q and let $\tau(r_q)$ be the set of objects intersected by r_q. For each q, we define $\tau(q) = \max_{r_q} \tau(r_q)$. This is the same as $\tau(q, S)$ proposed by Fulek *et al.* [2]. We can define $\tau(q)$ in another way: $\tau(q)$ is the smallest k, such that k-$PVP(q)$ is all the plane.

3 P-visibility polygon computation

In this section, we present an algorithm that computes k-$PVP(q)$. In this problem, we have a set S of convex polygons, with total complexity of $O(n)$, called objects, and a query point q. We can preprocess S so

*Department of Computer Engineering, Sharif University of Technology, nourybay@ce.sharif.edu

†Department of Computer Engineering, Sharif University of Technology, ghodsi@sharif.edu

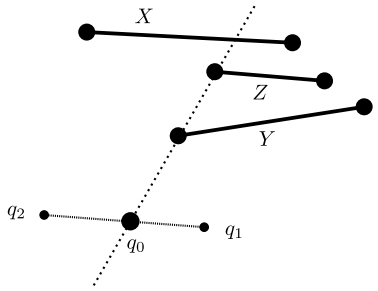


Figure 1: A tangent shows the place where the p-visibility changes occurs.

that for a given q k-$PVP(q)$ can be efficiently computed for any desired value of k.

We first describe the algorithm that uses $O(n^4)$ space, and then extend it to the general case with the space–query-time tradeoff.

3.1 Logarithmic query time

In order to compute k-$PVP(q)$ for a query point, we partition the plane into regions that all the points in each region have similar k-$PVP(q)$. The similarity is defined by the order of visibility of visible edges of the objects in S from observer. We denote this ordered list of visible edges by k-$\mathcal{PVP}(q)$. The following lemma helps identify the desired partition.

Lemma 1 *For a point q in the plane, k-PVP(q) for any k changes only if q moves across the tangent line of a pair of edges.*

Proof. Let q_1, q_2 be two points in the same cell in the arrangement, $\mathcal{A}$, of tangent lines of all pairs of the edges of the objects and k-$\mathcal{PVP}(q_1)$, k-$\mathcal{PVP}(q_2)$ differ, for some fixed value of k. Consider moving point q from q_1 towards q_2. When q moves, k-$\mathcal{PVP}(q)$ changes until it equals to k-$\mathcal{PVP}(q_2)$. Let q_0 be the first point at which k-$\mathcal{PVP}(q)$ changes. This change is in the form of removing a visible edge from or adding a new previously invisible edge to the list. Assume that it is in the latter form, so the object Z is added between X, Y. This means that before we reach q_0, the number of objects between q and any point on C was at least $k+1$, but in q_0, the number of objects between q to a point on C is at most k. This event happens only when a segment previously blocked C and then, in q_0, it becomes visible. As it can be seen in Figure 1, q_0 is on the tangent line of two segments (B, C in this example), which contradicts the assumption that q_1, q_2 are in the same cell. Similarly, the other case leads to a contradiction. □

As above lemma shows, we need to construct the arrangement $\mathcal{A}$ and compute k-$\mathcal{PVP}(p)$ for a point p in each cell. By this approach we can prove the following theorem.

Theorem 2 *Given a point q, we can compute k-PVP(q) for $0 \leq k \leq \tau(q,S)$, in $O(\log n + |k\text{-PVP}(q)|$ query time, while using $O(\tau(S)n^4)$ space and $O(\tau(S)n^4 \log n)$ preprocessing time.*

Proof. We construct the arrangement $\mathcal{A}$ and obtain a tour visiting all the cells of $\mathcal{A}$, such that each edge of $\mathcal{A}$ is visited at most twice, by a depth-first traversal of the cells of $\mathcal{A}$. Then, we select an arbitrary cell and compute k-$\mathcal{PVP}$ for all $0 \leq k \leq \tau(S)$ for an arbitrary point in that cell and store them, in a set of persistent red-black tree [5], each element of the set for one value of k. Since k-$\mathcal{PVP}$ is an ordered list of objects, it can be inserted in a binary search tree without any ambiguity. Afterwards, we move to the next cell in the tour and compute new lists of k-$\mathcal{PVP}$ in that cell. k-$\mathcal{PVP}$ in adjacent cells of $\mathcal{A}$ differ in 1 position, so we can store the new k-$\mathcal{PVP}$'s in the persistent data structures easily in $O(\tau(S)\log n)$.

In the query time, for a point q, we identify the cell in $\mathcal{A}$ that q lies in, and search in the persistent red-black tree for related k-$\mathcal{PVP}$. We can report this data structure as the ordered list of visible objects, or compute k-$PVP(q)$ precisely in the order of size of k-$PVP(q)$.

The construction of $\mathcal{A}$, which consists of $O(n^2)$ lines, takes $O(n^4)$ and in the same time we can create a tour. Computing k-$\mathcal{PVP}$ for $0 \leq k \leq \tau(S)$ in the first cell and storing them in the data structure takes $O(\tau(S)n\log n)$ time. Computing k-$\mathcal{PVP}$ for other cells in $\mathcal{A}$ each takes $O(\tau(S)\log n)$ and totally $O(\tau(S)n^4\log n)$. We should preprocess $\mathcal{A}$ for a point location data structure [1] which can be done in $O(n^4\log n)$. In the query time, a point location and a search in the persistent data structure is required to find the stored k-$\mathcal{PVP}(q)$, all of which takes $O(\log n)$. We can construct the actual k-$PVP(q)$ based on k-$\mathcal{PVP}(q)$. □

In above theorem, we assume that k is a parameter which is specified at query time, so we compute all the different p-visibility polygons and stored them in the persistent data structure. But if k is determined in the preprocessing step, we can reduce the memory space and preprocessing time considerably. In this case, we only compute p-visibility polygon for that specified k. This way the memory space (resp. preprocessing time) is reduced to $O(n^4)$ (resp. to $O(n^4\log n)$).

In the above result, we can compute k-$\mathcal{PVP}(q)$ for several values of k, without any change in the query time (except for reporting). This is because the search in the persistent data structure should be done only once and the associated lists can be returned easily.

Here, we provide two notes about how to optimize the arrangement $\mathcal{A}$. First, when we draw the line through a_1b_1 as a tangent for segments $A=(a_1,a_2)$ and $B=(b_1,b_2)$, the portion between a_1 and b_1 can be removed without any problem. This is because when

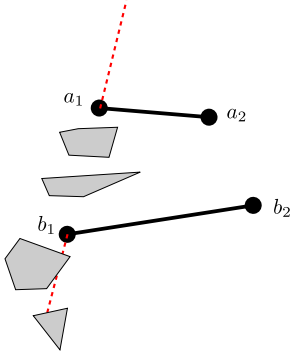


Figure 2: Optimization of the arrangement by reducing tangent lines. In this example, $k_{max} = 3$

q crosses from this part of the tangent, no changes occurs in the p-visibility polygons. Second, if the maximum value of k is determined in the preprocessing step, k_{max}, and we want to draw the tangent line through a_1b_1 as before, we can continue the tangent from b_1 as long as the points on it see at most k objects before a_1. The same is true for the other portion of the tangent (see Figure 2).

3.2 Space–query-time tradeoff

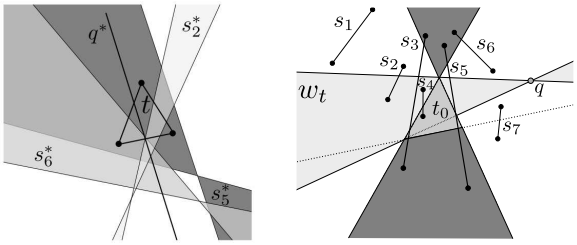
In this section, we show how to modify the previous method, and reduce the preprocessing space in the expense of an increase in the query time. The following lemma which is a modified version of the cutting theorem, in [4] is the main tool that helps achieve this.

Lemma 3 *Given a set of n lines in the plane, we can partition the plane into $O(r^2)$ triangles, for $1 \leq r \leq n$, such that each triangle is intersected by $O(n/r)$ lines and any arbitrary new line intersects at most $O(r)$ triangles.*

Theorem 4 *For any query point q, we can compute k-PVP(q) for all $0 \leq k \leq \tau(S)$, in $O(n^2 \log(\sqrt{m}/n)/\sqrt{m})$ query time, while using $O(\tau(S)m)$ space and $O(\tau(S)m \log(\sqrt{m}/n))$ preprocessing time, for an arbitrary $n^2 \leq m \leq n^4$.*

Proof. The set, P, of the vertices of the objects consists of $O(n)$ points and in the dual plane, this corresponds to a set of $O(n)$ lines, denoted by P^*. We start by constructing a cutting of size $O(r^2)$ for this set, such that each triangle of the cutting intersects $O(n/r)$ lines of P^*.

For a query point q in the primal plane, its dual line q^* intersects $O(r)$ triangles, and the intersection of q^* with these triangles, in the primal plane, partitions the plane into $O(r)$ co-centered disjoint double wedges, totally covering all the plane. Therefore, it is enough to compute the k-$PVP(q)$ in each double wedge, or equivalently, in each triangle intersected by q^*.



(a) Dual plane. (b) Primal plane.

Figure 3: (a) In the dual plane, the triangle t, dual of t_0, is intersected by q^*. The dual of some edges of objects are also shown. (b) The arrangement of t_0, q and edges of objects in the primal plane.

In Figure 3(a), the triangle t in the dual plane is shown which is intersected by q^*. The triangle t is also intersected by $O(n/r)$ lines which are dual to the end-points of $O(n/r)$ edges of objects in S. We call these segments the *closed segments* of t, denoted by the set CS_t. At least the dual of one end-point of each segment in CS_t intersects t. There are also some segments in O which are not in CS_t, but may appear in k-$\mathcal{PVP}(q)$, for example s_8 in Figure 3. We call these segments the *open segments* of t, denoted by the set OS_t. OS_t consists of all the segments whose dual double wedge contains t completely.

The triangle t_0, whose dual is t, as in Figure 3(b), partitions the plane into three regions. A region, which is brighter in the figure, is the region that q and at least one end-point of each segment of CS_t lie (for example segments s_6 and s_7). In contrast, we have two regions, which is shaded in the figure, and contains at most one end-point of each segment of CS_t and both end-points of the other segments of O. If an end-point of a segment o_i lies in a shaded region and the other end-point in the other shaded region, o_i belongs to OS_t (for example segments s_3 and s_5).

It is not hard to see that the segments in OS_t partition the bright region, the region which contains q, into $|OS_t|+1$ subregions. Let $R_t = \{r_t^1, \ldots, r_t^{|OS_t|+1}\}$ denote the set of subregions. In each subregion r_t^k, a set of segments $CS_t^k \subset CS_t$ is contained.

Since the view around q is bounded to w_t, we can imagine each segment in OS_t as an infinite line. Therefore, to identify the cells with uniform k-$\mathcal{PVP}(q)$, we only construct the arrangement of tangents for each pair of segments in CS_t and not OS_t. We only consider the objects of segments in OS_t as obstacles.

With this approach, we can use the previous method with logarithmic time in each double wedge. For each triangle t, we spend $O(\tau(S)(n/r)^4)$ space and $O(\tau(S)(n/r)^4 \log(n/r))$ preprocessing time.

To complete the argument, we should say how to compute CS_t and OS_t. CS_t of complexity $O(n/r)$, which is already computed during the construction of the cutting $\mathcal{R}$, is the set of segments with an end-point whose dual intersects t.

It is also notable that, we cannot compute OS_t for each triangle independently, since the size of OS_t may be large, compared to $O((n/r)^4)$, which we can spend for each triangle. The elements of OS_t for each t, is very similar to $OS_{t'}$ of an adjacent triangle t'. The differences are in the elements of $CS_{t'}$ and CS_t, that is some segments in CS_t may be removed from $OS_{t'}$ and some segments in $CS_{t'}$ may be added to $OS_{t'}$ to produce OS_t.

At query time, we find the cell in t^* that contains q by a point location and return the associated k-$\mathcal{PVP}$ in $O(\log(n/r))$ time for each triangle t that is intersected by q^*. We can easily compute k-$\mathcal{PVP}(q)$ bounded by w_t in the same time. Summing up these amounts for $O(r)$ triangles, we can compute k-$\mathcal{PVP}(q)$ in $O(r\log(n/r))$ time.

In summary, the total preprocessing time and space are $O(\tau(S)n^4\log(n/r)/r^2)$ and $O(\tau(S)n^4/r^2)$ respectively and the query time is $O(r\log(n/r))$. If the used space is denoted by m we can prove the claim.

□

4 Applications

In this section, we apply the techniques used in the previous section to two related problems and give solutions for them.

4.1 Maximum intersecting objects

Theorem 5 *For any query point q we can compute $\tau(q)$ in $O(n^2\log(\sqrt{m}/n)/\sqrt{m})$, using $O(m)$ space and $O(O(m\log(\sqrt{m}/n)))$ preprocessing time, for $n^2 \leq m \leq n^4$*

Proof. Consider the cutting we used before in the dual plane for the vertices of the objects. For each point q, the intersection of q' and triangle t, in the primal plane corresponds to the double wedge w_t. Here we should change the data structures, so that instead of storing the actual k-$\mathcal{PVP}(q)$ in each cell of the arrangements, we only store the maximum value of objects, that are intersected by any ray emanated from any point inside w_t, denoted by $\tau_t(q)$. To compute $\tau(q)$, we should compute $\tau_t(q)$, for all triangles t that are intersected by q', and choose the maximum value among them, which can be done in the same query time as before. □

4.2 P-visibility polygon of a moving point

Theorem 6 *For a moving point q in the plane, which moves on a straight line, we can detect the first place where k-$\mathcal{PVP}(q)$ changes for some k and update k-$\mathcal{PVP}(q)$ in $O(\frac{n^2}{\sqrt{m}}(\log\frac{\sqrt{m}}{n}))$. The preprocessing time and space are respectively $O(\tau(S)m\log(\sqrt{m}/n))$ and $O(\tau(S)m)$.*

Proof. Assume we can use $O(\tau(S)n^4)$ space for computing k-$PVP(q)$. In this case, q is a point which lies in a cell c in the arrangement of tangent lines. c is a convex polygon, therefore for a straight line, we can identify in $O(\log n)$, from which edge of c, q leaves it. Once q leaves c, k-$PVP(q)$ may change for some value of k. We can easily detect this event and update k-$\mathcal{PVP}(q)$ based on the edge q crosses.

For the case that we use tradeoff, consider line q^* in the dual plane. When q moves on a straight line, q^* rotates around a fixed center. In each triangle, we can easily detect the first change in the visibility similar to the previous case that was described. Here we need to choose the first place from these $O(r)$ places that changes occurred. All of these, can be accomplished in $O(r\log(n/r))$. Substituting r with $n^2/\sqrt{m}$ proves the theorem. □

5 Conclusion

In this paper, we introduced the p-visibility concept and presented an algorithm that computes p-visibility polygon of a query point in logarithmic time. We then extended the algorithm to reduce the space usage, but in the expense of an increase in the query time.

Finally, we used the method to solve two related problems: updating p-visibility polygon of a moving point and computing the maximum number of objects that are intersected by a ray emanated from a query point. For the future works, we intend to solve this problem: for a set of objects S, find a point p such that $\tau(S) = \tau(p)$ in optimal time.

References

[1] B. Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete Comput. Geometry*, 9:145–158, 1993.

[2] R. Fulek, A. F. Holmsen, and J. Pach. Intersecting convex sets by rays. *Discrete Comput. Geometry*, 42(3):343–358, 2009.

[3] C.-T. Liu and T.-Y. Hung. Method of building a locating service for a wireless network environment. patent no. 7203504, April 2007. www.freepatentsonline.com/7203504.html.

[4] M. Nouri and M. Ghodsi. Space–query-time tradeoff for computing the visibility polygon. In *FAW '09*, pages 120–131. Springer-Verlag, 2009.

[5] N. Sarnak and R. E. Tarjan. Planar point location using persistent search trees. *Commun. ACM*, 29(7):669–679, 1986.

Computing the visibility area between two simple polygons in linear time

Elmar Langetepe* Rainer Penninger* Jan Tulke†

Abstract

We consider a visibility problem for two non-intersecting open or closed simple polygonal chains P and Q in the plane. The visibility area between P and Q is the union of all line segments pq where p lies on the boundary of P and q on the boundary of Q and pq does neither intersect with P nor with Q. We present an optimal linear time algorithm for computing this area. The given work generalizes known visibility results and has an application in computer aided construction management.

Keywords: Visibility in the plane, polygonal chains, optimal algorithm

1 Introduction

Computing the visibility among geometric objects is one of the most natural subjects in the field of computational geometry. Furthermore, efficient solutions of visibility problems have application in many fields of computer science such as robotics [8], computer graphics [3] and computer vision [4]. For an overview of efficient solutions for visibility problems in the plane one can consider the survey of Asano et al.[1] or the textbook of Gosh[5].

The given result has two main benefits. On one hand, we generalize a known visibility result inside simple polygons. It was already shown how to compute the inner visibility region of an edge e of a simple polygon P efficiently. Instead of e and P we allow more general polygonal objects P and Q while maintaining optimal linear time. The visibility area between P and Q is the union of all line segments pq where p lies on the boundary of P and q on the boundary of Q and pq does neither intersect with P nor with Q, see Figure 1 for an example. Furthermore, with the same technique we can compute the visibility region of a polygon P inside a polygon Q in time proportional to $|P| + |Q|$ which is a natural extension of visibility of a single point or egde. On the other hand, the given problem arises in the context of a digital three dimensional building construction model. An efficient computation of the visibility area between two wall axes A and B in the digital model of a building is an important question for computer aided construction management.

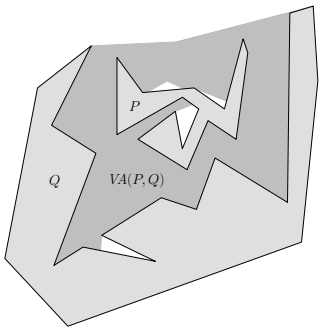


Figure 1: The visibility area between two non-intersecting polygonal chains P and Q.

In Section 2 we extend the algorithm of Guibas et al. [6] and compute the visibility region of a polygonal subchain C_P along the boundary of a polygon P in linear time. Afterwards, we make use of this result and compute the visibility area between two non-intersecting polygonal objects P and Q in time $O(|P| + |Q|)$. Note that due to lack of space some proofs are omitted.

2 Visibility of a boundary subchain

We would like to compute the inner visibility region of a connected subchain $C_P = (p_1, \ldots, p_k)$ along the boundary ∂P of a simple polygon $P = (p_1, \ldots, p_n, p_1)$, see Figure 2. A point $q \in P$ is *visible* from $p \in C_P$ if $pq \in P$ holds.

Definition 1 (Visibility polygon) *Given a polyon P, and a connected point set $X \in P$, the visibility polygon $Vis_P(X)$ is the set of points inside P that are visible from any point $x \in X$.*

For computing $Vis_P(C_P)$ we first compute the shortest path $\pi_{p_1,p_k} = (r_1, \ldots, r_l)$ in P from p_1 to p_k which can be done in linear time ([6], [2]). Replacing $(p_1, \ldots, p_k)$ by π_{p_1,p_k}, we obtain a new polygon $R = (r_1, \ldots, r_l, p_{k+1}, \ldots p_n, r_1)$. Since the number of vertices of π_{p_1,p_k} is at most n, the number of vertices of R is at most $2n \in O(n)$. For short, we define $C_r = (r_1, \ldots, r_l)$ and $C_l = (r_l, p_{k+1}, \ldots, p_n, r_1)$.

It can be shown that for computing $Vis_R(C_r)$ it suffices to compute $Vis_R(C_r)$.

*Department of Computer Science I, RFW University Bonn, Institut für Informatik, Abt. I, elmar.langetepe@cs.uni-bonn.de

†HOCHTIEF ViCon GmbH, Alfredstrasse 236, D-45133 Essen

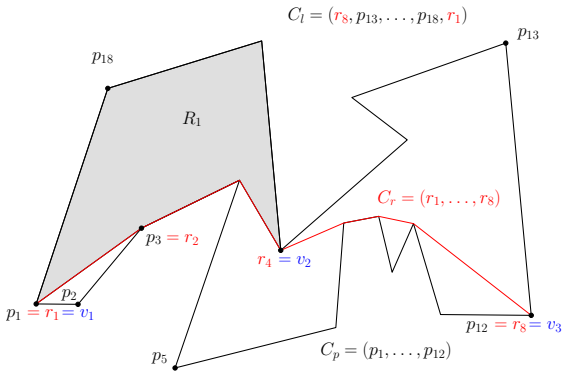


Figure 2: For computing the visibility polygon of a subchain $C_P = (p_1, \ldots, p_k)$ of P it suffices to compute visibility polygons along the shortest path π_{p_1,p_k}.

Lemma 1 *With the notation from above we have:*

$$Vis_P(C_p) = Vis_R(C_r) \cup (P \setminus R).$$

The remaining task is how to compute $Vis_R(C_r)$ of the shorthest path C_r. Note that R need not be a simple polygon because the shortest path C_r may touch C_l, compare r_4 in Figure 2. We will make use of such *tangent points* and subdivide the polygon R into several simple polygons R_i. The aim is to show that the union of the visibility polygons of the corresponding parts of C_r is equal to $Vis_R(C_r)$.

Formally, the division into subpolygons is defined as follows: Because, originally, C_r is a shortest path between two vertices of P, its set of vertices $V[C_r]$ also consists of the vertices of P. Let $(v_1, \ldots, v_t) = (V[C_l] \cap C_r)$ denote the vertices of C_l that lie on C_r. The vertices v_i lie consecutively on C_r. We have, by definition, $v_1 = r_1$ and $v_t = r_l$. We denote with $C_r(i) := (v_i, \ldots, v_{i+1})$ the polygonal chain on C_r between v_i and v_{i+1}. Since C_r is a shortest path we conclude that $C_r(i)$ has to be an *outward bent chain*, i.e., while walking from v_i to v_{i+1} on $C_r(i)$ we always turn to the right at the vertices in between. Now let $C_l(i)$ denote the polygonal chain between v_i and v_{i+1} on C_l. We conclude that $R_i := C_r(i) \cup C_l(i)$ is a simple polygon (unless $V[R_i]$ consists of only the two vertices v_i, v_{i+1}), see Figure 2. The following Lemma states that we can obtain $Vis_R(C_r)$ by computing $Vis_{R_i}(C_r(i))$ for all R_i.

Lemma 2 *With the notation from above we have:*

$$Vis_R(C_r) = \bigcup_{i=1}^{t-1} Vis_{R_i}(C_r(i)).$$

It remains to show how to compute the visibility polygon of an outward bent chain $C_r = (r_1, \ldots, r_l)$ of ∂R_i of a simple polygon R_i. Let $C_l = (r_l, p_{k+1}, \ldots, p_n, r_1)$ denote the remaining boundary of R_i, see Figure 3. The key idea of the algorithm of Guibas et al. [6] is the following: Roughly speaking, a point $p \in R_i$ is visible from an edge $e = (a, b)$ iff the two shortest paths $\pi_{a,p}$ and $\pi_{b,p}$ are outward convex[1]. The algorithm traverses two shortest path trees rooted at a and b, and cuts off those points p of which the two paths $\pi_{b,p}$ and $\pi_{a,p}$ are not outwards convex. In our case the situation is slightly different. The visibility condition is no longer correct if we replace e by a polygonal chain C_r that is bent outwards. It is possible that the shortest paths begin with a part of C_r, and that the resulting paths are not outward convex, although p is visible from C_r. This is fixed by ignoring the first vertices on the shortest paths that belong to C_r, thus adapting the test for outward convexity to the new situation. So the new algorithm does the same as before but ignores the first edges between vertices of C_r on the shortest path trees when testing for outward convexity of the shortest paths. Figure 3 illustrates these observations. Point z is not visible from C_r because the paths $\pi_{r_2,z}$ and $\pi_{r_4,z}$ are not outwards convex. Point x is visible from C_r.

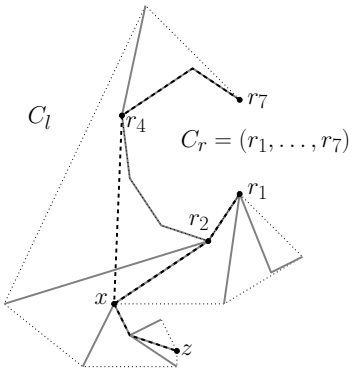


Figure 3: The shortest path tree rooted at r_1 decomposes into subtrees rooted at the vertices r_i.

Altogether we have:

Lemma 3 *The visibility polygon of an outward bent subchain C_r of the boundary of a polygon P can be computed in linear time.*

Finally, using the previous three lemmata we are able to compute the visibility polygon of a boundary subchain C_P of a simple polygon P in linear time.

Theorem 4 *Given a polygon P and a subchain $C_P = (c_1, \ldots, c_k)$ of ∂P, we can compute $Vis_P(C_P)$ in time $O(|P|)$.*

3 The visibility area of two boundary chains

The main tool for computing the visibility area between two polygons will be the computation of the

[1] A pair of paths $\pi_{a,p}$ and $\pi_{b,p}$ is outwards convex iff the convex hull of each path lies outside the open region bounded by $\pi_{b,p} \cup \pi_{a,p} \cup |ab|$, where $|ab|$ denotes the line segment connecting a and b.

visibility area between two chains of the boundary of a common polygon. Before we describe how to compute the visibility area of two chains we give a formal definition:

Definition 2 (Segment type) *A segment pq between two mutually visible points p and q is (of type) PQ, iff $p \in P$ and $q \in Q$, where P and Q are sets of points.*

Definition 3 (Visibility area) *Given two polygonal chains P and Q, then the visibility area $VA(P,Q)$ consists of all points that lie on a segment of type PQ.*

If P and Q are subchains of the same closed polygonal chain C, then we restrict $VA(P,Q)$ to points in the bounded region defined by C.

In a first step we would like to compute the visibility area between two subchains A and C on the boundary of a common polygon P. Here, the computation of the visibility area between A and C can be reduced in a natural way to the computation of constrained visibility chains:

Definition 4 (Constrained visibility chain) *Given two subchains X, C of a closed polygonal chain P, and the visibility polygon $Vis_P(X)$ of X, we define a polygonal chain $Vis_P(X)_{|C}$ as the union of all points $c \in C$ visible from X, and all visibility cuts connecting points of C visible from X.*

First, we compute the visibility polygon $P_A = Vis_P(A)$. The boundary of P_A contains the constrained visibility chain $C' = Vis_P(A)_{|C}$ of C w.r.t. visibility from A. Then computing the visibility polygon P_{AC} of C' in P_A we obtain all points of P that are visible from both A and C. Note that P_{AC} contains $A' = Vis_P(C)_{|A}$ on its boundary, as well as C'. There are two chains B, D connecting A', C' on the boundary of P_{AC}. By replacing B and D with the shortest paths between their endpoints, we subtract all points from P_{AC} that do not lie on a segment of type AC. A formal proof of this observation is omitted due to lack of space. Then we have finally obtained the visibility area between A and C.

Lemma 5 *The visibility area between two subchains A, C of a polygon P can be computed in time $O(|P|)$.*

Proof. The two visibility polygons can be computed in linear time by Theorem 4. After triangulating the resulting polygon P_{AC}, the two shortest paths can also be computed in linear time. □

4 Two arbitrary closed polygonal chains P and Q.

We now describe the most complex case where P lies outside of Q, but lies completely inside the convex hull $ch(Q)$ of Q. Otherwise if $P \subset Q$ we can imagine that instead Q totally surrounds P, but P lies outside of Q; if $P \not\subseteq ch(Q)$, the visibility area can be computed by a comparably simple algorithm, after the two outer tangents have been computed [10]. Note that we can decide which case applies in linear time.

Since P is contained in $ch(Q)$, but is not contained in Q, P is contained in a pocket of polygon Q. We have to compute the visibility area between P and this pocket. The pocket can be identified in linear time by counting the intersections with a ray starting from a vertex of P [7]. Let $pocket(Q)$ denote the part of Q's boundary contributing to the pocket, and $e = (a, b)$ be the edge of $ch(Q)$ defining the pocket. From now on we denote with $\overline{Q}$ the closed polygonal chain $pocket(Q) \cup e$, so P lies inside the bounded region defined by $\overline{Q}$. In order to compute $VA(P,Q)$ we do the following:

1. Compute $VA(P, \overline{Q})$.
2. Remove those points from $VA(P, \overline{Q})$ that only lie on segments of type Pe (but not on a segment of type PQ).

4.1 Compute $VA(P, \overline{Q})$

We consider the polygon $R = \overline{Q} \setminus Int(P)$. If R were a simple polygon we knew, by Lemma 5, how to compute $VA(P, \overline{Q})$. But P defines a hole in R, so R is no simple polygon. But there exist two edges e_l, e_r of type PQ on the convex hull of Q relative[2] to P, which can be identified in linear time and have a nice property: No segment of type $P\overline{Q}$ intersects *both* e_l and e_r.

Inserting two copies of e_x into R, where $x = l$ or $x = r$, we obtain a simple polygon R_x, of which P and Q are boundary chains, connected by the two copies of e_x. We compute $VA_{e_x} = VA(P, \overline{Q})$ in R_x. Now by the property of the edges e_l and e_r we have $VA(P, \overline{Q}) = VA_{e_l} \cup VA_{e_r}$. We unite VA_{e_l} and VA_{e_r} as follows: The boundary of VA_{e_x} consists of parts of the boundary of P, of $\overline{Q}$, and of visibility cuts of type PP or $\overline{Q}\overline{Q}$. We compute the set of cuts for both VA_{e_x} and cut off parts of the boundary of P or $\overline{Q}$ only if the part is cut off by a visibilty cut from VA_{e_l} and from VA_{e_r}. This is a matter of traversing the boundaries of P and of $\overline{Q}$ once, which can be done in linear time.

[2] See Toussaint [9] for details.

4.2 Remove points from $VA(P, \overline{Q})$

Our goal is, after computing $VA(P, \overline{Q})$, to remove the points of $VA(P, \overline{Q})$ that do not belong to $VA(P, Q)$. Only some special points have to be removed: The points p that see a point of $r \in Int(e)$ are candidates for subtraction. Suppose we look from p to r. Then p has to be subtracted iff turning clockwise until seeing r again

1. we have only seen points of P and of e, or
2. we have at some point seen a point of Q, then a point of P, before then again seeing a point of Q. Also, there is no segment of type PQ containing p.

The points of type 1 are not seen from Q – they lie in some sort of pocket of P. Those of type 2 are the points which lie close to edge e, and thus do not lie on a segment of type PQ, but on segments of type Pe.

Lemma 6 *The points of type 1 can be subtracted from $VA(P, \overline{Q})$ in linear time.*

Proof. Two subchains of the boundary of the visibility region of e in $VA(P, \overline{Q})$ stem from the polygon Q. From those we can compute the cuts defining the pockets containing the points of type 1 in linear time. As above, we only cut off points that lie behind cuts stemming from both subchains. □

Due to lack of space we state the following Lemma without proof.

Lemma 7 *The points of type 2 can be subtracted from $VA(P, \overline{Q})$ in linear time.*

The above observations prove our main theomem:

Theorem 8 *The visibility area $VA(P, Q)$ between two polygons P and Q can be computed in time $O(|P| + |Q|)$.*

5 Concluding remarks

An open polygonal chain $C = (c_1, \ldots, c_n)$ can be treated as a closed polygonal chain $C' = (c_1, \ldots, c_n, \ldots, c_1)$. The visibility of C and C' is identical and we can apply our algorithms on C'.

Given a polygon P and an obstacle polygon Q we can compute $Vis(P)$ w.r.t. Q – see Figure 4 – analogously to $VA(P, Q)$, with the following modifications: First, points not visible from Q need not be subtracted from the polygon R between P and Q. Second, if $Vis(P)$ is unbounded, we need to replace parts of the boundary of $VA(P, Q)$ with adequate visibility cuts of infinite length.

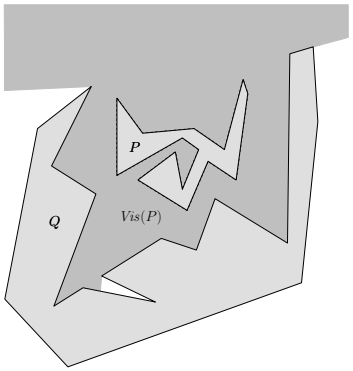


Figure 4: The visibility region of polygon P with obstacle polygon Q.

References

[1] T. Asano, S. K. Ghosh, and T. C. Shermer. Visibility in the plane. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 829–876. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000.

[2] B. Chazelle. Triangulating a simple polygon in linear time. *Discrete Comput. Geom.*, 6(5):485–524, 1991.

[3] D. P. Dobkin and S. Teller. Computer graphics. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 42, pages 779–796. CRC Press LLC, Boca Raton, FL, 1997.

[4] O. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, Cambridge, MA, 1993.

[5] S. K. Gosh. *Visisbility Algorithms in the plane*. Cambridge University Press, 2006.

[6] L. J. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. E. Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2:209–233, 1987.

[7] R. Klein. *Algorithmische Geometrie*. Addison-Wesley, Bonn, 1997.

[8] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, 1991.

[9] G. Toussaint. On separating two simple polygons by a single translation. *Discrete Comput. Geom.*, 4:265–278, 1989.

[10] G. T. Toussaint. Solving geometric problems with the rotating calipers. In *Proc. IEEE MELECON '83*, pages A10.02/1–4, 1983.

Triangulating a System of Disks

Daniel Peterseim**

Abstract

We introduce a generalization of Delaunay triangulations applicable to a system of disks. In this subdivision of the convex hull of the system of disks, the classical role of vertices is assumed by the disks while neighboring disks are connected by channel-like objects which therefore assume the classical role of edges. The generalized Delaunay triangulation is derived by performing a limiting process of classical Delaunay triangulations with respect to a convergent sequence of polygonal approximations of the system of disks. We comment on duality with respect to certain Voronoi diagrams.

1 Introduction

Let $\mathcal{B}$ be a system (finite set) of closed disks in the plane, i.e. for every $B \in \mathcal{B}$ there is a center $\mathbf{c}_B \in \mathbb{R}^2$ and radius $r_B > 0$ so that

$$B = \{\mathbf{x} \in \mathbb{R}^2 \mid dist(x, \mathbf{c}_B) \leq r_B\};$$

$dist(\cdot,\cdot)$ being the Euclidean distance in $\mathbb{R}^2$. We assume that the disks are pairwise disjoint, i.e. $B_1 \cap B_2 = \emptyset$ for all $B_1, B_2 \in \mathcal{B}$.

The restriction to the two dimensional setting conduces to keep the presentation of the underlying basic idea, which is simple, as clearly as possible. The simple setting yet has an interesting practical application. The disks in $\mathcal{B}$ can be considered as cross sections of fibers in fiber-reinforced composite materials, e.g. fiber glass. It is an important task in computational mechanics to derive insight about effective material properties, e.g. transport, mechanical, and electromagnetic properties, as well as properties associated with coupled phenomena, such as piezoelectric and thermoelectric coefficients. If the considered fiber composite is unidirectional, then effective material properties can often be modeled by partial differential equation on some cross section justifying the usefulness of our geometric model.

The solution of such partial differential equations is challenging due to the highly complicated geometry represented by the system of disks which either forms a part of the domain boundary or a region with significantly different coefficient in the corresponding differential operator. Using a standard finite element method, the geometry, i.e. the system of disks, has to be resolved by the underlying computational mesh.

The aim of this paper is to describe an efficient and problem adapted subdivision of the convex hull of the system of disks $\mathrm{conv}(\cup\mathcal{B})$ to be used within special finite element methods. The desired subdivision will turn out to be a generalization of the classical Delaunay triangulation [3].

Given a set of points $S \subset \mathbb{R}^2$, the classical Delaunay triangulation is a set of (closed) triangles $\mathcal{D}(S)$ determined by the classical Delaunay criterion saying that the open circumdisk of any triangle must not contain any elements of S. The Delaunay triangulation $\mathcal{D}(S)$ is not unique if S contains 4 points that are cocircular. We cure this issue, known as geometric degeneracy, by considering $\mathcal{D}(S)$ as a subdivision into (closed) cyclic polygons with 3 or more vertices such that a strict Delaunay criterion is fulfilled: The (unique) closed circumdisk of each cyclic polygon does not contain any vertices of S excepts its own ones.

In this paper we describe a generalized concept of a triangulation, in which the point set S is replaced by the system of disks. More precisely, we introduce generalized Delaunay triangulations as the limit of certain classical Delaunay triangulations approximating the system of disks (see Section 2). Finite element spaces based on these new geometry subdivisions can be derived similarly as indicated in Section 3. By duality, generalized Delaunay triangulations are closely related to certain (additively weighted) Voronoi diagrams (see Section 4).

Notation. We use capital letters $A, B, C, \ldots$ to indicate sets, bold letters $\mathbf{x}, \mathbf{y}, \mathbf{z}, \ldots$ indicate points in $\mathbb{R}^2$. Systems of sets are denoted by calligraphic capital letters $\mathcal{B}, \mathcal{D}, \ldots$ with the only exception of the power set of a set denoted by $\mathbb{P}$. For systems of sets $\mathcal{B}$ we denote the union of its elements by $\cup\mathcal{B} := \bigcup_{B\in\mathcal{B}} B$. We further make use of basic topological notations: For any $X \subset \mathbb{R}^2$ we denote its closure by $cl(X)$, its relative interior by $relint(X)$, and its boundary by $bnd(X)$.

2 Derivation of the Generalized Delaunay Triangulation

Consider polygonal approximations $\mathcal{B}^n$, $n \in \mathbb{N}$, of $\mathcal{B}$ in which the disks are approximated by certain regular

*Institut für Mathematik, Humboldt-Universität zu Berlin, peterseim@math.hu-berlin.de

*The author is supported by the DFG Research Center MATHEON, Berlin.

polygons. More precisely, $\mathcal{B}^n := \{B^n \mid B \in \mathcal{B}\}$, where B^n is a regular, i.e. equiangular and equilateral, polygon with 2^n vertices $V(B^n)$ located on $bnd(B)$ (the circumcircle of B^n). By

$$V(\mathcal{B}^n) := \bigcup_{B^n \in \mathcal{B}^n} V(B^n)$$

we denote the set of vertices of $\mathcal{B}^n$. We assume the polygonal approximations $\mathcal{B}^n$ to be nested, i.e. $V(\mathcal{B}^n) \subset V(\mathcal{B}^{n+1})$ for all $n \in \mathbb{N}$. The classical Delaunay triangulation with respect to the point set $V(\mathcal{B}^n)$ is denoted by $\mathcal{D}^n := \mathcal{D}(V(\mathcal{B}^n))$. Figure 1(a,b,c) shows some set of disks, the corresponding polygonal approximations, and triangulations for n being 3 and 5. We observe that for n tending to infinity, new structures are evolving, namely channel-like connections between neighboring disks.

In order to investigate this process let us fix some vertex $\mathbf{x} \in V(B^n)$, where $B^n \in \mathcal{B}^n$ for some $n \in \mathbb{N}$. By construction, $\mathbf{x} \in V(B^m)$ for all $m > n$. Consider the set

$$\mathcal{T}_{\mathbf{x}}^m := \{T \in \mathcal{D}^m \mid \mathbf{x} \in V(T)\}$$

containing cyclic polygons that have $\mathbf{x}$ as a vertex. It is obvious that B^m itself is an element of $\mathcal{T}_{\mathbf{x}}^m$. The circumdisks of the remaining elements of $\mathcal{T}_{\mathbf{x}}^m$ form the set

$$C_{\mathbf{x}}^m := \bigcup\{\mathrm{conv}(C_T) \mid T \in \mathcal{T}_{\mathbf{x}}^m \setminus \{B^m\}\},$$

where C_T denotes the circumcircle of the cyclic polygon T. Due to the Delaunay property, $C_{\mathbf{x}}^m$ converges to the maximal (w.r.t. to the diameter) disk $C_{\mathbf{x}} \subset \mathbb{R}^2 \setminus (\cup\mathcal{B})$ that is tangential to B in $\mathbf{x}$. Note that $C_{\mathbf{x}}$ is infinite (a shifted halfspace) if and only if $\mathbf{x} \in bnd(\mathrm{conv}(\cup\mathcal{B}))$. The disk $C_{\mathbf{x}}$ tangentially intersects other disks of $\mathcal{B}$ besides B. These (finitely many) intersection points collected in

$$C_{\mathbf{x}} \cap (\cup\mathcal{B}) = \{\mathbf{x}\} \cup \lim_{m\to\infty} \mathcal{T}_{\mathbf{x}}^m \setminus \{B\} \qquad (1)$$

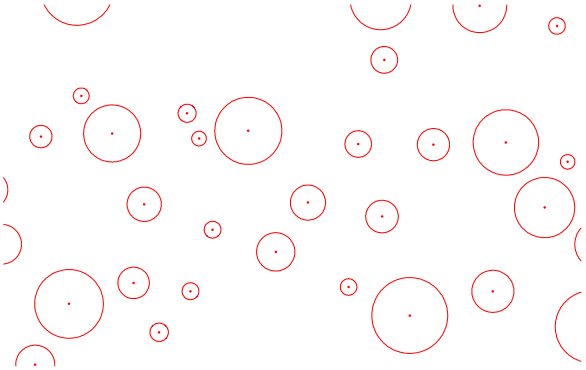
span a unique cyclic polygon $T_{\mathbf{x}} := \mathrm{conv}(C_{\mathbf{x}} \cap (\cup\mathcal{B}))$. Considering disks as cyclic polygons with infinitely many vertices, a subdivision of $\mathrm{conv}(\cup\mathcal{B})$ into cyclic polygons is given by

$$\mathcal{D}^\infty := \mathcal{B} \cup \{T_{\mathbf{x}} \mid \mathbf{x} \in bnd(\cup\mathcal{B})\}. \qquad (2)$$

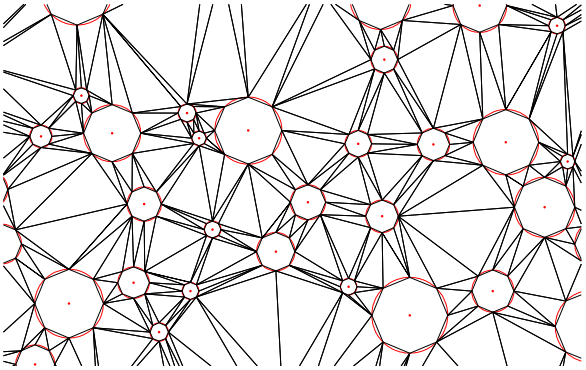
The above subdivision fulfills the (strict) Delaunay criterion. However, $\mathcal{D}^\infty$ has infinitely many elements which is not desired from a practical point of view.

As it can be seen in Figure 1(d) and as indicated earlier the elements of $\mathcal{D}^\infty$ form new structures. The desired (finite) subdivision is derived by grouping the elements of $\mathcal{D}^\infty$ according to the disks that contain their vertices. For $T \in \mathcal{D}^\infty$ we define

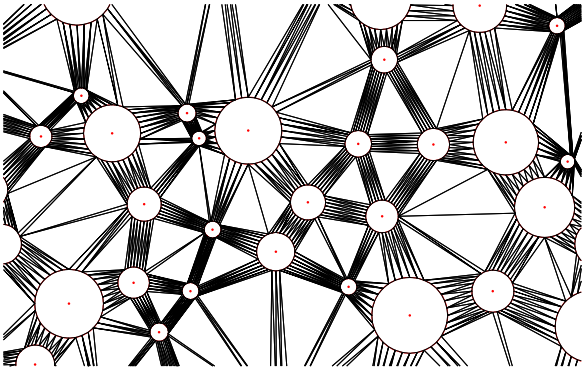
$$\mathcal{B}(T) := \{B \in \mathcal{B} \mid V(T) \cap B \neq \emptyset\};$$



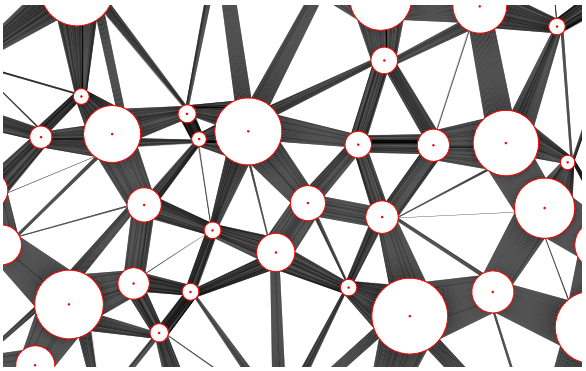
(a) Some system of disks $\mathcal{B}$ (detail).



(b) Approximation of the disks by octagons and the corresponding Delaunay triangulation $\mathcal{D}^3$.



(c) Approximation of the disks by 32-gons and the corresponding Delaunay triangulation $\mathcal{D}^5$.



(d) The infinite Delaunay triangulation $\mathcal{D}^\infty$.

Fig. 1: Construction of the generalized Delaunay triangulation.

we say that $T \in \mathcal{D}^\infty$ connects a subset of disks $\mathcal{A} \subset \mathcal{B}$ if $\mathcal{B}(T) = \mathcal{A}$. $\mathcal{B}(\cdot)$ maps the cyclic polygons of $\mathcal{D}^\infty$ into $\mathbb{P}(\mathcal{B})$, which is a finite set. The generalized Delaunay triangulation is closely related to the preimages of ensembles of disks under $\mathcal{B}(\cdot)$. In this context we distinguish 3 characteristic classes of cyclic polygons:

1. Cyclic polygons that connect only a single disk, i.e. either the disks itself or single points in $relint(bnd(\cup\mathcal{B}) \cap \mathrm{conv}(\cup\mathcal{B}))$, which can be neglected. We refer to the disks as generalized vertices.
2. Cyclic polygons that connect exactly two disks. The union of polygons that connect a certain pair of disks is denoted as a generalized edge. More precisely, a generalized edge connecting $B_1, B_2 \in \mathcal{B}$ is given by
$$E_{B_1,B_2} := \bigcup \{T \in \mathcal{D}^\infty \mid \mathcal{B}(T) = \{B_1, B_2\}\}.$$
3. Cyclic polygons that connect 3 or more disks.

The generalized Delaunay triangulation is now given as the set of generalized vertices, generalized edges and triangles connecting 3 or more disks:

$$\mathcal{G} := \mathcal{B} \cup \mathcal{E} \cup \mathcal{T},$$

where $\mathcal{B}$ is the system of disks under consideration,

$$\mathcal{E} := \{E_{B_1,B_2} \neq \emptyset \mid B_1 \neq B_2 \in \mathcal{B}\}$$

and

$$\mathcal{T} := \left\{ T \in \mathcal{D}^\infty \;\middle|\; \begin{array}{l} \mathcal{B}(T) = \{B_1, B_2, \ldots, B_k\}, \\ B_1 \neq \ldots \neq B_k \in \mathcal{B},\ k \geq 3 \end{array} \right\}.$$

In contrast to classical triangulations, the consideration of vertices and edges as genuine elements of the subdivision is essential to ensure that the union of all elements indeed covers $\mathrm{conv}(\cup\mathcal{B})$. As in the classical setting intersections of distinct elements are of lower (at most 1) dimension.

3 Finite Element Methods

As currently worked out in [9] and already indicated in [8] generalized Delaunay triangulations, as introduced here, are suitable for the use within finite element simulations of effective properties of composite materials. The according finite element spaces can be derived in a similar manner as the triangulation itself, i.e. by first taking the limit of classical spaces with respect to the approximative triangulations $\mathcal{D}^n$ and then choosing a finite subspace appropriate for the problem to be solved. Considering, e.g., heat conductivity in a fiber composite with perfectly conducting fibers, a suitable discrete space is derived by taking the limit of classical continuous first order elements and then requiring the shape functions to be constant with respect to every disk (due to perfect conductivity). The latter space is spanned by the generalized nodal basis; an element of this basis is depicted in Figure 2.

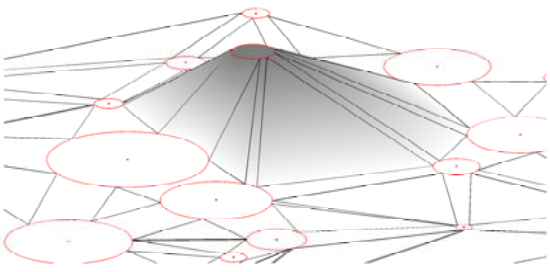


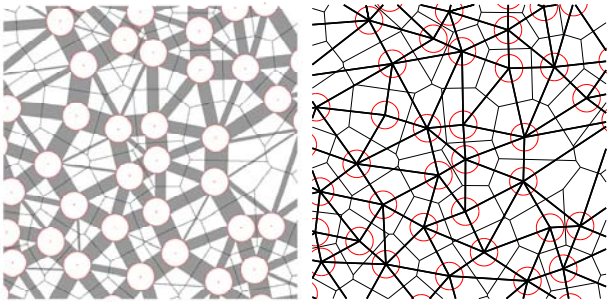
Fig. 2: A generalized nodal basis function taking value 1 in a certain disk and 0 in all others.

4 Generalized Voronoi-Delaunay Duality

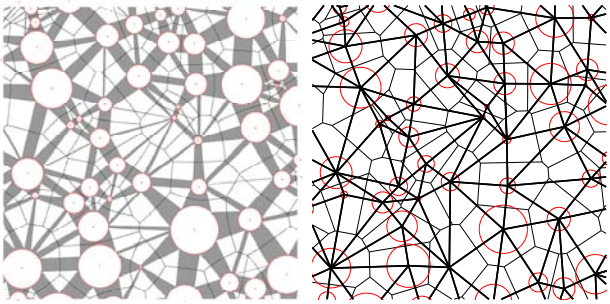
Consider again the circumcircles $C_{\mathbf{x}}$, $\mathbf{x} \in bnd(\cup\mathcal{B})$, of the elements of the (infinite) Delaunay triangulation $\mathcal{D}^\infty$ (see (1) and (2)) and note that $C_{\mathbf{x}} = C_{\mathbf{y}}$ for all $\mathbf{y} \in C_{\mathbf{x}} \cap (\cup\mathcal{B})$. Moreover, the center of $C_{\mathbf{x}}$ is equidistant to all $\mathbf{y} \in C_{\mathbf{x}} \cap (\cup\mathcal{B})$. Thus, the union of all circumcircle centers $\mathbf{c}_{C_{\mathbf{x}}}$, $\mathbf{x} \in bnd(\cup\mathcal{B})$, defines the Voronoi diagram with respect to the system of disks $\mathcal{B}$, i.e. the circumcircle centers form curves that tessellate the plane into regions reflecting proximity with respect to one of the disks in $\mathcal{B}$. The latter Voronoi tessellation is known as the additively weighted Voronoi tessellation. We refer to [1], [4, Section 4.5.3], and references therein; a visualization is given in Figure 3(a,b). The relation between the generalized Delaunay triangulation and the Voronoi diagram with respect to the system of disks can be regarded as a generalization of the classical straight line duality (see, e.g., in [10]) between the Delaunay triangulation and the Voronoi tessellation [11] with respect to a set of points.

Note that, if the disks in $\mathcal{B}$ are of equal size, then the Voronoi tessellation with respect to $\mathcal{B}$ and the Voronoi tessellation with respect to disk centers coincide; see Figure 3(a). The generalized Delaunay triangulation, also known as triangle-neck partition in the special case of equally sized disks [2], and the classical Delaunay triangulation of the disk centers are combinatorially equal, i.e. they connect the same vertices. Moreover the cyclic polygons of the generalized triangulation are simply scaled versions of their classical counterparts. If the radii of the disks tend to zero we recover the classical Delaunay triangulation.

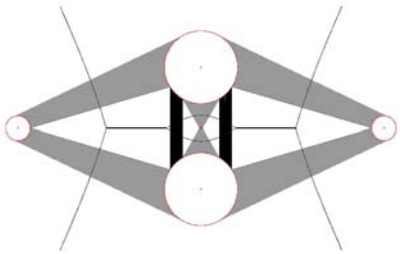
If the disks are not of equal size, the generalized Delaunay triangulation is not induced by the Delaunay triangulation of the disk centers since combinatorial changes might appear as it can be observed in Figure 3(b). Similarly, the Voronoi tessellation with respect to $\mathcal{B}$ and the Voronoi tessellation with respect to the disk centers do not coincide. In addition, Figure 3(c) illustrates that Voronoi edges, i.e. 1-dimensional in-



(a) A system of equally sized disks: The generalized Delaunay triangulation (left) coincides with Delaunay triangulation of the disk centers (right, bold lines) with regard to combinatorics. Weighted Voronoi (left, thin line segments) and classical Voronoi (right, thin line segments) are equal.



(b) A system of non-equally sized disks: No combinatorial equivalence relation between generalized Delaunay triangulation (left) and Delaunay triangulation of the disk centers (right, bold). Weighted Voronoi (left, thin line segments) and classical Voronoi (right, thin line segments) do not coincide either.



(c) Multiple connectivity of a Voronoi edge (bold line segments) and the (dual) generalized Delaunay edge (black shaded channels).

Fig. 3: Generalized and Classical Delaunay triangulation and their dual Voronoi tessellations.

tersections of neighboring Voronoi cells, might not be connected [7] which leads to multiple connectivity of the corresponding dual generalized Delaunay edge.

There are fast algorithms available for Voronoi diagrams with respect to a system of disks [5, 7, 6]. These algorithms, by duality, can also be employed for the computation of generalized Delaunay triangulation.

5 Conclusion

We have introduced a subdivision of the convex hull of a union of disks which contains the disks itself as elements. Further elements are simple geometric objects, i.e. generalized edges and cyclic polygons. The number of elements of the subdivision is of order $card(\mathcal{B})$ which is minimal in comparison to the descriptional complexity of $\mathcal{B}$.

The new generalized triangulations further inspire the design of new finite element methods perfectly fitted to the difficulties and requirements originating from the complicated geometries of random composite materials.

Our approach is not restricted to systems of disks; generalizations to multidimensional systems of convex sets, and therefore to practically relevant materials, are straight forward. In addition, the observations from Section 4 motivate equivalent definitions of generalized Delaunay triangulations based on duality.

References

[1] F. Aurenhammer and R. Klein. Voronoi diagrams. In *Handbook of computational geometry*, pages 201–290. North-Holland, Amsterdam, 2000.

[2] L. Berlyand and A. Kolpakov. Network approximation in the limit of small interparticle distance of the effective properties of a high-contrast random dispersed composite. *Arch. Ration. Mech. Anal.*, 159(3):179–227, 2001.

[3] B. Delaunay. Sur la sphère vide. *Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskikh i Estestvennykh Nauk*, 7:793–800, 1934.

[4] H. Edelsbrunner. *Geometry and topology for mesh generation*, volume 7 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, 2006. Reprint of the 2001 original.

[5] S. Fortune. A sweepline algorithm for Voronoĭ diagrams. *Algorithmica*, 2(2):153–174, 1987.

[6] M. Gavrilova and J. Rokne. Swap conditions for dynamic Voronoi diagrams for circles and line segments. *Comput. Aided Geom. Design*, 16(2):89–106, 1999.

[7] D.-S. Kim, D. Kim, and K. Sugihara. Voronoi diagram of a circle set from Voronoi diagram of a point set. I. Topology. *Comput. Aided Geom. Design*, 18(6):541–562, 2001.

[8] D. Peterseim. Finite element analysis of particle-reinforced composites. In C. Carstensen, editor, *Multiscale Methods*, volume 25 of *Oberwolfach Reports*. Mathematisches Forschungsinstitut Oberwolfach, 2009.

[9] D. Peterseim and C. Carstensen. Finite element analysis of particle-reinforced randomly dispersed composites. *(in preparation)*, 2010.

[10] F. P. Preparata and M. I. Shamos. *Computational geometry*. Texts and Monographs in Computer Science. Springer-Verlag, New York, 1985. An introduction.

[11] G. F. Voronoi. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. *Journal für die Reine und Angewandte Mathematik*, 133:97–178, 1907.

One-Reporting Queries

Saladi Rahul* K.S.Rajan†

Abstract

We are given a set S of n points in a d-dimensional space. Given a query orthogonal box q=$\Pi_{i=1}^{d}[a_i, b_i]$ or a semi-infinite query box q=$\Pi_{i=1}^{d}[a_i, \infty)$, a *One-reporting query* reports YES if $S \cap q \neq \phi$, else reports NO. The data structures introduced in this paper do not have their query time exponentially dependent on the dimension size, d, if the points are assumed to be randomly generated. Hence we partially break the "curse of dimensionality" with which almost all the data structures for range searching sufffer. The model of computation assumed is word-RAM.

1 Introduction

In a typical *orthogonal range reporting* problem we store a set of n d-dimensional points S in a data structure so that for an arbitrary d-dimensional query orthogonal box q= $[a_1, b_1]\times$ $[a_2, b_2]\times \ldots [a_d, b_d]$, all the points lying in $S \cap q$ need to be reported efficiently. *One-reporting or Emptiness* queries are a special case of range-searching problems where in for a query q we report YES iff $S \cap q \neq \phi$, else we report NO. In other words, given a query box q, we need to check if there exist any point in S which lies inside q.

A typical orthognal range query would be to "Find out all the employees in a company of age between 30 and 40 years with income being between $\$10,000$ and $\$20,000$". Range Searching problem has tremendous applications and has been widely studied for the past few decades [9]. Survery by Agarwal [2] and the book by M. De Berg et. al [7] are good sources for survey.

Our model of computation is the RAM model as modified by Fredman and Willard [12]. In this model it is assumed that each word is of size w and that the number of data elements n never exceeds 2^w, that is, $w \geq \log_2 n$. In addition, arithmetic and bitwise logical operations take constant time.

2 Comparision with previous results

There has been very little work done exclusively on "One-reporting" problem except for [11]. Dube et. al. [11] initiated the work on "One-reporting" problem for queries of the form $\Pi_{i=1}^{d}[a_i, \infty)$. The general line of attack to solve this problem is to build a data structure which does *range reporting* and for a given query say "YES" once a point gets reported. If no point gets reported, say "NO". Due to lack of previous work on this problem, we compare our data structures with previous structures which did range searching for query orthogonal box. We breifly review the existing range searching data structures.

Range Trees [6] take up $O(n \log^{d-1} n)$ space and "One-emptiness" queries can be answered by it in $O(\log^{d} n)$ time. The query time can be reduced to $O(\log^{d-1} n)$ by applying fractional cascading technique [10]. Chazelle et. al. [9] further improved the space of range tree. Specifically, for word-RAM model the following structures exist. Alstrup et. al. [3] came up with a structure that answered queries in $O(\log^{d-2} n/(\log\log n)^{d-3})$ time using $O(n \log^{d-2+\epsilon} n)$ space, $\epsilon > 0$. The query time was improved by Nekrich [14] to $O(\log^{d-3} n/(\log\log n)^{d-5})$ but with an increase in space to $O(n \log^{d+1+\epsilon})$. Later, Afshani [1] reduced the space to $O(n \log^{d+\epsilon} n)$. Recently, Karpinski et. al. [13] gave a structure which uses $O(n \log^{d-2+\epsilon})$ space and answers query in $O(\log^{d-3} n/(\log\log n)^{d-6})$ time.

The *primary focus* in this paper was to come up with structures which answer "One-reporting" queries fast. The performance of the structures in this paper are *dependent on the distribution of the points*. The performance of a standard range searching data structure is unaffected by the distribution of the points as can be seen in Table 1. In Table 1, "Best" refers to those configurations of points which leads to the best possible performance of our structure. "Average" refers to the case where the points are assumed to be generated randomly. Finally, "Worst" refers to those configurations which leads to the poorest possible performance of our structure.

As can be seen in the table, our structure clearly outperforms the existing structures in the "best" and "average" case scenario. Even in the "worst" case our structure either performs better or far off from some of the structures by only a small fraction. The efficiency of our structures can be claimed from the observation that a "worst" possible configuration of the points will not happen frequently. Performance of our structure is mentioned in Theorem 9.

*Lab for Spatial Informatics, IIIT-Hyderabad, India, saladi.rahul@gmail.com

†Lab for Spatial Informatics, IIIT-Hyderabad, India, rajan@iiit.ac.in

Source	Best	Average	Worst
New	$O(\frac{\log n}{\log\log n})$	$O(\frac{\log n}{\log\log n})$	$O((\frac{\log n}{\log\log n})^{d-1})$
[13]	$O(\log^{d-3} n/(\log\log n)^{d-6})$		
[1, 14]	$O(\log^{d-3} n/(\log\log n)^{d-5})$		
[3]	$O(\log^{d-2} n/(\log\log n)^{d-3})$		
[8]	$O(\log^{d-2} n)$		
[10, 9]	$O(\log^{d-1} n)$		
[6]	$O(\log^{d} n)$		

Table 1: The query time of the data structures in the best, average and the worst case scenario for $d>3$ are mentioned. Queries are of the form $\Pi_{i=1}^{d}[a_i, b_i]$.

3 Maximal Points

We start of by defining and discussing some interesting properties of Maximal Points. Maximal Points form the basis for solving the One-Reporting Problem in a d-dimensional space. A couple of definitions follow next.

Definition 1 ***(Dominance)***. *Let p_1=$(x_1, \ldots, x_d)$ and p_2=$(y_1, \ldots, y_d)$ be two d-dimensional points. If $x_i > y_i$, $\forall 1 \leq i \leq d$, then we define that p_1 dominates p_2 and that p_2 is dominated by p_1.*

Definition 2 ***(Maximal Point)***. *Let S be a set of points. A point $p_i \in S$ is a maximal point if there is no other point $p_j \in S$, such that p_j dominates p_i.*

For the given point set S, we denote M ($\subseteq S$) to be the set of maximal points.

Lemma 1 *For a point set S and a query quadrant q=$\Pi_{i=1}^{d}[a_i, \infty)$, $S \cap q \neq \phi$ iff $M \cap q \neq \phi$. Therefore, it is enough to consider only the maximal points ($M \subseteq S$) for answering "One-reporting dominance problem".*

4 One-reporting dominance queries in $\mathbb{R}^2$

We start building our solutions by considering dominance queries in $\mathbb{R}^2$. Specifically, for a point set S in $\mathbb{R}^2$ and a given query quadrant q=$[a_1, \infty)\times[a_2, \infty)$, we need to report YES if $S \cap q \neq \phi$, else report NO.

Lemma 2 *For a point set S, let M be the set of maximal points. Consider a point $p_i(i_x, i_y) \in M$. Now all the points $p_j(j_x, j_y) \in M$, which satisfy the condition $j_x > i_x$, will lie below the line $y = i_y$.*

Firstly, following Lemma 1 we find out the set of maximal points M in S [7]. A *Static Fusion Tree*, T, [12] is built based on the x-coordinates of the maximal points M. Given a query q=$[a_1, \infty)\times[a_2, \infty)$, we perform a *successor* query on T with a_1. Let $p(p_x, p_y)$ be the point reported. Clearly, $p_x \geq a_1$. If $p_y \geq a_2$, then we report YES. Else if $p_y < a_2$, then from Lemma 2 it can be inferred that the other points in M which have x-coordinate $\geq a_1$, will also have their y-coordinate values $< a_2$. Hence, NO will be reported. Also, while performing a *successor* query if no point is found, report NO.

Theorem 3 *A set S of n points in $\mathbb{R}^2$ having m maximal points can be preprocessed into a data structure of size $O(m)$, such that given a query quadrant q=$[a_1, \infty)\times[a_2, \infty)$, the "One-reporting dominance" query can be answered in $O(\log m/\log\log n)$ time.*

If the points in S are assumed to be randomly generated on a plane, then expected (or average) number of maximal points is $O(\log n)$ [5]. In the best case the no. of maximal points will be $O(1)$ and in the worst case the no. of maximal points will be $O(n)$. This leads to the following corollary.

Corollary 4 *Let $S(n)$ and $Q(n)$ denote the space and the query time of the above data structure. Then the following results can be inferred from the above discussion :-*

1. *$S(n)$=$O(1)$ and $Q(n)$=$O(1)$, in the best case.*
2. *$S(n)$=$O(\log n)$ and $Q(n)$=$O(1)$, in the average (or expected) case.*
3. *$S(n)$=$O(n)$ and $Q(n)$=$O(\log n/\log\log n)$, in the worst case.*

5 One-reporting Dominance problem in $\mathbb{R}^3$

In this section we provide a solution to the One-reporting dominance problem on $\mathbb{R}^3$ (referred to as xyz-space in this section). First, we find out the maximal points (M) of S, w.r.t., xyz-space [7]. The primary structure will be a Static Fusion Tree, D, built based on the z-coordinates of the points in M. The points in D are stored in non-decreasing order of their z-coordinate values. $p(v)$ denotes the set of points lying in the subtree rooted at an internal node $v \in D$. For an arbitrary internal node $v \in D$, let v_1,v_2,...,v_k be its children from left to right. With each child node v_i, we associate a point set P_i as follows : P_i=$\bigcup_{j=i}^{k} p(v_j)$. Each point set P_i is projected onto the xy plane and based on the xy-projections of points in P_i, its maximal points M_i are found out (the z-coordinates are ignored). At node v_i, based on these maximal points M_i we build a secondary structure of Theorem 3 to handle "One-reporting" dominance queries in the plane.

Given a query $q = [a_1, \infty) * [a_2, \infty) * [a_3, \infty)$, we first run a *successor* query on the primary structure of D with a_3. Let v be the leaf node selected. Now we choose some cannonical nodes in D and on

each of them a secondary query is performed. Let C=v_1,v_2,...,v_l, where v_i is the immediate right sibling of the i^{th} node on the path from v to the root, excluding v and *root*. v_i will not exist if this i^{th} node turns out to be the rightmost child. Then $C \cup v$ form our cannonical nodes. We query the secondary structures at each of the cannonical nodes with $q' = [a_1, \infty) * [a_2, \infty)$. If any of the secondary structures reports the presence of a point in q', then we say YES, else we say NO.

Theorem 5 *A set S of n points in $\mathbb{R}^3$ having m maximal points can be preprocessed into a data structure of size $O(m \times \frac{\log m}{\log\log n} \times \log^{1/5} n)$, such that given a query q=$[a_1, \infty)$×$[a_2, \infty)$×$[a_3, \infty)$, the "One-reporting dominance" query can be answered in $O((\log m/\log\log n)^2)$ time.*

If the points in S are assumed to be randomly generated on a xyz-space,then the expected (or average) number of maximal points is $O(\log^2 n)$ [5]. In the best case the no. of maximal points will be $O(1)$ and in the worst case the no. of maximal points will be $O(n)$. This leads to the following corollary.

Corollary 6 *Let $S(n)$ and $Q(n)$ denote the space and the query time of the above data structure. The following results can be inferred:-*

1. *$S(n)$=$O(1)$ and $Q(n)$=$O(1)$, in the best case.*
2. *$S(n)$=$O(\log^{11/5} n)$ and $Q(n)$=$O(1)$, in the average (or expected) case.*
3. *$S(n)$=$O(n(\log^{6/5} n/\log\log n))$, $Q(n)$=$O((\log n/\log\log n)^2)$, in the worst case.*

6 One-reporting dominance problem in $\mathbb{R}^d$, $d > 3$

In this section we shall generalize the the solution built for $d = 3$ to higher dimensional points as well. Assume that we already have a data structure for this problem in $\mathbb{R}^{d-1}$. The construction is shown in an inductive manner. Let x_1, x_2, ...,x_d be the individual coordinates of our d-dimensional space. First, we find out the maximal points (M) of S, w.r.t., $\mathbb{R}^d$ [7]. The primary structure will be a Static Fusion Tree, D, built based on the x_d-coordinates of the points in M. Then as done for $d = 3$, at each internal node an instance of the "One-reporting dominace" problem for $\mathbb{R}^{d-1}$ is built. The query algorithm is similar to the one described in the previous section.

If the points in S are assumed to be randomly generated in $\mathbb{R}^d$, then expected (or average) number of maximal points is $O(\log^{d-1} n)$ [5]. In the best case the no. of maximal points will be $O(1)$ and in the worst case the no. of maximal points will be $O(n)$. This leads to the following corollary.

Corollary 7 *A set S of n points in $\mathbb{R}^d$ having m maximal points can be preprocessed into a data structure of size $O(S(n))$, such that given a query region $\Pi_{i=1}^{d}[a_i, \infty)$, the "One-reporting dominance" query can be answered in $O(Q(n))$ time, where*

1. *$S(n)$=$O(1)$ and $Q(n)$=$O(1)$, in the best case.*
2. *$S(n)$=$O(\log^{d-4/5} n)$ and $Q(n)$=$O(1)$, in the average (or expected) case.*
3. *$S(n)$=$O(n(\log^{6/5} n/\log\log n)^{d-2})$, $Q(n)$=$O((\log n/\log\log n)^{d-1})$, in the worst case.*

7 One-reporting for bounded orthogonal rectangle queries on $\mathbb{R}^2$

Now, we generalize our queries to orthogonal bounded rectangles on $\mathbb{R}^2$. First we consider queries of the form q'=$[a_1, b_1] \times [a_2, \infty)$. The solution is based on the structure built in Theorem 3.

Based on the x-coordinates of the points in S we build a static fusion tree SFT. We store the points of S sorted by x-coordinate at the leaves of a complete balanced binary tree T'. At each internal node v, we store an instance of the structure of Theorem 3 for handling queries of the form $[a_1, \infty) \times [a_2, \infty)$ (resp., $(-\infty, b_1]$×$[a_2, \infty)$) built on the points in v's left (resp., right) subtree. Let $X(v)$ denote the average of the x-coordinate in the rightmost leaf in v's left subtree and the x-coordinate in the leftmost leaf in v's right subtree; for a leaf we take $X(v)$ to the x-coordinate of the point stored at v. The root of T' holds a pointer to SFT.

Given a query q', we first find out the successor of a_1 (say a'_1) and the predecessor of b_1 (say b'_1) in SFT. Leaf nodes of points a'_1 and b'_1 are found out in the primary structure of T. This identification can be done in O(1) time if we maintain appropriate pointers. Since T is a complete balanced binary tree, the LCA (least common ancestor) of a'_1 and b'_1 can found out in $O(1)$ time. Let the LCA be v. Then we query the structures at v using the NE-quadrant and the NW-quadrant derived from q' (i.e. the quadrants with corners at (a_1, a_2) and (b_1, a_2), respectively). If any of the structure reports YES, then the overall answer will be YES. If both the structures report NO, then the overall answer will be NO.

To solve the problem for general bounded orthogonal rectangles q=$[a_1, b_1] \times [a_2, b_2]$, we use the above approach again, but now the points in the tree T are stored by sorted y-coordinates. At each internal node v of T, we store an instance of the data structure above to answer queries of the form $[a_1, b_1] \times [a_2, \infty)$ (resp. $[a_1, b_1] \times (-\infty, b_2]$) on the points in v's left (resp. right) subtree. Also, a static fusion tree is built based on the y-coordinates of the points in S and a

pointer from the root of T to this structure is maintained. The query strategy is similar to the previous one, but now we use the interval $[a_2, b_2]$ to search in T.

Theorem 8 *A set S of n points on $\mathbb{R}^2$ can be preprocessed into a data structure of size $O(S(n))$ such that given a query orthogonal box q=$[a_1, b_1]$×$[a_2, b_2]$, the "One-reporting problem" can be solved in $O(Q(n))$ time, where*

1. *$S(n)$=$O(n \log n)$ and $Q(n)$=$O(\log n / \log\log n)$, in the best case.*
2. *$S(n)$=$O(n \log n)$ and $Q(n)$=$O(\log n / \log\log n)$, in the average (or expected) case.*
3. *$S(n)$=$O(n \log^2 n)$ and $Q(n)$=$O(\log n / \log\log n)$, in the worst case.*

8 One-reporting queries for bounded orthogonal boxes in $\mathbb{R}^d$

Here we generalize our queries to orthogonal bounded boxes in $\mathbb{R}^d$, i.e., the query box q=$[a_1, b_1]$×$[a_2, b_2]$×…×$[a_d, b_d]$. The solution built here is an extention of the structure built in Theorem 8 for d=2. First we build a structure for handling queries of the form q'=$[a_1, b_1]$×$\Pi_{i=2}^{d}[a_i, \infty)$ by the same technique that was used in the previous section. In the same manner, we next build a structure for handling queries of the form q''=$[a_1, b_1]$×$[a_2, b_2]$×$\Pi_{i=3}^{d}[a_i, \infty)$. In this way we iteratively build a structure D which finally handles queries q=$\Pi_{i=1}^{d}[a_i, b_i]$. The space occupied by D will increase by a factor of $O(\log^{d-2} n)$ compared to the structure built for d=2 in Theorem 8.

Theorem 9 *A set S of n points on $\mathbb{R}^d$ can be preprocessed into a data structure of size $O(S(n))$ such that given a query orthogonal box q=$[a_1, b_1]$×$[a_2, b_2]$×…$[a_d, b_d]$, the "One-reporting problem" can be solved in $O(Q(n))$ time, where*

1. *$S(n)$=$O(n \log^{d-1} n)$,$Q(n)$=$O(\log n / \log\log n)$, in the best case.*
2. *$S(n)$=$O(n \log^{d-1} n)$,$Q(n)$=$O(\log n / \log\log n)$, in the average (or expected) case.*
3. *$S(n)$=$O(n \log^{d} n)$, $Q(n)$=$O((\log n / \log\log n)^{d-1})$, in the worst case.*

References

[1] Peyman Afshani, On Dominance Reporting in 3D, Proceedings of the 16th annual European symposium on Algorithms, September 15-17, 2008, Karlsruhe, Germany

[2] P. Agarwal. Range searching. In Handbook of Discrete and Computational Geometry, CRC Press. 1997.

[3] S. Alstrup, G.S. Brodal, T. Rauhe: New data structures for orthogonal range searching, FOCS 2000: Proceedings of the 41st annual symposium on foundations of computer science, Washington, DC, USA, p. 198. IEEE Computer Society, Los Alamitos (2000)

[4] J. L. Bentley and H. A. Maurer. Efficient worst-case data structures for range searching. Acta Informatica, 13(2):155-168, 1980.

[5] J. L. Bentley, H. T. Kung, M. Schkolnick, and C. D. Thompson. On the average number of maxima in a set of vectors and applications. Journal of the ACM, 25(4):536-543, October 1978.

[6] J.L.Bentley: Multidimensional Divide-and-Conquer. Commun. ACM 23, 214-229 (1980)

[7] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf. Computational Geometry, Springer Verlag, 2nd ed., 2000.

[8] P. Bozanis, N. Kitsios, C. Makris, A. Tsakalidis: New Results on Intersection Query Problems. The Computer Journal 40(1), 22-29 (1997)

[9] B.Chazelle. A functional approach to data structures and its use in multidimensional searching. SIAM Journal on Computing, 17(3):427-462, 1988.

[10] B. Chazelle, L.J. Guibas. Fractional Cascading: I. A Data Structuring Technique. Algorithmica 1, 133-162 (1986); see also ICALP 1985

[11] T. Dube. Dominance range-query: the one-reporting case. SCG '93: Proceedings of the ninth annual symposium on Computational geometry, 1993, 0-89791-582-8, 208-217.

[12] M. L. Fredman and D. E. Willard. Surpassing the information theoretic bound with fusion trees. Journal of Computer and System Sciences, 47:424-436, 1993.

[13] M. Karpinski, Y. Nekrich. Space Efficient Multidimensional Range Reporting. COCOON '09: Proceedings of the 15th Annual International Conference on Computing and Combinatorics, 2009, 978-3-642-02881-6, 215–224.

[14] Y. Nekrich: A Data Structure for Multi-Dimensional Range Reporting. In: Proc. SoCG 2007, pp. 344-353 (2007)

Partial Least-Squares Point Matching under Translations

Günter Rote*

Abstract

We consider the problem of translating a given *pattern set* B of size m, and matching every point of B to some point of a larger *ground set* A of size n in an injective way, minimizing the sum of the squared distances between matched points. We show that when B can only be translated along a line, there can be at most $m(n-m)+1$ different matchings as B moves along the line, and hence the optimal translation can be found in polynomial time.

1 Introduction

In the partial pattern matching problem we are looking for an occurrence of some pattern B as part of a larger structure A. In this paper, we consider the case when A and B are finite points sets in the plane of size n and m respectively. (The results extend to higher dimensions, but for simplicity, we remain in the plane.)

Thus, we are looking for a subset $A' \subset A$ of size m that is as similar to B as possible. In this paper we measure similarity by the sum of the squared distances between corresponding points in some bijective mapping between B and A'. In other words, we insist that every point of B is matched with a distinct point of A.

In addition we allow B to be translated by some vector t. Thus, we are trying to solve the following problem:

$$\begin{aligned} &\text{minimize} && f(\pi,t) := \sum_{i=1}^{m} \|(b_i+t)-a_{\pi(i)}\|^2 \qquad (1)\\ &\text{subject to} && \pi\colon B \to A, \text{ injective},\\ &&& t \in \mathbb{R}^2. \end{aligned}$$

Related Work. This is a rich area of research. See for example [7, 8] for the case of least-squares matching between two equal sets. See [1, 4, 5] for other distance measures.

*Freie Universität Berlin, Institut für Informatik, Takustraße 9, 14195 Berlin, Germany. rote@inf.fu-berlin.de

2 Basic Observations. The Partial Matching Subdivision

For a fixed assignment π, the objective function f can be rewritten in the form

$$\begin{aligned} f(\pi,t) &= \sum_{i=1}^{m} \|(b_i+t)-a_{\pi(i)}\|^2\\ &= \sum_{i=1}^{m} \|b_i-a_{\pi(i)}\|^2\\ &\qquad + \Big\langle t, \sum_{i=1}^{m}(b_i-a_{\pi(i)})\Big\rangle + m\|t\|^2\\ &= c_\pi + \langle t, d_\pi\rangle + m\|t\|^2, \qquad (2) \end{aligned}$$

for a constant $c_\pi \in \mathbb{R}$ and a vector $d_\pi \in \mathbb{R}^2$.

We can thus rewrite the objective function (1) as

$$\min_t F(t) + m\|t\|^2,$$

where

$$F(t) = \min_{\substack{\pi\colon B\to A\\ \pi \text{ injective}}} (c_\pi + \langle t, d_\pi\rangle)$$

For a given translation t, minimizing $f(\pi,t)$ over all π is equivalent to determining the minimum in the expression for $F(t)$, since the difference is the constant term $m\|t\|^2$. The function $F(t)$ is the minimum of a finite number of linear functions. The regions where the minimum is attained by a particular linear function is hence a convex polygonal region. We call the subdivions of the plane into these regions the *partial matching subdivision* $\mathcal{D}_{B,A}$:

Theorem 1 *The space of parameters $t \in \mathbb{R}^2$ is subdivided into finitely many polygonal regions R_π, $\pi \in \Pi_0$. For all values t in one region R_π the same optimum assignment π optimizes* (1) (*or the expression in $F(t)$*).

When B consists of a single point, the partial matching subdivision $\mathcal{D}_{B,A}$ is just the Voronoi diagram of A. When A is a large dense point set and B consists of few points that are relatively spread out the subdivision looks like an overlay of several translated copies of the Voronoi diagram of A, since each point of B is just independently matched to its nearest neighbor in A. At least, this is true as long as the points of B lie "within" the set A; when they move

far away, several points of B will have the same closest point, and they have to compete for the point to which they are matched. Unfortunately, I could not produce interesting illustrations of partial matching subdivisions so far.

3 Exploring the Parameter Space

Inside each region R_π, the function $f(\pi, t)$ is a convex quadratic function of t, and hence it can be optimized easily. Thus, the straightforward approach to solving (1) is to search all regions R_π and compute the optimum in each region.

For a fixed vector t, the problem (1) is a minimum-cost bipartite matching problem and can be solved in polynomial time, for example by using network flow techniques. In this way, one can find the region R_π to which a parameter t belongs. By parametric linear programming techniques, one can then find the boundaries of this region, and one can also determine the adjacent regions across the boundary edges.

The running time of this approach is, up to a polynomial factor, determined by the number of regions that are to be explored. The crucial question is therefore, how many regions R_π there are.

We know a polynomial bound only for a very restricted case: namely when the translations t are restricted to a line only. In other words, we consider the intersection of the partial matching subdivision with a line.

Theorem 2 *A line can intersect the interior of at most $1 + m(n - m)$ different regions of the partial matching subdivision $\mathcal{D}_{B,A}$, for $|A| = n$ and $|B| = m$.* □

For the special case $m = n$, this means that there is only one region, and we get the well-known fact that the least-squares assignment between two sets of equal size is independent of t [7], which is also obvious from the calculation leading to (2).

Proof. The problem of finding an optimal matching in (1) (for a fixed t) can be formulated as a network flow problem.

We are given an $m \times n$ *cost matrix* (c_{ij}) with $c_{ij} = \|(b_i + t) - a_j\|^2$

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \\ \text{subject to} \quad & \sum\nolimits_{j=1}^{n} x_{ij} = 1, \text{ for } i = 1, \ldots, m \\ & \sum\nolimits_{i=1}^{m} x_{ij} \le 1, \text{ for } j = 1, \ldots, n \\ & 0 \le x_{ij} \le 1 \end{aligned}$$

By network flow theory, there is an optimal solution with $x_{ij} \in \{0, 1\}$, and it represents an assignment where each row i is assigned to exactly one column j and each column j is assigned to at most one row i. (The special case where $m = n$ is the usual assignment problem.) Among the n points of A, there will be m matched and $n - m$ unmatched vertices. We denote the set of matched vertices by $M(x)$.

Now, if we change t continuously, the solution (x_{ij}) will at some point change to a different solution $(\bar{x}_{ij})$. Some vertices will become matched and others will become unmatched.

Lemma 3 *Let (x_{ij}) and $(\bar{x}_{ij})$ be optimal solutions for parameter values t and $\bar{t}$, respectively. Then there is a one-to-one matching σ between the points in $M(x) \setminus M(\bar{x})$ and the points in $M(\bar{x}) \setminus M(x)$ such that*

$$\langle a_{\sigma(j)} - a_j, \bar{t} - t \rangle \ge 0,$$

for all $j \in M(x) \setminus M(\bar{x})$ As a consequence, we have

$$\Big\langle \sum_{j \in M(\bar{x})} a_j - \sum_{j \in M(x)} a_j,\ \bar{t} - t \Big\rangle \ge 0 \qquad (3)$$

Proof. The difference $\bar{x} - x$ between two assignments can be decomposed into an edge-disjoint union of (a) alternating even-length cycles and (b) alternating paths of even length starting at a matched vertex a_- of (x_{ij}) and ending at an unmatched vertex a_+ of (x_{ij}). For each such path of type (b), the vertex a_+ will be matched in the new assignment, and the vertex a_- will become unmatched.

Now let $a_- = a_0, b_1, a_1, b_2, \ldots, a_{k-1}, b_k, a_k = a_+$ be such an alternating path or cycle (for $a_- = a_+$).

The cost difference $\Delta c = c(\bar{x}) - c(x)$ between the old matching x and the new matching $\bar{x}$ can be expressed as follows. In order to simplify notation, we have first written the formulas without translation ($t = 0$).

$$\begin{aligned} \Delta c &= \sum_{i=1}^{k} \|b_i - a_i\|^2 - \sum_{i=1}^{k} \|b_i - a_{i-1}\|^2 \\ &= \sum_{i=1}^{k} (\|b_i\| - 2\langle b_i, a_i\rangle + \|a_i\|^2) \\ &\qquad - \sum_{i=1}^{k} (\|b_i\| - 2\langle b_i, a_{i-1}\rangle + \|a_{i-1}\|^2) \\ &= \|a_+\|^2 - \|a_-\|^2 - 2\sum_{i=1}^{k} \langle b_i, a_i\rangle + 2\sum_{i=1}^{k} \langle b_i, a_{i-1}\rangle \\ &= \|a_+\|^2 - \|a_-\|^2 - 2\sum_{i=1}^{k} \langle b_i, a_i - a_{i-1}\rangle \end{aligned}$$

Now let us bring in the dependence on t and replace

b_i by $b_i + t$:

$$\begin{aligned}\Delta c(t) &= \|a_+\|^2 - \|a_-\|^2 - 2\sum_{i=1}^{k}\langle b_i + t, a_i - a_{i-1}\rangle \\ &= \|a_+\|^2 - \|a_-\|^2 - 2\sum_{i=1}^{k}\langle b_i, a_i - a_{i-1}\rangle \\ &\quad - 2\sum_{i=1}^{k}\langle t, a_i - a_{i-1}\rangle \\ &= \|a_+\|^2 - \|a_-\|^2 - 2\sum_{i=1}^{k}\langle b_i, a_i - a_{i-1}\rangle \\ &\quad - 2\langle t, a_+ - a_-\rangle\end{aligned}$$

The only term that depends on t is the last term $-2\langle t, a_+ - a_-\rangle$. Now if x is optimal at t, then $\Delta c(t)$ must be nonnegative; otherwise we could use the alternating path or cycle to obtain a better solution. Similarly, since $\bar{x}$ is optimal at $\bar{t}$, we must have $\Delta c(\bar{t}) \leq 0$. Thus we get $\Delta c(t) - \Delta c(\bar{t}) \geq 0$, or

$$\langle \bar{t} - t,\, a_+ - a_-\rangle \geq 0$$

If we add this relation for all alternating paths and cycles the form the difference $\bar{x} - x$, we obtain (3). (The alternating cycles give no contribution.) □

Now we can conclude the proof of the theorem. Let us vary t along a line in direction s. Lemma 3 tells us that, whenever the assignment changes, a matched point a_- can only be replaced by a new matched point a_+ with $\langle \bar{t} - t,\, a_+ - a_-\rangle > 0$, or in other words, $\langle a_+, s\rangle > \langle a_-, s\rangle$. If we sort the points a by $\langle a, s\rangle$, and classify the subsets $M(x)$ of matched points of A by the sum of the ranks in this order, this means that the sum of the ranks can only go up. The minimum sum of ranks is $\sum_{i=1}^{m} i = m(m+1)/2$, and the maximum sum of ranks is $\sum_{i=1}^{m}(n+1-i) = (n+1)m - m(m+1)/2$. Between these two extreme values, there can be only $(n-m)m$ changes. □

An example showing that the bound is tight can be easily constructed in one dimension already: the set A consists of n uniformly spaced points, and B consists of m points very close together (much closer than the spacing between the points of A).

4 Conclusion

Still, the most important question is open: is the complexity of the partial matching subdivision $\mathcal{D}_{B,A}$ bounded by a polynomial? It is possible that a bound can already be derived from Theorem 2.

Another question arises if we allow rotations. Even if A and B have the same size and we consider only the one-parameter family of rotations about a fixed point, there can be many different optimal assignments, No polynomial bound is known. This problem can be formulated as a special parametric assignment problem where the costs depend linearly on a parameter x. For this more general problem, a superpolynomial lower bound of the form $2^{\sqrt{n}}$ on the number of optimal assignments has been proved by Patricia Carstensen [3, 2], based on a construction of Zadeh [6].

References

[1] A. Bishnu, S. Das, S. C. Nandy, and B. B. Bhattacharya. Simple algorithms for partial point set pattern matching under rigid motion. *Pattern Recognition*, 39(9):1662–1671, 2006.

[2] P. J. Carstensen. Complexity of some parametric integer and network programming problems. *Mathematical Programming*, 26(1):64–75, 1983.

[3] P. J. Carstensen. *The Complexity of Some Problems in Parametric Linear and Combinatorial Programming.* PhD thesis, University of Michigan, 1983.

[4] A. Efrat, A. Itai, and M. Katz. Geometry helps in bottleneck matching and related problems. *Algorithmica*, 31:1–29, 2001.

[5] M. T. Goodrich, J. B. Mitchell, and M. W. Orletsky. Approximate geometric pattern matching under rigid motions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21:371–379, 1999.

[6] N. Zadeh. A bad network problem for the simplex method and other minimum cost flow algorithms. *Mathematical Programming*, 5(1):255–266, 1973.

[7] K. Zikan. The Frobenius metric in image registration. *ORSA Journal on Computing*, 3:169–172, 1991.

[8] K. Zikan and T. M. Silberberg. The Frobenius metric in image registration. In L. Shapiro and A. Rosenfeld, editors, *Computer Vision and Image Processing*, pages 385–420. Elsevier, 1992.

A new separation theorem with geometric applications

Farhad Shahrokhi
Department of Computer Science and Engineering, UNT
P.O.Box 13886, Denton, TX 76203-3886, USA farhad@cs.unt.edu

Abstract

Let $G = (V(G), E(G))$ be an undirected graph with a measure function μ assigning non-negative values to subgraphs H so that $\mu(H)$ does not exceed the clique cover number of H. When μ satisfies some additional natural conditions, we study the problem of separating G into two subgraphs, each with a measure of at most $2\mu(G)/3$ by removing a set of vertices that can be covered with a small number of cliques G. When $E(G) = E(G_1) \cap E(G_2)$, where $G_1 = (V(G_1), E(G_1))$ is a graph with $V(G_1) = V(G)$, and $G_2 = (V(G_2), E(G_2))$ is a chordal graph with $V(G_2) = V(G)$, we prove that there is a separator S that can be covered with $O(\sqrt{l\mu(G)})$ cliques in G, where $l = l(G, G_1)$ is a parameter similar to the bandwidth, which arises from the linear orderings of cliques covers in G_1. The results and the methods are then used to obtain exact and approximate algorithms which significantly improve some of the past results for several well known NP-hard geometric problems. In addition, the methods involve introducing new concepts and hence may be of an indepandant interest.

1 Introduction and Summary

Separation theorems have shown to play a key role in the design of the divide and conquer algorithms, as well as solving extremal problems in combinatorial topology and geometry. The earliest result in this area is a result of Lipton and Tarjan [10] that asserts any n vertex planar graph can be separated into two subgraphs with at most $\frac{2n}{3}$ vertices by removing only $O(\sqrt{n})$ vertices. This result is extended by many authors including Miller et al [11], Fox and Pach [6], [7], and Chan [3].

Clearly if a graph contain a large clique, then it can not have a separation property that resembles the planar case. Fox and Pach [6, 7] have recently studied the string graphs which contain the class of planar graphs, and have shown that when these graphs do not contain a $K_{t,t}$, of fixed size t, as a subgraph, then a suitable separator exits. Although this powerful result is extremely effective in solving extremal problems, its computational power is limited to graphs that do not contain a "large" complete bipartite subgraph. Chan [3] studied the problem of computing the packing and piercing numbers of fat objects in R^d, where the dimension d is fixed. He drastically improved the running time of the first polynomial time approximation scheme (PTAS) for packing of fat objects due to Erlebach et al [5], and also provided the first PTAS for the piercing problem of fat objects. Parts of Chan's [3] work involved proving a separation theorem with respect to the abstract concept of a measure on fat objects. Motivated by his work we have defined the notation of a measure in a more combinatorial fashion on graphs. Furthermore, we have proven a combinatorial separation theorem. It should however be noted that the results in [3] do not imply ours, and our results do not apply to the general fat objects.

Let μ be a function that assigns non-negative values to subgraphs of G. μ is called a measure function if the following hold.

(i) $\mu(H_1) \leq \mu(H_2)$, if $H_1 \subseteq H_2 \subseteq G$, (ii) $\mu(H_1 \cup H_2) \leq \mu(H_1) + \mu(H_2)$, if $H_1, H_2 \subseteq G$, (iii) $\mu(H_1 \cup H_2) = \mu(H_1) + \mu(H_2)$, if there are no edges between H_1 and H_2, and (iv) $\mu(H)$ does not exceed the clique

cover number of any $H \subseteq G$.

Central to our result is a length concept similar to the bandwidth. Let H be a graph with $V(G) = V(H)$ and $E(G) \subseteq E(H)$ and let $C = \{C_1, C_2, ..., C_k\}$ be a clique cover in H. For any $e = xy \in E(G), x \in C_l, y \in C_t$, define $l(e, G, H, C)$ to be $|t-l|$. Let $l(G, H, C)$ denote $\max_{e \in E(G)} l(e, G, H, C)$, and let $l(G, H)$ denote $\min_{C \in \mathcal{C}} l(G, H, C)$, where $\mathcal{C}$ denotes the set of all ordered clique covers in H. We refer to $l(G, H)$ as the *length* of G in H. It is important to note that when $l(G, H)$ is small, then G exhibits some nice separation properties. For instance, one can partition $V(G)$ into blocks of $l(G, H)$ consecutive cliques of H, and argue that removal of any block separates G. Particularly, when $l(G, H) = 1$, then one can separate G by removing one clique form H. Similar important concepts such as treewidth, pathwidth, and bandwidth have been introduced in the past [2], but none is identical to the concept of length introduced here. Clearly $l(G, G) \leq BW(G)$, where $BW(G)$ is the bandwidth of G. Moreover as we will see, there is a simple but important connection between $L(G, G)$ and the dimension of interval orders.

Recall that a chordal graph does not have a chordless cycle of length at least 4. Our main result which is Theorem 1 is a generalization of the result stated earlier in the abstract, to $p \geq 2$ graphs, where G_p is a chordal graph.

Theorem 1 *Let μ be a measure on $G = (V(G), E(G))$, and let $G_1, G_2, ..., G_p$ be graphs with $V(G_1) = V(G_2) =, ..., V(G_p) = V(G)$, $p \geq 2$ and $E(G) = \cap_{i=1}^{p} E(G_i)$ so that G_p is chordal. Then there is a vertex separator S in G whose removal separates G into two subgraph so that each subgraph has a measure of at most $2\mu(G)/3$. In addition, the induced graph of G on S can be covered with at most $2^p l^* {}^{\frac{p-1}{p}} \mu(G)^{\frac{p-1}{p}}$ many cliques from G, where $l^* = max_{1 \leq i \leq p-1} l(G, G_i)$.*

Proof of Theorem 1 combines the clique separation properties of chordal graphs and perfect elimination trees, together with the properties associated with the length of a graph. The theorem either finds a suitable clique separator in the chordal graph G_p, or identifies a graph G_j, for which the cardinality of the clique cover is large, and the separates G using length properties. The application of Theorem 1 to a specific problem normally requires to define $\mu(G)$ to be the size of a clique cover C in G, and $\mu(H)$ is defined to be the size of C restricted to H, where H is subgraph of G.

The time complexity of finding the separator depends on the structure of the measure and how fast we can compute the measure on any subgraph. In typical applications of interest with $p = 2$, the separation algorithm can be implemented to run better than $O(|V(G)|^2)$.

2 Applications

Proper applications of Theorem 1 gives rise to the following.

Theorem 2 *Let $G = (V(G), E(G))$ be the intersection graph of a set of axis parallel unit height rectangles in the plane. Then, a maximum independent set in G can be computed in $|(V(G)|^{O(\sqrt{\alpha(G)})}$, where $\alpha(G)$ is the independence number of G. Moreover, there is a PTAS that gives a $(1-\epsilon)-$approximate solution to $\alpha(G)$ in $|V(G)|^{O(\frac{1}{\epsilon})}$ time and requires $O(|V(G)|)^2$ storage.*

Proof sketch. For $R_1, R_2 \in V(G)$, define $R_1 \prec_1 R_2$, if there is a horizontal line L so that R_1 is above L and R_2 is below L. Likewise, define $R_1 \prec_2 R_2$, if there is a vertical line L so that R_1 is to the left of L and R_2 is to the right of L. Observe that G_i, the incomparability graphs for $\prec_i$ is an interval graph, $i = 1, 2$, and hence is chordal. It is further easy to verify that $l(G, G_1) = 1$. Finally, let C be a $2-$approximate solution for the clique cover number of G that also provides for a $1/2-$approximate solution to independence

number of G, and for any induced subgraph F, define $\mu(F)$ to be the restriction of C to F. (Note that $\mu(G)$ can be computed in $O(|V(G)|\log(|V(G)|))$ time [4].)

For obtaining the sub-exponential time algorithm one can adopt the method in [10] proposed for planar graphs, by enumerating independent sets inside of separator and then recursively applying Theorem 1 to G. For the PTAS, one can also use the original approach in [10] adopted by Chan [3], by recursively separating G, but terminating the recursion when for a subgraph F, $\mu(F) = O(\frac{1}{\epsilon}^2)$ and then applying the sub-exponential algorithm to F. □

Similarly, one can prove the following.

Theorem 3 *Let S be a set of axis parallel unit height rectangles in the plane. Then, the piercing number of S can be computed in $|S|^{O(\sqrt{P(S)})}$, where $P(S)$ is the piecing number of of S. Moreover, there is a PTAS that gives a $(1+\epsilon)-$approximate solution to $P(S)$ in $|S|^{O(\frac{1}{\epsilon})}$ time and requires $O(|S|)^2$ storage.*

Our sub-exponential time algorithms in Theorems 2 and 3 are the first ones for the unit hight rectangles, and we are not aware of any previous sub-exponential algorithms for these problem. Moreover, the storage requirement for the PTAS in Theorem 2 drastically improves upon $|V(G)|^{O(\frac{1}{\epsilon})}$ storage requirement of the best known previous algorithm in [1], due to Agarwal et al, that was combining dynamic programming with the shifting method. Finally the time complexity of PTAS in Theorem 3 drastically improves upon $|S|^{O(\frac{1}{\epsilon}^2)}$ in [4].

Theorem 4 *Let $P, |P| = n$ be a set of points in the plane. There is an algorithm for computing the minimum number of discs of unit diameter needed to cover all points in P that gives an answer in $n^{O(\sqrt{opt(P)})}$ time, and $O(n^2)$ storage where $opt(P)$ is the value of an optimal solution. Furthermore, there is a PTAS that gives a $(1+\epsilon)-$approximate solution in $n^{O(\frac{1}{\epsilon})}$ time, and $O(n^2)$ storage.*

Proof sketch. For graph G, let $V(G) = P$, and for any $x, y \in P$, if they of distance at most 1, then place $xy \in E(G)$. Next, as suggested in [9] consider a square n by n grid in the plane containing all the points, so that each cell in the grid is a unit square. Note that the grid can be placed so that no two points appear in the boundary of a cell. Define two interval orders $\prec_1$ and $\prec_2$ on $V(G)$ as follows. $x \prec_1 y$, if $xy \notin E(G)$ and x and y are in different vertical strips of the grid so that x is to the left of y. $x \prec_2 y$, if $xy \notin E(G)$ and there is horizontal line L in the plane so that x is above L and y is below L. Let G_i, $i = 1, 2$ be the incomparability interval graph associated with $\prec_i$, and note that points in any vertical strip of the grid constitute a clique of G_1 and hence $l(G, G_1) = 1$.

For any $xy \in E(G)$, $x, y \in E(G)$, place two discs in the plane that has x and y in its boundary and call the resulting multi-set of discs $\mathcal{C}$, and note that $|\mathcal{C}| = O(n^2)$. If G is disconnected, then we would solve the problem for each component, and take the union of the solutions, so we will assume that G is connected. Thus, we can assume with no loss of generality that any feasible solution C' for any $P' \subseteq P$ is a subset of $\mathcal{C}$, for otherwise we can replace any $D \in C - \mathcal{C}$ by one disc from $\mathcal{C}$. Furthermore, it is easy to construct a feasible solution C so that $|C| \leq c\beta(G)$, where $\beta(G)$ is the clique cover number of G and c is a constant no more than 16, in about $O(n^2)$ time. Thus $\beta(G) \leq opt(P) \leq |C| \leq 16\beta(G) \leq 16opt(P)$. For any induced subgraph F in G, define $\mu(F)$ to be $|C_F|$, or, the cardinality of the restriction of C to F, and note that $\beta(F) \leq \mu(F)$ and consequently Theorem 1 applies.

Finally follow the details in [10] and the previous theorems by noting that we will always select our disc cover solutions from $\mathcal{C}$. (Note that the time complexity of enumeration inside of the separator $n^{O(\sqrt{opt(P)})}$.) □

Our sub-exponential time algorithm in Theorem 4 is the first one for the covering problem of Hochbaum and Mass [9]. In addition the time complexity of

our PTAS drastically improves time complexity of the original algorithm that was $n^{O(\frac{1}{\epsilon}^2)}$ in [9].

Let $\prec$ be a partial order on a finite set S. The dimension of $\prec$, denoted by $dim(\prec)$, is the minimum number of total orders on S whose intersection is $\prec$ [12].

We finish this section by stating a simple theorem that establishes some connections between partial orders, the dimension of interval orders and the concept of length introduced here.

Theorem 5 *Let G be a graph.*

(i) If $l(G,G) = 1$, then G is an incomparability graph.

(ii) If G is an interval graph whose underlying interval order is $\prec$, then $dim(\prec) \leq l(G,G) + 2$.

Justification. For (i), let $(C_1, C_2, ..., C_k)$ be a clique cover of G so that for $x \in C_i$ and $y \in C_j$, we have $xy \in E(G)$ only if $|i-j| \leq 1$. Now for any $x, y \in V(G)$ with $x \in C_i$, $y \in C_j$ with $j > i$, define $x \prec y$, if and only, if $xy \notin E(G)$. It is easily seen that $\prec$ is partial order on $V(G)$ so that $\bar{G}$ or the complement of G is a comparability graph, and hence G is an incomparability graph. We omit the proof (ii). □

References

[1] Agarwal, P.K., Kreveld, M., Suri, S., Label placement by maximum independent sets in rectangles, Comput. Geometry:Theory and Appl. 11(3-4) 209-218, 1998.

[2] Bodlaender H.L, A Tourist Guide through Treewidth. Acta Cybern. 11(1-2) 1-22, 1993.

[3] Chan T., Polynomial-time approximation schemes for packing and piercing fat objects , Journal of Algorithms, 46(2), 178 - 189, 2003.

[4] Chan T., Mahmood A., Approximating the piercing number for unit-height rectangles. CCCG 2005, 15-18, 2005.

[5] Erlebach T., Jansen K. and Seidel E., Polynomial-time approximation schemes for geometric graphs. Proc. 12th ACM-SIAM Symposium on Discrete Algorithms (SODA'01), 671-679, 2001.

[6] Fox J., Pach J., A separator theorem for string graphs and its applications, Combinatorics, Probability and Computing, 2009.

[7] Fox J., Pach J., Separator theorems and Turn-type results for planar intersection graphs, Advances in Mathematics 219, 1070-1080, 2008.

[8] Harry B. Hunt III, Marathe M. V., Radhakrishnan V., Ravi S. S, Rosenkrantz D. J., and Stearns R. E.. A Unified Approach to Approximation Schemes for NP- and PSPACE-Hard Problems for Geometric Graphs. Journal of Algorithms, 26, 135-149, 1996.

[9] Hochbaum D.S., Maass W., Approximation Schemes for Covering and Packing Problems in Image Processing and VLSI. Journal of the Association for Computing Machinery, 32(1), 130-136, 1985.

[10] Lipton, R.J., Tarjan, R.E., Applications of a planar separator theorem, SIAM J. Comput., 9(3),615-628, 1980.

[11] Miller, G.L., Teng, S., Thurston W., and Vavasis, S. A., Separators for Sphere-Packings and Nearest Neighborhood graphs, JACM, 44(1), 1-29, 1997.

[12] Trotter, W.T., New perspectives on Interval Orders and Interval Graphs, Surveys in combinatorics, 1997.

Towards Non-Uniform Geometric Matchings*

Christian Knauer † Klaus Kriegel † Fabian Stehn †

Abstract

Geometric matching problems are a well studied topic in computational geometry: Given two geometric objects P (the pattern) and Q (the model) and a transformation class $\mathcal{T}$, find a transformation $t \in \mathcal{T}$ that minimizes the distance between $t(P)$ and Q with respect to a given distance measure (e.g., the Fréchet distance or the Hausdorff distance).

In this abstract we introduce a generalization of this problem to so-called *non-uniform geometric matchings*, where a set of transformations is computed instead of a single transformation. The principal idea is to decompose the space containing the pattern into disjoint cells and to compute locally valid transformations for each cell, in order to make the whole matching less dependent on local deformations. The transformations of the resulting set are mutually interdependent to control the degree of continuity of the whole mapping.

We present constant factor approximations and an approximation scheme for point sequences under translations.

1 Introduction

In a geometric matching problem, one asks for a transformation t that minimizes the distance (value of on objective function) of a geometric object P (called pattern) transformed by t to a geometric object Q (called model). Geometric matching problems are among the most intensely studied fields in computational geometry, see [1] for a survey of this topic.

Geometric matchings are often used to align spaces e.g., in medical navigation systems. Here, the task is to find a mapping from a pattern space (the operation theatre) into a model space (containing a 3D-model of the relevant anatomic area of the patient). Such a mapping, in this context also called registration, is often computed by matching geometric features that are measured from both spaces. Using a single transformation of a usual transformation class to register the two spaces has the drawback that local deformations and non-uniform disturbances of the measured features influence the entire mapping.

*supported by the German Research Foundation (DFG), grant KN 591/2-2

†Institut für Informatik, Freie Universität Berlin, {knauer|kriegel|stehn}@inf.fu-berlin.de

In this abstract we introduce a generalization of the geometric matching concept to so-called *non-uniform geometric matchings* that address the stated problems by computing *a set of transformations*. The basic strategy is to partition the pattern space into regions of interest and to compute one transformation for each cell. A registration process then consists of two steps, first determining the cell that contains the point that is to be mapped and then applying the transformation associated with this cell. Non-uniform registrations have to optimize two competing objectives: to match the pattern features close to the model features while simultaneously assuring conformity of the mapping by demanding that transformations of two neighbored cells are "similar" (with respect to their effect).

1.1 Problem Definition

The input for an usual geometric matching problem is a class $\mathcal{G}$ of geometric objects, a pattern $P \in \mathcal{G}$, a model $Q \in \mathcal{G}$, a distance measure $\epsilon : \mathcal{G} \times \mathcal{G} \to \mathbb{R}^+$ and a transformation class $\mathcal{T}$ admissible on $\mathcal{G}$. For non-uniform geometric matchings we additionally have a partition $C = \{C_1, \ldots, C_k\}$ of the pattern space so that for all $i \in [k] : C_i \cap P \in \mathcal{G} \cup \{\emptyset\}$ and a similarity measure $\delta : \mathcal{T}^k \to \mathbb{R}^+$ in transformation space.

The task of a non-uniform geometric matching is to compute a set $T = \{t_1, \ldots, t_k\} \subset \mathcal{T}$ of transformations minimizing

$$f(P, Q, T, C) = \max\left(\max_{i \in [k]} \epsilon(t_i(C_i \cap P), Q), \delta(T)\right).$$

The objective f consists of a distance measure ϵ in object space and a a similarity measure δ in transformation space. The measure δ depends on the considered transformation class and measures the degree of continuity of a set of transformations is with respect to the image of the registration.

2 Non-Uniform Matchings for Point Sequences

In this abstract we consider first variants of the non-uniform matching problem where the transformation class $\mathcal{T}$ is restricted to translations. We further assume the geometric features to be point sequences of equal size ($|P| = |Q| = n$) measured in the pattern space and defined in the model space. We also assume that the correspondence between the point sequences is known, that is, point p_i is mapped to q_i

for all $i \in [n]$. As the distance measure ϵ we consider the maximum Euclidean 1-to-1 distance, that is $\epsilon(A,B) := \max_{i\in[n]} \|a_i - b_i\|$, for $A = (a_1, \ldots, a_n)$ and $B = (b_1, \ldots, b_n)$. We consider decompositions of the pattern space into n cells so that each cell contains exactly one point of P (as it is the case e.g., for a Voronoi diagram of P). To bound the degree of continuity of the combined mapping, we want that two translations whose corresponding cells share parts of their boundary are similar with respect to the distance of their image points. To a measure of similarity of two translations t_a and t_b we consider the Euclidean distance $\|t_a(x) - t_b(x)\|$ of their images of a point $x \in \mathbb{R}^d$. From now on, we don't distinguish between a translation and its translation vector and measure the similarity two translations by the Euclidean norm of their translation vector difference $\|t_a - t_b\|$.

To make the similarity measure of the translation set independent from the actual partition C we introduce a graph $G = (T, E)$ called *neighborhood graph*. The vertex set of G are the translations T and $\{t_i, t_j\} \in E$, if the cells corresponding to translations t_i and t_j are adjoining. Note, that the edges of the neighborhood graph could also be selected by criteria other than by the adjacency of cells (i.e., could be manually chosen). The algorithms presented in this abstract do not require that the neighborhood graph resembles the partition of the pattern space.

To simplify notation, we introduce the symbol d_{ij} for $t_i, t_j \in T$ and set $d_{ij} = 1$ if $\{t_i, t_j\} \in E(G)$ and $d_{ij} = 0$ otherwise. As the measure δ for the similarity of the translation set T we take the maximum of the similarity of any two translations that are connected in G, i.e., $\delta(T, G) := \max_{i,j\in[n]} d_{ij}\|t_i - t_j\|$.

With this specifications we get the following non-uniform matching problem which we discuss in the remainder of this paper:

Problem 1 *Given P, Q and G as above, compute a sequence T of translations $(t_1, \ldots, t_n)$ minimizing*

$$f(P,Q,G,T) = \max\left(\max_{i\in[n]} \|t_i(p_i) - q_i\|, \max_{i,j\in[n]} d_{ij}\|t_i - t_j\|\right), \quad (1)$$

where $\|\cdot\|$ denotes the Euclidean norm. The first term accounts for the distance of the matched point set P to Q by considering the L_∞ norm of the vector $t_i(p_i) - q_i$.

One advantage of considering translations is that the distance of a matched point p_i to its corresponding point q_i and also the similarity of two translations can be measured in translation space. Consider the translations $s_i = q_i - p_i$ for $1 \le i \le n$ and let $S := (s_1, \ldots, s_n)$. The distance $\|t_i(p_i) - q_i\|$ for a point p_i matched with translation t_i to q_i can be expressed as the distance $\|s_i - t_i\|$, as

$$\|t_i(p_i) - q_i\| = \|t_i + p_i - q_i\| = \|q_i - p_i - t_i\| = \|s_i - t_i\|.$$

The problem of computing a non-uniform matching for point sequences under translations can also be formulated in the following way: Consider a straight line embedding of the graph $G' = (S \cup T, E')$ with $E' = \{\{s_i, t_i\} \mid i \in [n]\} \cup \{\{t_i, t_j\} \mid d_{ij} = 1\}$. As the vertices of G' represent translations, we call G' the *translation graph* of S. The edge set E' consists of two sorts of edges: (1) edges connecting two translations t_i and t_j indicating that they have to be similar, (2) n edges $\{s_i, t_i\}$ whose lengths measure the Euclidean distance of $t_i(p_i)$ to q_i. Note, that the positions of all $s \in S$ are already determined by the input. The problem of computing a non-uniform registration for point sequences can hence be formulated as:

Problem 2 *Find a placement for all $t \in T$ such that the length of the longest edge in the induced straight line embedding of G' is minimal.*

Convex Programming Formulation The problem of computing a non-uniform registration optimizing Equation 1 can be phrased as a convex optimization problem:

$$\begin{array}{lll} \text{minimize} & \epsilon & \\ \text{subject to} & \|s_i - t_i\| \le \epsilon, & i = 1, \ldots, n, \\ & d_{ij}\,\|t_i - t_j\| \le \epsilon & 1 \le i < j \le n. \end{array}$$

As any metric norm is convex and the maximum of two convex functions is also convex. Convex optimization problems (such as Problem 2) can be solved in polynomial time e.g., by using the interior-point or the ellipsoid method [2].

Due to space limitations, we present in this abstract only non-uniform matching problems that have a convex optimization formulation. In the full version of this paper we also discuss other variants, some of which being not convex. The geometric insights and approximation techniques presented here however can be adapted to this problems.

3 Constant Factor Approximations

Let T_{opt} be an optimal solution to Problem 2 and let $\text{OPT} := f(P, Q, G, T_{opt})$.

Theorem 1 *Choosing $t_i = q_i - p_i$ for $1 \le i \le n$ results in a 3-approximation of* OPT.

Proof. Assume T to be in optimal position. For any i and j with $d_{ij} = 1$ we have that $\|t_i - t_j\| \le \text{OPT}$ as well as $\|t_i - s_i\| \le \text{OPT}$ and $\|t_j - s_j\| \le \text{OPT}$. Moving t_i upon s_i and t_j upon s_j increases the distance $\|t_i - t_j\|$ by at most $2 \cdot \text{OPT}$ while setting the distances $\|t_i - s_i\|$ and $\|t_j - s_j\|$ to zero, hence $\|t_i - t_j\| \le 3 \cdot \text{OPT}$ for all i, j with $d_{ij} = 1$, see Figure 1. □

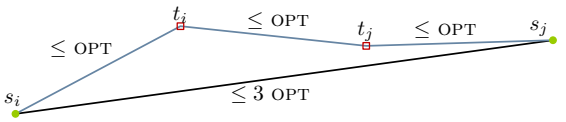


Fig. 1: Illustration of the 3-approximation of Theorem 1

3.1 Complete Graphs in the Plane

Assume P and Q to be point sequences in the plane and G to be complete, that is, any two translations have to be compared.

Lemma 2 *Let T_{opt} be an optimal choice of translations. The center c_{opt} of the smallest disc enclosing T_{opt} provides a $1 + 1/\sqrt{3}$ approximation for points in the plane and complete neighborhood graphs, if c_{opt} is chosen for all $t \in T$:*

$$f(P, Q, G, (t_1 = c_{opt}, \ldots, t_n = c_{opt})) \leq (1+1/\sqrt{3})\ \text{OPT}.$$

Proof. In optimal position, the distance $\|s_i - t_i\|$ for any $1 \leq i \leq n$ is bounded by OPT. All translations of T_{opt} lie within the smallest disc enclosing T_{opt} whose radius is bounded by $\text{OPT}/\sqrt{3}$, as stated in the following lemma (whose proof is omitted due to space limitations).

Lemma 3 *The radius of the smallest disc enclosing a point set of width μ in the plane is bounded by $\mu/\sqrt{3}$.*

The distance of each point $s \in S$ to c_{opt} is bounded by $\text{OPT} + 1/\sqrt{3}\,\text{OPT}$. Therefore, the center c_{opt} implies a $1 + 1/\sqrt{3}$ approximation as stated in Lemma 2. □

But as T_{opt} is unknown, the center c_{opt} of its smallest enclosing disc is unknown as well. On the other hand, the translation that *minimizes* the largest distance to any point of S can be computed in linear time [3]. It is easy to see, that the center c of the smallest disc enclosing S is the translation that minimizes the distance to any translation in S. We have determined the approximation factor for choosing $t_i = c_{opt}$ for $i \in [n]$ and know that c is the best possible choice of a single translation.

Theorem 4 *The center c of the smallest disc enclosing the point sequence S results in a $(1 + 1/\sqrt{3})$ approximation:*

$$f(P, Q, G, (t_1 = c, \ldots, t_n = c)) \leq (1 + 1/\sqrt{3})\,\text{OPT}.$$

The approximation factor can be improved to $2/3\,(1 + 1/\sqrt{3}) \approx 1.05157$ by choosing n different translations in the following way: Let APP be the value of the approximation as presented in Theorem 4. Choose t_i to be the intersection o_i of the straight line $\overline{s_i\, c}$ for $i \leq 1 \leq n$ with the circle δ_{app} centered in c with radius $\text{APP}/3$. If δ_{app} does not intersect the line segment $\overline{s_i\, c}$, then t_i is chosen to be s_i. For this choice of t_i, the distance $\|s_i - t_i\|$ is bounded by $2/3\ (1 + 1/\sqrt{3})\text{OPT}$ for each $i \in [n]$ which is also the diameter of the circle δ_{app}, implying that the distances $\|t_i - t_j\|$ for $i, j \in [n]$ are also bounded by $2/3\ (1 + 1/\sqrt{3})\text{OPT}$, see Figure 2.

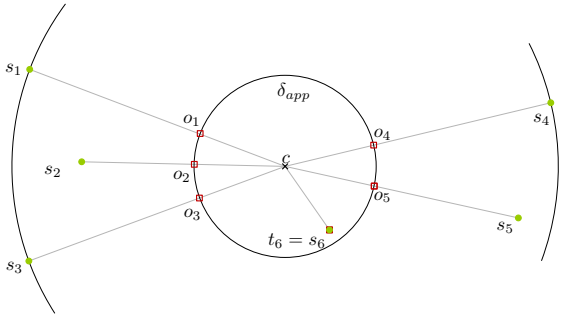


Fig. 2: The outer circle is the smallest circle enclosing S (radius APP), the inner circle δ_{app} has a radius of $\text{APP}/3$ with the same center.

Theorem 5 *Choosing the translations $t_1, \ldots, t_n$ in the described manner results in a $2/3\ (1 + 1/\sqrt{3})$ approximation that can be computed in linear time.*

4 A FPTAS for trees in the plane

In this section we present an approximation scheme for neighborhood graphs that are trees and point sequences in the plane. This approximation is based on a relaxed decision problem formulation. The decision variant of Problem 2 can be stated as:

Problem 3 *For a given $\mu \geq 0$ and a translation graph $G' = (S \cup T, E')$ that is a tree with $S, T \subset \mathbb{R}^2$, is there a placement of T s.t. the length of the longest edge in the induced straight line embedding of G' is at most μ?*

Let us first introduce some notation: we impose the structure of the neighborhood graph $G(T, E)$ on S and hence call s_i and s_j neighbored if $\{t_i, t_j\} \in E$. Let $\delta(c, r)$ be a disc of radius r centered in c and define $\delta_\mu := \delta((0, 0), \mu)$. The Minkowski sum $X \oplus Y$ of two sets X and Y is defined as $X \oplus Y := \{x+y \mid x \in X,\ y \in Y\}$. For X being a geometric figure and $Y = \delta_\mu$ the set $X \oplus \delta_\mu$ is the set of all points z so that there is a point $x \in X$ with $\|z - x\| \leq \mu$.

The following geometric observations hold for embeddings that meet the edge length constraint:

1. for all $t_i \in T$ we have that $t_i \in \delta(s_i, \mu)$
2. if $\{t_i, t_j\} \in E(G)$ then $t_i \in \delta(s_j, 2\mu)$ and $t_j \in \delta(s_i, 2\mu)$

3. if $c_1, \ldots, c_k$ are the children of s_i then $t_i \in \bigcap_{j\in[k]} \delta(c_j, 2\mu) \cap \delta(s_i, \mu)$

These observations lead to the definition of the *admissible region* $\mathrm{reg}_\mu(s)$ for a point $s \in S$, which is the set of all translations t for which a straight line embedding of the subtree rooted in s exists that satisfies the edge length constraint. The region $\mathrm{reg}_\mu(s)$ for a given μ is convex and can be described inductively: if s is a leaf in G then $\mathrm{reg}_\mu(s) = \delta(s, \mu)$. If s is an internal node with children $c_1, \ldots, c_k$, then

$$\mathrm{reg}_\mu(s) = \bigcap_{i\in[k]} \left(\mathrm{reg}_\mu(c_i) \oplus \delta_\mu\right) \cap \delta(s, \mu). \quad (2)$$

Let $r \in S$ be an arbitrarily chosen root of G. By construction we have that the answer to Problem 3 is "yes" exactly if $\mathrm{reg}_\mu(r) \neq \emptyset$.

Solving the decision problem exactly involves computing Minkowski sums of all admissible regions and their intersections. The boundary of an admissible region of a point s can in the worst case be defined by all "μ inflated" admissible regions of the children of s. Fortunately, it is not necessary to maintain the exact shape of the admissible regions to compute a $(1+\lambda)$ approximation. Instead, we approximate these regions by convex polygons $\widetilde{\mathrm{reg}}_\mu(s)$ so that $\vec{h}\left(\widetilde{\mathrm{reg}}_\mu(s), \mathrm{reg}_\mu(s)\right) \leq \epsilon$ and additionally $\mathrm{reg}_\mu(s) \subset \widetilde{\mathrm{reg}}_\mu(s)$, where $\vec{h}(A, B)$ is the directed Hausdorff distance from A to B. We also approximate the inflated admissible regions $\mathrm{reg}_\mu(s) \oplus \delta_\mu$ by convex polygons $\mathrm{infl}_\mu(s)$ so that $\mathrm{reg}_\mu(s) \oplus \delta_\mu \subset \mathrm{infl}_\mu(s)$ and $\vec{h}\left(\mathrm{infl}_\mu(s), \mathrm{reg}_\mu(s) \oplus \delta_\mu\right) \leq \epsilon$.

Relaxing the decision problem An algorithm $\mathcal{A}$ that uses the described inductive strategy implied by Equation 2 to decide Problem 3 for given $\mu \geq 0$ and $\epsilon > 0$ while maintaining the approximated instead of the exact regions returns

- FALSE for any $\mu < \mathrm{OPT} - \epsilon$,
- TRUE for any $\mu > \mathrm{OPT}$,
- either TRUE or FALSE if $\mu \in [\mathrm{OPT} - \epsilon, \mathrm{OPT})$.

Note, that two inflated approximative admissible regions $\mathrm{infl}_\mu(s)$ and $\mathrm{infl}_\mu(s')$ might intersect, even though $\mathrm{reg}_\mu(s) \oplus \delta_\mu \cap \mathrm{reg}_\mu(s') \oplus \delta_\mu = \emptyset$. Let $s \in S$ be an internal node of G and let $c_1, \ldots, c_k$ be the children of s. For any $t \in \widetilde{\mathrm{reg}}_\mu(s)$ we have that $\|t - s\| \leq \mu + \epsilon$ and $\forall i \in [k]\ \exists t' \in \widetilde{\mathrm{reg}}_\mu(c_i) : \|c_i - t'\| \leq \mu + \epsilon \wedge \|t - t'\| \leq \mu + \epsilon$.

Let APP′ be the value of the 3-approximation as described in Theorem 1, which gives us $\mathrm{APP}'/3 \leq \mathrm{OPT} \leq \mathrm{APP}'$. Set ϵ to $\lambda \cdot \mathrm{APP}'/3$. Consider an uniform sampling of the interval $[\mathrm{APP}'/3, \mathrm{APP}']$ with sample width ϵ (i.e., the distance of two consecutive samples is ϵ). The smallest sample μ' of the sample set for which the approximated admissible region of r is not empty satisfies that $|\mu - \mathrm{OPT}| \leq \lambda \cdot \mathrm{APP}'/3 < \lambda \cdot \mathrm{OPT}$, hence the embedding computed for the value μ' realizes a $(1+\lambda)$ approximation.

Theorem 6 *For G, P, Q given as before and a $\lambda > 0$, a sequence of translations T can be computed in $O\left((n\,1/\sqrt{\lambda})\,\log 1/\lambda\right)$ time so that*

$$f(P, Q, G, T) \leq (1+\lambda)\,\mathrm{OPT}.$$

Proof. Using binary search, it takes $O(\log 1/\lambda)$ time to find the smallest value μ' for which the approximated admissible region of r is not empty. A single relaxed decision problem for a $\mu \in [\mathrm{APP}'/3, \mathrm{APP}']$ can be decided in $O\left(n\sqrt{1/\lambda}\right)$ time: as shown by Rote [4], any convex planar figure can be approximated by a convex polygon that surrounds the figure and has $O\left(\sqrt{B/\epsilon}\right)$ points on its boundary and is in ϵ Hausdorff distance to the figure, where B is the length of the boundary of the figure. Any admissible region is defined as – or intersected by – a disc of radius μ and is inflated (by taking the Minkowski sum) by a disc of radius μ. Hence any admissible convex region can be covered by a disc of radius 2μ which bounds the length of the boundary of an admissible region to $4\pi\mu$. By choosing $\epsilon = \lambda \cdot \mathrm{APP}'/3$ we have that any inflated admissible region can be approximated by a convex polygon using $O\left(\sqrt{4\pi\mu/\lambda \cdot \mathrm{APP}'/3}\right) = O\left(1/\sqrt{\lambda}\right)$ vertices, as $\mu \leq \mathrm{APP}'$. Each region $\mathrm{infl}_\mu(s)$ for all nodes $s \in S \setminus \{r\}$ is intersected exactly once to gain the (approximated) admissible region of the parent of s. As shown by Toussaint [5], two convex polygons can be intersected in linear time, which leads to a total runtime of $O\left(n/\sqrt{\lambda}\right)$ to compute a single relaxed decision problem instance. □

References

[1] H. Alt and L.J. Guibas. Discrete geometric shapes: Matching, interpolation, and approximation. In *Handbook of Computational Geometry*, pages 121–153. Elsevier B.V., 2000.

[2] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, March 2004.

[3] Nimrod Megiddo. On the ball spanned by balls. *Discrete & Computational Geometry*, 4:605–610, 1989.

[4] G. Rote. The convergence rate of the sandwich algorithm for approximating convex functions. *Computing*, 48(3-4):337–361, 1992.

[5] Godfried T. Toussaint. A simple linear algorithm for intersecting convex polygons. *The Visual Computer*, 1:118–123, 1985.

How to Cope with Undesired Side Effects of Symbolic Perturbation

Shuhei Takahashi* Kokichi Sugihara†

Abstract

Although symbolic perturbation is a powerful technique for eliminating geometric degeneracy, it sometimes generates undesired side effects. This paper studies these side effects observed in the tetrahedral mesh generation problem and the segment intersection problem, and proposes strategies for avoiding them. On the basis of these individual strategies, we also consider a paradigm called "selective perturbation," which might be a general principle for coping with undesired side effects.

Key words Side effect, symbolic perturbation, selective perturbation, tetrahedral mesh, segment intersection

1 Introduction

Degeneracy in geometric problems is one of the most serious issues in the implementation of practical software, firstly because degeneracy requires exceptional processing branches in geometric algorithms, and secondly because degenerate situations, as well as nearly degenerate situations, often cause failures of geometric tests, and thus, cause geometric software to fail.

Indeed, degeneracy is one of the main sources of fragility of geometric software. In practical computers, computation is done in finite precision, and hence, numerical computation usually contains round-off errors. Because of these errors, which happen more often in degenerate situations, we sometimes misjudge the topological structures of the geometric objects .

In order to make geometric software robust against numerical errors and degeneracy, many approaches have been proposed over the last twenty years. They can be classified into three groups based on to what extent they rely on numerical values [6].

The first group of approaches relies on numerical values only moderately. The numerical computation in each geometric test is accompanied with error analysis, and according to the estimated error bounds the results of the test are divided into three cases: "yes," "no," and "unknown." If the result is "yes" or "no," the result is adopted.

The "unknown case" implies that the situation is degenerate or close to degeneracy, and the treatment depends on the particular situation of the problem being solved [3].

The second group of approaches does not rely on numerical values at all. The highest priority is placed on the preservation of purely topological properties, and numerical results are used only when they are consistent with the topological properties; thus the topological consistency is guaranteed no matter how large the numerical errors are in the computation. These approaches are called the topology-oriented approaches [4, 9].

The topology-oriented approaches avoid degeneracy automatically because we cannot recognize degeneracy in imprecise arithmetic, and hence, we need not treat degenerate cases. However, as this approach eliminates all the degenerate cases automatically, it is difficult to avoid undesired side effects.

The third group of approaches uses exact computation. Most geometric tests can be carried out exactly if we use sufficiently high precision, but still finite-precision, arithmetic. Employing this fact, we can always get correct answers to geometric tests, thus, avoiding inconsistency. This approach is called the exact-computation approach [5, 8, 12].

In exact computation, we recognize the degenerate cases exactly. Therefore, we have to prepare exceptional branches of processing for all degenerate cases. However, it is tedious work to design the exceptional processing for degenerate cases.

Symbolic perturbation is a technique to circumvent this difficulty. In this technique, input numerical data are replaced with polynomials of a symbol, which is intuitively interpreted as an infinitesimal small number. If the input is not degenerate, the replacement does not give any essential effect to the original problem. If the input is degenerate, on the other hand, the input data are slightly perturbed and are changed to a nondegenerate case. Thus, symbolic perturbation is a powerful technique for eliminating degeneracy, resulting in simple and robust geometric software [2, 11].

In this way, both the topology-oriented approach and the exact-computation approach can eliminate degenerate cases. However, they eliminate all the degeneracy completely, and as the result of this, sometimes generate unwanted situations as a side effect.

For example, in Delaunay tetrahedrization, the perturbation of the coordinates of the vertices might generate zero volume tetrahedrons, which should be

*Department of Mathematical Informatics, University of Tokyo, shuhei@nya3.jp

†Meiji Institute for Advanced Study of Mathematical Sciences, Meiji University, kokichis@isc.meiji.ac.jp

avoided because they cause large approximation errors if we use it to the finite element method for solving partial differential equations. In a layout check of VLSI circuits, perturbation of wires might generate apparent intersections of wires although they are correctly placed.

For these situations we should not apply the topology-oriented approach or symbolic perturbation directly; instead we should perturb only some restricted aspects of degeneracy in order to avoid such unwanted side effects.

In this paper, we consider individual methods for avoiding side effects in the above two situations, and also discuss the possibility of a general principle for selective perturbation.

2 Review of Symbolic Perturbation

Let P be a geometric problem, and $\boldsymbol{d} = (d_1, \ldots, d_m)$ be a vector of real numbers that specifies the problem P. For example, if P is the problem of generating the Delaunay tetrahedrization for n points in a three-dimensional space, $\boldsymbol{d}$ is the vector consisting of the x, y, z coordinates of the n points: $\boldsymbol{d} = (x_1, y_1, z_1, \ldots, x_n, y_n, z_n)$.

Next, let $f(\boldsymbol{d})$ be a function of the input whose sign gives the answer to a geometric test used in an algorithm to solve P. We call $f(\boldsymbol{d})$ a *test function*. In Delaunay tetrahedrization, the most important test is sphere inclusion: given five points, $p_1, p_2, \ldots, p_5$, test whether the sphere passing through the first four points includes the fifth point in its interior.

For this test, we can use

$$f(p_1,p_2,p_3,p_4,p_5) = \begin{vmatrix} 1 & x_1 & y_1 & z_1 & {x_1}^2+{y_1}^2+{z_1}^2 \\ 1 & x_2 & y_2 & z_2 & {x_2}^2+{y_2}^2+{z_2}^2 \\ 1 & x_3 & y_3 & z_3 & {x_3}^2+{y_3}^2+{z_3}^2 \\ 1 & x_4 & y_4 & z_4 & {x_4}^2+{y_4}^2+{z_4}^2 \\ 1 & x_5 & y_5 & z_5 & {x_5}^2+{y_5}^2+{z_5}^2 \end{vmatrix}. \quad (1)$$

Indeed $f(p_1, p_2, p_3, p_4, p_5)$ changes its sign according to whether p_5 is inside this sphere or not.

In general, there are many test functions for an algorithm to solve P. If at least one test function satisfies $f(\boldsymbol{d}) = 0$, we say that the problem P is degenerate.

Let $\mathbf{Z}$ be the set of all integers, and let $\mathbf{Z}[\varepsilon]$ be the set of all polynomials of ε with integer coefficients. For $g(\varepsilon) = g_0 + g_1\varepsilon + g_2\varepsilon^2 + \cdots + g_k\varepsilon^k \in \mathbf{Z}[\varepsilon]$, we define the sign of $g(\varepsilon)$ as the sign of the coefficient of the lowest-degree term. That is, $\mathrm{sign}(g(\varepsilon)) = \mathrm{sign}(g_i)$ for i that satisfies $g_0 = g_1 = \cdots = g_{i-1} = 0$ and $g_i \neq 0$. It is known that the sign thus defined is consistent with the additions and multiplications in $\mathbf{Z}[\varepsilon]$, and $\mathbf{Z}[\varepsilon]$ with this sign forms an ordered ring.

Symbolic perturbation is to perturb the input by adding polynomials of ε and determine the sign of the test functions in the ordered ring $\mathbf{Z}[\varepsilon]$. We adjust the perturbation polynomials so that the test functions never vanish while guaranteeing the sign of the original input in nondegenerate cases. Thus we can construct software that eliminates all the degeneracy automatically.

3 Avoidance of Zero-Volume Delaunay Tetrahedrons

In tetrahedral mesh generation, Delaunay tetrahedrization is often preferred because it has many good properties and can be computed efficiently. However, if we apply symbolic perturbation, zero volume tetrahedrons, which we call *slivers*, may appear, typically when integer grid points are used for mesh nodes.

A typical perturbation is to replace the coordinates (x_i, y_i, z_i) of node p_i with a perturbed node

$$(x_i + \varepsilon^{M^i}, y_i + \varepsilon^{M^{n+i}}, z_i + \varepsilon^{M^{2n+i}}), \quad (2)$$

where n is the number of nodes and M is a sufficiently large integer. This intuitively means that the largest perturbation is given to the x coordinate, and each node has a different amount of perturbation.

Consider the eight nodes of a cube, which are frequently used in initial set of nodes for mesh generation. They are degenerate because they are on a common sphere, and the equation (1) vanishes for any five points. The perturbation above implies that the nodes are replaced in the x direction in different (still infinitesimal) distances, so that originally coplanar four points become noncoplanar and can form a Delaunay tetrahedron.

However, zero-volume tetrahedrons should definitely be avoided in applications such as finite element methods. So, we want to selectively eliminate such degeneracy that imposes exceptional processing on the algorithm while keeping the coplanarity of the original nodes.

We found that we can achieve this goal by changing the problem itself in the following way [7]. Delaunay tetrahedrization can be considered as the Laguerre-Delaunay tetrahedrization with all the weights 0. Hence we can perturb the weights. Then, the degeneracy (i.e., the cocircularity) is eliminated while coplanarity of the original nodes are kept as it is. Thus we can achieve the selective perturbation by perturbing the problem itself instead of perturbing the input data.

4 Selective Perturbation for Segment Intersection

Suppose that we are given a VLSI layout plan, and we want to guarantee that there is no unintended intersection between wires. Assume that the electric wires

comprise line segments, and they are connected only at their end points; if a segment contacts another at a midway, we divide the segment into two at that point. So, the electric wire layout is a collection of line segments with high degeneracy in the sense that they contact at end points. Our goal is to check whether segments contact at other than end points.

Suppose that each segment is represented by the coordinates of the end points. The segment intersection check can be done by the Bentley–Ottmann plane sweep method in $O(n \log n)$ time, where n is the number of line segments [1]. However, we cannot use symbolic perturbation directly, because the perturbation results in detecting false intersections and missing true intersections. For example, as shown in Fig. 1(a), consider the T-shape contact of segments. If the three segments contact at the center node, this is an allowed layout, but perturbation may generate a false intersection as shown in (b). On the other hand, if the horizontal line in (a) consists of one segment, the layout is not allowed, but perturbation may miss the detection of intersection as shown in (c). In this way symbolic perturbation generates unwanted side effects.

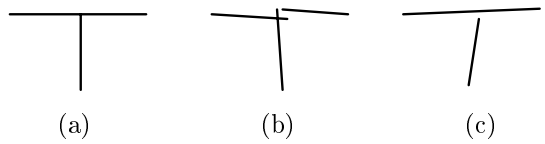


Figure 1: Line segments and their perturbation: (a) initial layout; (b) detection of false intersection; (c) missing of true intersection.

Our goal is to modify the situation so that there is no degeneracy (i.e., the plane sweep method does not require exceptional processing) and yet there is no false detection or no failure in detection of true intersections.

In order to avoid the false intersection shown in Fig. 1(b), the segments need to be shortened. On the other hand, in order to avoid the failure in detection of true intersection shown in Fig. 1(c), the segments need to be thickened. To fulfill these requirements, we replace each segment with four segments in the following way [10].

We sort the segments by their lengths from shorter to longer and renumber them from 1 to n. Then, we replace the i-th segment $[p_i, q_i]$ with the four segments:

$$[p_i + (q_i - p_i)\varepsilon^{n+i}, (p_i + q_i)/2 + R(q_i - p_i)\varepsilon^i], \quad (3)$$
$$[(p_i + q_i)/2 + 2R(q_i - p_i)\varepsilon^i, q_i + 2(p_i - q_i)\varepsilon^{n+i}], \quad (4)$$
$$[q_i + 2(p_i - q_i)\varepsilon^{n+i}, (p_i + q_i)/2 + R(q_i - p_i)\varepsilon^i], \quad (5)$$
$$[(p_i + q_i)/2 + 2R(p_i - q_i)\varepsilon^i, p_i + 2(q_i - p_i)\varepsilon^{n+i}], \quad (6)$$

where R represents the rotation of the vector by $\pi/2$.

Fig. 2 shows the effect of the replacement schematically. The four segments do not intersect, and together simulate the change of the original segment to a rhombus-like shape that is obtained by slightly shortening and thickening the original segment. Furthermore, shorter segments get larger perturbations than longer segments, and the amount of thickening (the term ε^i in eqs. (3)–(6)) is larger than the amount of shortening (the term ε^{n+i} in eqs. (3)–(6)). Therefore, we can schematically represent the perturbed version of a segment by a rhombus.

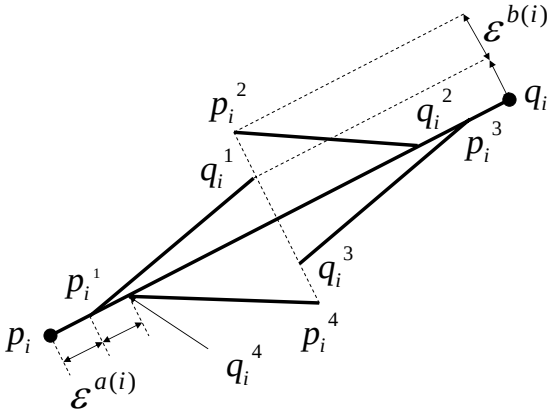


Figure 2: Replacement of a segment with four new segments, which together shorten and thicken the original segment.

Fig. 3 shows the resulting configurations of the rhombuses in various situations. Panels (a) and (b) show that an allowed contact in the original layout is converted to a collection of nonintersecting rhombuses. On the other hand, (c), (d) and (e) show that disallowed contacts are converted to intersecting rhombuses (in (d) and (e), the two segments are assumed to be collinear though they are drawn in parallel). Hence selective perturbation is achieved.

We implemented our method and compared the time complexity for various input sets of line segments with existing methods. The results are summarized in Fig. 4. The "naive"method means the exhaustive check between all pairs of segments, and the "exception handling"method means the ordinary plane sweep method with additional exception handling. Our method was much slower than the plane sweep method; this is firstly because our implementation was based on symbolic processing of mathematical expressions instead of numerical computation, and secondly because our method enlarges the input data size four times bigger. Though the CPU time does not show the advantage of our method, the time for implementation of the software is comparable because case-by-case exceptional processing is not necessary.

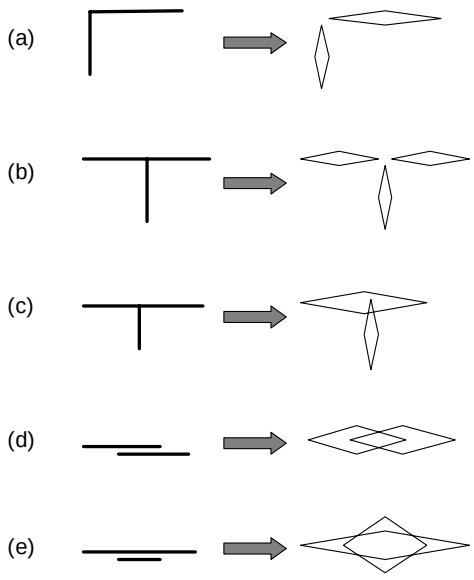


Figure 3: Effects of the replacement of the segments: (a) and (b) false detection is avoided; (c), (d) and (e) true intersections are detected.

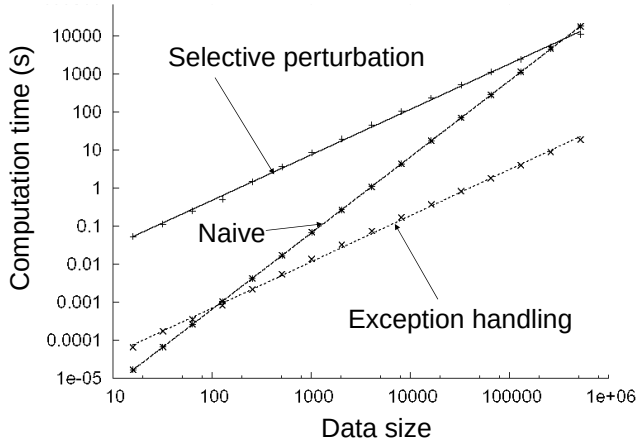


Figure 4: Comparison of CPU time.

5 Concluding Remarks

We pointed out that symbolic perturbation sometimes generates unwanted side effects, and proposed new methods for avoiding those side effects in the Delaunay tetrahedrization problem and the circuit layout check problem. Our basic idea is to perturb the problem itself instead of perturbing the input data. In this sense we may call this strategy "hyper-perturbation."

In the hyper-perturbation strategy, we first modify the problem itself, then apply symbolic perturbation to the modified problem. We have succeeded in avoiding unwanted side effects of symbolic perturbation in at least two problems. Studying the validity of this strategy for other side effects of symbolic perturbation is our next challenge.

Acknowledgments

This work is supported by the Grant-in-Aid for Scientific Research (B) No. 20360044 and Grant-in-Aid for Exploratory Research No. 19650003 of the Japanese Society for the Promotion of Science.

References

[1] J. L. Bentley and T. A. Ottmann. *Algorithms for reporting and counting geometric intersections.* IEEE Transactions on Computers, vol. C-28 (1979), pp. 643–647.

[2] H. Edelsbrunner and E. P. Mücke. *Simulation of simplicity — A technique to cope with degenerate cases in geometric algorithms.* ACM Transactions on Graphics, vol. 9 (1990), pp. 67–104.

[3] L. Guibas and J. Stolfi. *Epsilon geometry — Building robust algorithms from imprecise calculations.* Proceedings of the 5th ACM Annual Symposium on Computational Geometry, 1989, Saarbrücken, pp. 208–217.

[4] M. Held and S. Huber. *Topology-oriented incremental computation of Voronoi diagrams of circular arcs and straight-line segments.* Computer-Aided Design, vol. 41 (2009), pp. 327–338.

[5] K. Mehlhorn and S. Näher. *LEDA — A Platform for Combinatorial and Geometric Computing.* Cambridge University Press, 1999, Cambridge.

[6] K. Sugihara. *How to make geometric algorithms robust.* IEICE Transactions on Information and Systems, vol. E83-D (2000), pp. 447–454.

[7] K. Sugihara. *Sliver-free perturbation for the Delaunay tetrahedrization.* Computer-Aided Design, vol. 39 (2007), pp. 87–94.

[8] K. Sugihara and M. Iri. *A solid modeling system free from topological inconsistency.* Journal of Information Processing, vol. 12 (1989), pp. 380–393.

[9] K. Sugihara and M. Iri. *A robust topology-oriented incremental algorithm for Voronoi diagrams.* International Journal of Computational Geometry and Applications, vol. 4 (1994), pp. 179–228.

[10] S. Takahashi. *Symbolic Perturbation for the Circuit Layout Check Problem.* Thesis for Bachelor's Degree, Department of Mathematical Informatics, the University of Tokyo, 2008.

[11] C.-K. Yap. *Symbolic treatment of geometric degeneracies.* Journal of Symbolic Computation, vol. 10 (1990), pp. 349–370.

[12] C.-K. Yap. *Toward exact computation.* Proceedings of the 5th Canadian Conference on Computational Geometry, 1993, pp. 405–419.

Geometric realization of a triangulation on the Klein bottle with one face removed

Atsuhiro Nakamoto* Shoichi Tsuchiya†

Abstract

Let M be a map on a surface F^2. A *geometric realization* of M is an embedding of F^2 into a Euclidian 3-space $\mathbb{R}^3$ with no self-intersection such that each face of M is a flat polygon. In earlier researches, it has been proved that every triangulation on the sphere and on the torus has a geometric realization. Moreover, it has been proved that every triangulation G on the projective plane has a face f such that the triangulation $G - f$ on the Möbius band obtained from G by removing the interior of f has a geometric realization. In this paper, we shall prove that every 5-connected triangulation G on the Klein bottle has a face f such that $G - f$ has a geometric realization.

1 Introduction

A *map* on the surface F^2 is a fixed embedding of a graph on F^2. A *triangulation* on a surface F^2 is a map on F^2 such that each face is bounded by a 3-cycle, where a k-cycle means a cycle of length k. A *link* of a vertex v of a graph G is the boundary walk of the union of the faces incident to v. We suppose that the graph of a map is always *simple*, i.e., with no multiple edges and no loops. Let M be a map on a surface F^2. A *geometric realization* of M is an embedding of F^2 into a Euclidian 3-space $\mathbb{R}^3$ with no self-intersection such that each face of M is a flat polygon.

Steinitz has proved that a spherical map G has a geometric realization if its graph is 3-connected [10]. (He actually proved that G has a geometric realization such that no two adjacent faces lie on the same plane in $\mathbb{R}^3$ if and only if G is 3-connected.) His theorem claims that every spherical triangulation has a geometric realization since a triangulation is 3-connected. Moreover, Archdeacon et al. have proved that every toroidal triangulation has a geometric realization [1]. In general, Grünbaum conjectured that every triangulation on any orientable closed surface has a geometric realization [6], but Bokowski et al. showed that a triangulation by the complete graph K_{12} with twelve vertices on the orientable closed surface of genus 6 has no geometric realization [3]. Hence Grünbaum's conjecture is no longer true now but it is still open for the orientable closed surface of genus from 2 to 5.

Let us consider nonorientable surfaces. It is known that any nonorientable closed surface is not embeddable in $\mathbb{R}^3$. So, no map on nonorientable closed surfaces has a geometric realization. However, a nonorientable surface with one open disk removed is known to be embeddable in $\mathbb{R}^3$. Hence a map G on a nonorientable closed surface might have a face f such that $G - f$ has a geometric realization.

Let G be a triangulation on the projective plane and let f be a face of G. For simple notations, we call a triangulation on the projective plane and that on the Möbius band a *projective triangulation* and a *Möbius triangulation*, respectively, throughout the paper. For projective triangulations, Bonnington and Nakamoto have proved the following [2].

Theorem 1 (Bonnington and Nakamoto [2]) *Every projective triangulation G has a face f such that $G - f$ has a geometric realization.*

Although Theorem 1 asserts that every projective triangulation G has a face f such that $G - f$ has a geometric realization, a projective triangulation might have a face f such that $G - f$ has no geometric realization. We obtain such a triangulation from Brehm's example. Brehm [4] found a Möbius triangulation with no geometric realization shown in Figure 1. (In Figure 1, identify vertices with the same label.)

Brehm's counterexample implicitly assures that a projective triangulation G has no geometric realization if G has a face f and a 3-cycle C such that the boundary cycle of f and C are disjoint and bound an annular region. In this case, we say that C is a *nesting* 3-cycle of f. (In Figure 1, if 123 is the boundary of f, then 456 is a nesting 3-cycle of f.) Recently, we have characterized a face f of a projective triangulation G such that $G - f$ has a geometric realization, as follows.

Theorem 2 (Nakamoto and Tsuchiya [8]) *Let G be a projective triangulation and let f be a face of G. Then $G - f$ has a geometric realization if and only if there is no nesting 3-cycle of f in G.*

*Department of Mathematics, Yokohama National University, 79-2 Tokiwadai, Hodogaya-ku, Yokohama 240-8501, Japan Email: nakamoto@edhs.ynu.ac.jp

†Department of Information Media and Environment Sciences, Graduate School of Environment and Information Sciences, Yokohama National University, 79-7 Tokiwadai, Hodogaya-ku, Yokohama 240-8501, Japan

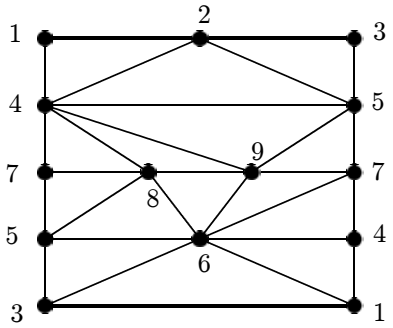


Figure 1: A Möbius triangulation with no geometric realization constructed by Brehm

Theorem 2 asserts that only the structure of Brehm's example with a boundary 3-cycle and its nesting 3-cycle breaks the geometric realizability of Möbius triangulations.

In the current work, we consider the case of the Klein bottle. For a simple notation, we call a triangulation on the Klein bottle a *Klein triangulation*. Although every projective triangulation G has a face f such that $G-f$ has a geometric realization by Theorem 1, there exists a Klein triangulation with no such face. We can construct such a triangulation from two Brehm's counterexamples by pasting along their boundaries since the Klein bottle is obtained from two Möbius bands by pasting along their boundaries. (See Figure 2.) The Klein triangulation G shown in Figure 2 has three disjoint 3-cycles each of which divides the Klein bottle into two Möbius bands. (The 3-cycles 123, 456, $4'5'6'$.) Hence G has no face f such that $G-f$ has a geometric realization, since $G-f$ must have a Brehm's counterexample as a submap.

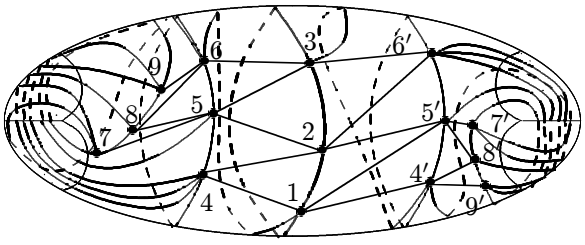


Figure 2: A Klein triangulation G with no face f such that $G-f$ has a geometric realization.

Proposition 3 *There exists a Klein triangulation G with no face f such that $G-f$ has a geometric realization.*

Which Klein triangulations G have a face f such that $G-f$ has a geometric realization? In this paper, we shall prove the following.

Theorem 4 *Every 5-connected Klein triangulation G has a face f such that $G-f$ has a geometric realization.*

Observe that if G is 5-connected, G contains no 3-cycle which divides the Klein bottle into two Möbius triangulation.

2 Decomposition of Klein triangulation into two Möbius triangulations

To prove Theorem 4, we shall construct a geometric realization of $G-f$. To do so that, we devide G into two Möbius triangulations with good property for making a geometric realization of $G-f$.

First, we shall prove that G has a cycle C separating G into two Möbius triangulations M_1 and M_2. Choosing C carefully, we can take C in G so that the following is satisfied.

Lemma 5 *A 5-connected Klein triangulation G contains a cycle C dividing G into two Möbius triangulations M_1, M_2 such that:*

(i) M_2 contains a cycle D homotopic to the center line of M_2 and D is disjoint from C and

(ii) any vertex of C is neighboring to vertices on D.

Let ∂M_1 (resp., ∂M_2) denote the boundary cycle of M_1 (resp., M_2). Note that $C=\partial M_1=\partial M_2$.

Let G be a map on a surface and let xy be an edge of G. *Contraction* of xy is to remove xy and identify x and y. (If the resulting map has a face bounded by a 2-cycle, then we replace the multiple edge with a single edge.) Its inverse operation is called a *splitting* of a vertex.

A *K_5-triangulation* is a triangulation on a Möbius band whose graph is isomorphic to a complete graph K_5 with five vertices. A K_5-triangulation plays an essential role for a geometric realization of a Möbius triangulation [2, 5].

Let H_0 be a K_5-triangulation with vertices v_i, for $i=1,2,3,4,5$, where $\partial H_0=v_1v_2v_3v_4v_5$ is the boundary. Let H be a map obtained from H_0 by a sequence of a splitting of each vertex v_i into two vertices v_i and v_i' of degree 3 or no splitting of vertex of H_0. Then we call H a *split-K_5*. We call a vertex of H whose degree greater than 2 a *node* and a path in H containing only two nodes as its two endpoints a *segment*. There are two ways of a splitting v_i whether v_i' lies on the boundary or not. A node v of H is called a *boundary node* if it lies on the boundary of H. Otherwise it is called an *inner node*. (In Figure 3, v_3' is a boundary node and v_2' is an inner node.) By the lemmas proved in [2, 5], we can prove the following.

Lemma 6 *The Möbius triangulation M_1 contains a split-K_5 with five or six boundary nodes as a subdivision.*

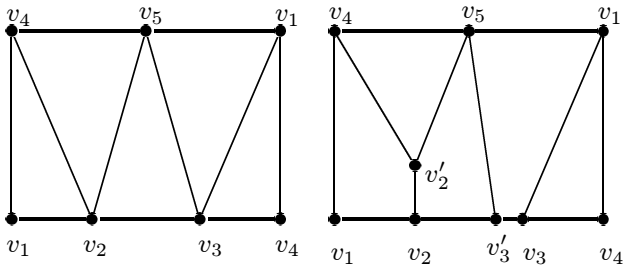


Figure 3: A K_5-triangulation (left) and a split-K_5 (right).

Now we consider a structure of M_2. We prove that M_2 contains an *inner vertex* v (i.e., $v \notin V(\partial M_2)$) which satisfies the following lemma.

Lemma 7 *The Möbius triangulation M_2 contains an inner vertex v whose link contains at least three vertices of ∂M_2.*

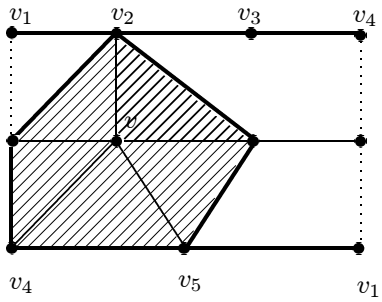


Figure 4: A link of v in M_2. (The shaded region shows the interior of L.)

3 Construction of geometric realization

In this section, we would like to show that a 5-connected Klein triangulation G has a face f such that $G - f$ has a geometric realization. Let us put an example of a geometric realization of $G - f$. By Lemmas 5, 6 and 7, G has a cycle C dividing G into two Möbius triangulations M_1 and M_2 such that:

(i) M_1 contains a split-K_5 with five or six boundary nodes as a subdivision and

(ii) M_2 contains an inner vertex v such that the link of v contains at least three vertices of ∂M_2.

Let H be a Möbius triangulation which contains a split-K_5 with five or six boundary nodes as a subdivision. It has been proved that H has a geometric realization each segment of ∂H is a straight segment [2, 5]. Let $\hat{H}$ denote such a geometric realization of H. Figure 5 shows an example of $\hat{H}$ with five boundary nodes and no inner node. The boundary of $\hat{H}$ is said to *satisfy an s-condition* if there exists a view point s such that the segment of $\partial\hat{H}$ can be seen from s except for only one segment. Observe that $\partial\hat{H} = v_1v_2v_3v_4v_5$ shown in Figure 5 satisfies an s-condition since we can see all the segments of $\partial\hat{H}$ except for v_1v_2. In this case, we can easily see that we can put four triangular disks $v_2v_3s, v_3v_4s, v_4v_5s$ and v_5v_1s to the body of $\hat{H}$ without collisions.

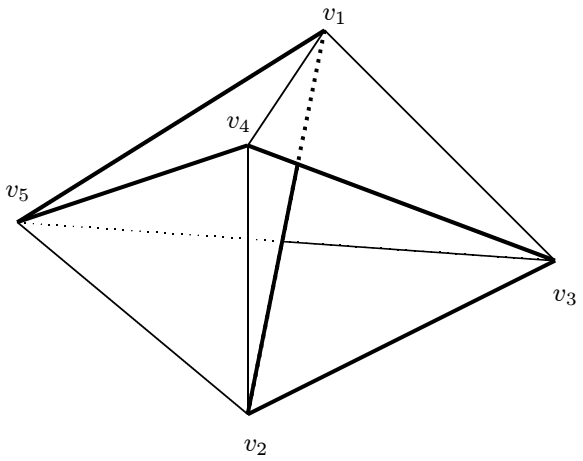


Figure 5: A geometric realization of H with five boundary nodes and no inner node.

Since M_1 contains a split-K_5 with five or six boundary nodes as a subdivision, we obtain a geometric realization of M_1, denoted by $\hat{M}_1$, such that $\partial\hat{M}_1$ satisfies an s-condition. Let L be the link of an inner vertex v in M_2 such that $|V(L) \cap V(\partial M_2)| \geq 3$. We would like to put faces of M_2 in the exterior of L into the body of $\hat{M}_1$ so that the boundary of the resulting geometric realization satisfies an s-condition. So we proved the following.

Lemma 8 *Let L be the link of an inner vertex v in M_2 such that $|V(L) \cap V(\partial M_2)| \geq 3$. The triangulation obtained from G by removing the interior of L has a geometric realization $\hat{G}_L$ such that $\partial\hat{G}_L$ satisfies an s-condition.*

It is not difficult to see that all faces in the interior of L except one can be added to $\hat{G}_L$ without collisions since $\partial\hat{G}_L$ satisfies an s-condition. Therefore G has a face f such that $G - f$ has a geometric realization and hence we can prove Theorem 4.

4 Conjecture

In this paper, we have considered 5-connected Klein triangulations. However, if we want to avoid the three

disjoint 3-cycles each of which separates a Klein triangulation into two Möbius triangulations, then the 4-connectivity of G suffices. So we do not know whether the 5-connectivity of Klein triangulations in Theorem 4, and hence we have the following.

Conjecture 1 *Every 4-connected Klein triangulation G has a face f such that $G - f$ has a geometric realization.*

5 Conclusion

To deal with geometric realizations of the nonorientable maps, we consider the geometric realizability of a triangulation on a nonorientable closed surface with one face removed.

For a projective triangulation G, Theorem 1 claims that G has a face f allowing a geometric realization of $G - f$, and Theorem 2 characterizes such a face f.

For a Klein triangulation G, we have been able to prove that G has a face f allowing a geometric realization of $G - f$, if G is 5-connected. However, every G does not necessarily have such a face. We want to know a geometric realizability of a Klein triangulation with one face removed, but the problem does not seem to be easy, since the topology of the Klein bottle is not so simple as that of the projective plane.

References

[1] D. Archdeacon, C.P. Bonnington and J.A. Ellis-Monanghan, How to exhibit toroidal maps in space, *Discrete Comp. Geom.* **38** (2007), 573–594.

[2] P. Bonnington and A. Nakamoto, Geometric realization of a projective triangulation with one face removed, *Discrete Comp. Geom.* **40** (2008), 141–157.

[3] J. Bokowski and A. Guedes de Oliveira, On the generation of oriented matroids, *Discrete Comput. Geom.* **24** (2004), 197–208.

[4] U. Brehm, A nonpolyhedral triangulated Möbius strip, *Proc. Amer. Math. Soc.* **89** (1983), 519–522.

[5] M. Chávez, G. Fijavž, A. Márquez, A. Nakamoto and Suárez, Geometric realization of triangulations on the Möbius band, *SIAM J. Discrete Math.* **23** (2008), 221–232.

[6] B. Grünbaum, "Convex polytopes", *Pure and Applied Mathematics* Vol. 16, Interscience-Wiley, New York, 1967.

[7] K. Menger, Zur allgemeinen Kurventheorie, *Fund. Math.* **10** (1927), 95–115

[8] A. Nakamoto, S. Tsuchiya, Geometrically realizable triangulations on the Möbius band, *preprint*

[9] D. Rolfsen, *Knot and links*, Math. Lecture Series 7, Publish or Perish, Berkeley, Calif., 1976.

[10] E. Steinitz, Polyeder und Raumeinteilungen, *Enzykl. Math. Wiss.* Vol. 3, Teil 3A612 (1922), 1–139.

[11] T. Sulanke, Note on the irreducible triangulation of the Klein bottle, *Journal of Combinatorial Theory.* **B 96** (2006), 964–972.

[12] W. Tutte, How to draw a graph, *Proc. London Math. Soc.* **13**(1963),743–767

Real-Time Offset Surfaces

A. von Dziegielewski * R. Erbes† E. Schömer ‡

Abstract

We present a novel technique for the direct rendering of offset surfaces for polygonal meshes. The visible part of the offset surface of each triangle, defined as the union of three spheres, three cylinders and a prism, is constructed in a shader program utilizing the geometry shader. Our method creates exact offset surfaces, up to pixel resolution. Possible applications are real-time visualization of offset surfaces, e.g. for GPU-based collision detection or conservative voxelization and rasterization of complex triangle meshes.

1 Introduction

When passing geometry to a graphics card, whether for rendering-related reasons or for general purpose processing, one always has to deal with the problem that hardware rasterization in general is not conservative [11], i.e. the fragments produced do not form a superset of the geometry projected onto the viewing plane.

A common approach to achieve conservativeness is not to render the geometry itself but an adequate offset surface. For a given geometry $\mathcal{A} \subset \mathcal{R}^3$ and $\delta > 0$ we define the **offset** $\mathcal{O}_\delta(\mathcal{A})$ of $\mathcal{A}$ to be the Minkowski sum of $\mathcal{A}$ and the solid ball $\mathcal{B}_\delta(0)$ centered at the origin,

$$\mathcal{O}_\delta(\mathcal{A}) = \mathcal{A} \oplus \mathcal{B}_\delta(0) = \left\{ x \,\middle|\, \exists a \in \mathcal{A} \ : \|x - a\| \leq \delta \right\}$$

and the **offset surface** of $\mathcal{A}$ to be its boundary: $\mathcal{S}_\delta(\mathcal{A}) = \partial\mathcal{O}_\delta(\mathcal{A})$. We call δ the **offset radius**.

Given a $N \times N$-viewport, the rasterization of the offset surface can be shown to be a conservative rasterization of the initial geometry if $\delta > \frac{\sqrt{3}}{2N}$ [6].

We present a novel technique for generating offset surfaces for polygonal meshes as a rendering process.

Besides conservative rasterization, our method can be directly used for GPU-based collision detection as described in [2]. The authors point out accuracy problems with overlap tests, inherent to image-based intersection techniques. Applying the technique presented here will overcome these issues, making the collision detection conservative (reliable) and easily extendable to distance testing with an arbitrary safety clearance.

Another possible application is hardware voxelization [7, 1]. Applying an adequate offset to a geometry with our method yields a conservative voxelization of the initial geometry with a given error bound.

2 Previous work

Attempts to overcome the problem of nonconservative rasterization have been made [4]. Similar to our work a combination of a geometry and a fragment shader are used to achieve conservativeness, but their method only applies rather coarse depth values, and their computation highly overestimates depth values for triangles with normals perpendicular to the viewing plane. Further their method cannot render arbitrary offsets. We overcome this inaccurate depth value estimation and therefore can give proper bounds on the deviation from each produced fragment to the original geometry.

Previous work on offset surfaces [9, 3, 8] mainly deals with the computation and extraction of offset geometry, therefore these existing methods are not applicable for real-time offset rendering. In one of the few publications on offset visualization [5], the authors propose to generate an adaptive distance field around the mesh and to extract the offset boundary as an isosurface at interactive rates using splatting. In contrast to our approach, a time consuming process of creating a distance field is needed.

Our approach follows the general idea of [9, 3]. They use the fact that an offset surface of a triangle equals the union of the surfaces of three spheres, three cylinders and a prism, trimmed at their intersection points. Our method circumvents this challenging process of trimming. As we are only interested in rendering the offset surface, it is implicitly done by the depth buffer.

A method for extracting offset geometry is presented in the recent work of [8]. The authors compute an adaptive distance field around the input geometry, storing the minimum and an approximated maximum distance to each triangle using special distance-functions (ball, cylinder prism). They generate the offset in an isosurface extraction process followed by feature reconstruction and mesh simplification.

*Department of Computer Science, Johannes Gutenberg-Universität Mainz, dziegiel@uni-mainz.de

†Department of Computer Science, Johannes Gutenberg-Universität Mainz, erbes@uni-mainz.de

‡Department of Computer Science, Johannes Gutenberg-Universität Mainz, schoemer@uni-mainz.de

3 Rendering Offsets on the Fly

Given a, b, $c \in \mathcal{R}^3$ we denote $T_{(a,b,c)}$ to be the **triangle** with its **vertices** a, b, c and **edges** e_{ab}, e_{bc}, e_{ac}. For an edge e_{ab} and $\delta > 0$ we define the **solid cylinder** $C_\delta(e_{ab})$ with axis e_{ab} and radius δ to be the set

$$C_\delta(e_{ab}) = \Big\{ x \in \mathcal{R}^3 \;\Big|\; \big\|(x-a) \times \tfrac{b-a}{\|b-a\|}\big\|^2 \le \delta^2) \\ \wedge \quad 0 \le \tfrac{(x-a)\cdot(b-a)}{\|b-a\|^2} \le 1 \Big\}.$$

For a given triangle $T_{(a,b,c)}$ with **normal** $n = \frac{(b-a)\times(c-a)}{\|(b-a)\times(c-a)\|}$ we define a (δ-) **offset-prism** of $T_{(a,b,c)}$ to be the convex hull of the two shifted triangles $T_{(a+\delta n, b+\delta n, c+\delta n)}$ and $T_{(a-\delta n, b-\delta n, c-\delta n)}$:

$$P_\delta(T_{(a,b,c)}) = \mathrm{CH}\big\{ \quad a+\delta n,\, b+\delta n,\, c+\delta n, \\ a-\delta n,\, b-\delta n,\, c-\delta n \big\}.$$

The offset to a triangle $T_{(a,b,c)}$ equals the union of three spheres, three cylinders and a prism:

$$\mathcal{O}_\delta(T_{(a,b,c)}) = \quad P_\delta(T_{(a,b,c)}) \cup C_\delta(e_{ab}) \cup C_\delta(e_{bc}) \\ \cup C_\delta(e_{ac}) \cup \mathcal{B}_\delta(a) \cup \mathcal{B}_\delta(b) \cup \mathcal{B}_\delta(c).$$

The straightforward method, to compute and tessellate this geometry for every triangle, is not practical in the case of a high triangle count. In our applications we use meshes consisting of up to one million triangles. Let us assume the tessellation process yields roughly 100 triangles per sphere or cylinder, then the resulting storage would be around 10 gigabytes. This by far exceeds the memory of even modern graphics cards, and hence the resulting triangles could not be stored in a display list [12]. The usage of display lists is absolutely essential when we think of visualizing the offsets of a mesh at interactive rates.

We therefore present a new approach to efficiently compute and render the exact offset geometry up to pixel resolution. The original mesh can easily be compiled in a display list and resides in graphics memory. A combination of a geometry shader [11] and fragment shader [10] computes the depth-values of fragments of the offset boundary on the fly, hence no extra storage is needed.

In our new method, every triangle is processed by the following rendering pipeline: First the vertex shader passes on the vertices of the triangle to the geometry shader stage without changing anything. The geometry shader creates a patch for each edge of the triangle in the xy-plane and also emits the two triangles of the offset prism in 3d. The latter get rasterized with the correct depth value and the fragments can be written to the depth buffer directly. The rectangular patches of the edges however still have a depth value according to $z = 0$, hence we calculate the correct depth value for each patch-fragment with a ray casting method, as shown in Figure 1.

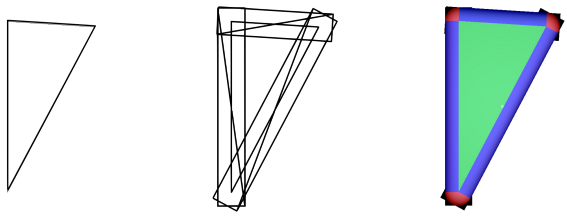


Figure 1: For each edge of the triangle projected to the xy-plane, a patch is constructed in the geometry shader stage and the appropriate depth values are computed in the fragment shader (blue, red). Together with the shifted triangle (green), the final offset surface is rendered, and dispensable fragments (black) are clipped.

We now give a detailed description of the shaders. We used the OpenGL Shading Language (GLSL) [10].

3.1 Vertex stage

In the rendering pipeline, the vertex shader is responsible to apply any kind of transformations on the vertex data. In our case the vertex shader simply passes on the position values for each vertex to the geometry shader.

3.2 Geometry stage

The geometry shader [11] is able to create new primitives (in our case triangles) that are passed on to the rasterization stage just as primitives directly rendered by OpenGL. In contrast to the previous stage, where each vertex was treated independently, now information on vertex-connectivity is provided, i.e. we can work on the input triangles. In the general case we emit eight triangles per input triangle: two for each edge-patch and two triangles for the offset prism.
For each edge e_{ab} of the triangle we compute a rectangular patch that is a bounding box of the projection of the δ-balls $\mathcal{B}_\delta(a)$, $\mathcal{B}_\delta(b)$ and the δ-cylinder $C_\delta(e_{ab})$ on the viewing plane. The geometry shader computes and emits the triangles of the tessellated patch

$$\big\{s,\, t,\, v,\, u\big\} = T_{(s,t,u)} \cup T_{(t,v,u)},$$

as shown in Figure 2. The dimensions of the patch are given as

$$d_1 = \frac{\delta h}{\|h\|},\; d_2 = \begin{pmatrix} -(d_1)_y \\ (d_1)_x \\ 0 \end{pmatrix},\; h = \begin{pmatrix} (p_z(b-a))_x \\ (p_z(b-a))_y \\ 0 \end{pmatrix},$$

where $p_z(\cdot)$ is the orthogonal projection onto the viewing plane.

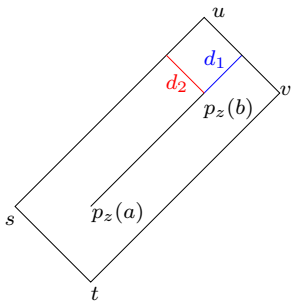


Figure 2: the patch for the edge e_{ab}

The offset triangles of the δ-offset prism can be computed by shifting the vertices a, b, c by δ once in normal direction and once in the opposite direction. The rectangular faces of the prism are not needed, since they are completely occluded by the cylinders.
All emitted triangles are now rasterized and cut into loose fragments. To be able to relate the fragments to their edges or offset triangles, we pass on two 4-vectors (to the fragment shader) holding all information needed. The coordinates of the vertices of the edge are passed as color values and the information, whether the fragment belongs to an offset triangle or a patch, is encoded in the alpha channels.

3.3 Fragment stage

The fragment shader now has to cope with the following three tasks: It must decide whether the fragment f belongs to an offset triangle or an edge-patch, it must discard the fragments that are outside the offset and finally adjust the correct depth values.

The first one can be done easily by checking the alpha value of the variables received from the geometry shader. In the case of an offset triangle, the z coordinate of the built-in fragment coordinates can be mapped directly to the output depth value just like in the fixed function pipeline.

For the computation of the correct depth value for a patch-fragment we define the line $\tilde{e}_{ab}$ as the edge e_{ab} extended to infinity and the endless cylinder $C_\delta(\tilde{e}_{ab}) = \mathcal{S}_\delta(\tilde{e}_{ab})$, given in the implicit form:

$$C_\delta(\tilde{e}_{ab}) = \left\{ p \in \mathcal{R} \;\middle|\; \left\|(p-a)\times(b-a)\right\|^2 \le \delta^2 \|b-a\|^2 \right\}.$$

The intersection of the ray

$$r(\lambda) = \begin{pmatrix} f_x \\ f_y \\ 0 \end{pmatrix} + \lambda \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix}, \quad \lambda > 0$$

with $\partial C_\delta(\tilde{e}_{ab})$ yields the quadratic equation

$$\begin{aligned} & \left\|(f-a-\lambda e_z)\times(b-a)\right\|^2 = \delta^2\|b-a\|^2 \\ \Leftrightarrow \quad & \Big\| \underbrace{(f-a)\times\frac{(b-a)}{\|b-a\|}}_{=:c} - \lambda\Big(\underbrace{e_z\times\frac{(b-a)}{\|b-a\|}}_{=:d}\Big)\Big\|^2 = \delta^2 \end{aligned}$$

that we can solve for λ :

$$\lambda_{1,2} = \tfrac{c\cdot d}{d\cdot d} \pm \sqrt{\left(\tfrac{c\cdot d}{d\cdot d}\right)^2 - \tfrac{c\cdot c-\delta^2}{d\cdot d}}.$$

Of the two possible values we choose the greater one (λ_1) and set $f_z = -\lambda_1$, since we are interested in the intersection closer to the viewing plane.

So far we have only considered an endless cylinder with axis e_{ab}. To get the correct offset $\mathcal{S}_\delta(e_{ab})$, we need to clip the cylinder at a and b, i.e. all fragments not fulfilling

$$0 \le (f-a)\cdot(b-a) \le (b-a)\cdot(b-a)$$

have to be either projected on

- the ball $\mathcal{B}_\delta(a)$ for $(f-a)\cdot(b-a) < 0$ or
- the ball $\mathcal{B}_\delta(b)$ for $(f-a)\cdot(b-a) > (b-a)\cdot(b-a)$.

Both tasks are easy as we just have to intersect $r(\lambda)$ with the respective ball to get the correct depth value. In the case of no intersection, we discard the fragment. This only happens in the corners of the patch, where it is outside the projected balls around a and b, as shown in Figure 1.
For example, in the case of $\mathcal{B}_\delta(a)$, the intersection yields

$$f_z = -\lambda = a_z - \sqrt{\delta^2 - (f_x-a_x)^2 - (f_y-a_y)^2}.$$

4 Results

For benchmarking we used a GeForce GTX 280 with 1 GB of memory and a 3.0 GHz CPU. The models were rendered to a $N \times N = 800 \times 800$ viewport under orthographic projection and scaled by $0.75/\Delta$, with Δ being half the diameter of the bounding box of the model. The offset radius δ was chosen to be a multiple of the length of a voxel diagonal: $\delta = n\frac{\sqrt{3}}{N}$. The resulting FPS for different choices of n can be found in Figure 3.

5 Conclusions

With our new approach we are able to render offset surfaces for highly complex meshes at interactive rates on a GeForce GTX 280 graphics card.

A comparable non-GPU-approach, to calculate and tessellate offset spheres, cylinders and prisms at a preprocessing stage for the later rendering, is not only

model		FPS					
name	# triangles	$n = 0.5$ (conservative rasterization)	$n = 1$	$n = 5$	$n = 10$	$n = 20$	$n = 40$
X-Wing	6,073	1363.0	993.4	271.2	141.0	76.9	29.4
Buddha	26,999	505.1	463.0	113.6	43.5	21.7	7.6
Gargoyle	40,348	263.0	169.0	34.5	16.4	11.8	4.5
Steering gear	261,807	51.3	28.9	5.3	2.4	1.9	6.9
Dragon	871,414	18.2	14.7	3.5	1.4	0.9	0.3

Figure 3: Frame rates for different input models with different offset radii.

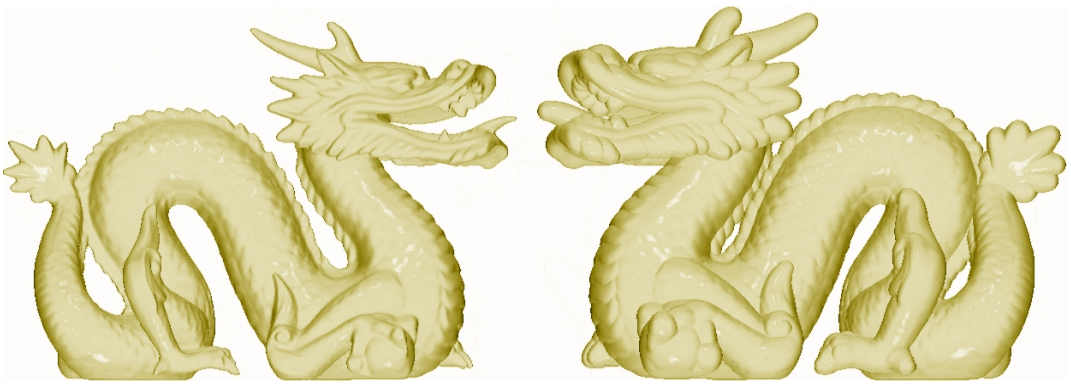


Figure 4: Original dragon model and offset surface ($n = 10$).

significantly slower and less accurate (due to tessellation errors) but would produce an exorbitant number of triangles and therefore exceed the rendering capacity even of modern graphics hardware.

Our method is lightweight and, due to the utilization of the geometry shader, no preprocessing is needed. Since the algorithm works on every triangle independently, and does not use any information on the mesh connectivity, we do not need the input meshes to meet any topological requirements and can even process scattered triangles or triangle soups.

References

[1] E. Eisemann and X. Décoret. Single-pass gpu solid voxelization for real-time applications. In *GI '08: Proceedings of graphics interface 2008*, 2008.

[2] C. Ericson. *Real-Time Collision Detection (The Morgan Kaufmann Series in Interactive 3-D Technology)*. Morgan Kaufmann, January 2005.

[3] M. Forsyth. Shelling and offsetting bodies. In *SMA '95: Proceedings of the third ACM symposium on Solid modeling and applications*, pages 373–381, New York, NY, USA, 1995. ACM.

[4] J. Hasselgren, T. Akenine-Mller, and L. Ohlsson. *GPU Gems 2*, chapter Conservative Rasterization, pages 677–690. Addison-Wesley Professional, 2005.

[5] J. Huang, Y. Li, R. Crawfis, S. C. Lu, and S. Y. Liou. A complete distance field representation. In *VIS '01: Proceedings of the conference on Visualization '01*, pages 247–254, Washington, DC, USA, 2001. IEEE Computer Society.

[6] J. Huang, R. Yagel, V. Filippov, and Y. Kurzion. An accurate method for voxelizing polygon meshes. In *VVS '98: Proceedings of the 1998 IEEE symposium on Volume visualization*, pages 119–126, New York, NY, USA, 1998. ACM.

[7] E.-A. Karabassi, G. Papaioannou, and T. Theoharis. A fast depth-buffer-based voxelization algorithm. *J. Graph. Tools*, 4(4):5–10, 1999.

[8] D. Pavic and L. Kobbelt. High-resolution volumetric computation of offset surfaces with feature preservation. *Comput. Graph. Forum*, 27(2):165–174, 2008.

[9] J. R. Rossignac and A. A. G. Requicha. Offsetting operations in solid modelling. *Comput. Aided Geom. Des.*, 3(2):129–148, 1986.

[10] R. J. Rost. *OpenGL Shading Language*, chapter 4.2: The Fragment Processor, page 104 ff. Addison-Wesley Longman, 2 edition, 2006.

[11] M. Segal and K. Akeley. *The OpenGL Graphics System: A Specification (Version 3.2)*. 2009.

[12] D. Shreiner, M. Woo, J. Neider, and T. Davis. *OpenGL Programming Guide*, page 487 ff. Addison-Wesley Longman, 6 edition, 2008.

A fast and easy-to-implement algorithm for the Minimal Translational Distance (MTD) of boxes

Kai Werth* Elmar Schömer†

Abstract

Several methods for penetration depth and Euclidean distance calculation exist for convex bodies. They mostly are uneasy to describe and complex in implementation. Due to their general specification they also are not practical in the case of primitive bodies like boxes or triangles. In this work we present an easy-to-implement algorithm to compute the MTD of two convex polyhedra that is runtime optimized for primitive shapes. We describe a detailed implementation for boxes that runs faster than existing approaches and give the theory to use the scheme for general convex polyhedra.

1 Introduction

The minimal translational distance (MTD) of two objects is the shortest translation vector "that results in the objects being in contact" [1]. In the case of intersecting objects, we call its length the penetration depth and Euclidean distance otherwise.

Little is published on penetration depth methods. But there exist many implementations for penetration depth of convex objects as it is of relevance for applications like physics simulation where illegal situations of penetrating objects have to be legalized (c.f. [12] or [4]). In broad-and-narrow-phase based algorithms it is used to detect bounding volume collisions like [7] or [5].

Only few implementations exist to compute the distance of convex objects like [3] and [13], and they are often hard to implement. The majority of implementations is based one of these three concepts: GJK-algorithm from [2], external Voronoi regions ([11] and [10]) or Minkowski sums ([1]). One remarkable box/box distance test exists in [3]. It reduces the problem two $6*6$ rectangle tests of which each one needs 81 case distinctions, which makes it very uneasy to implement. Further it only provides the squared distance and no real MDT for intersecting boxes.

The algorithm described in this work provides a solution for both, penetration and distance. It concentrates on box pairs, but works in the same way for triangles, rectangles or pyramids. Further, the pair partners do not have to be of the same type which is applicable for common operations such as box/triangle tests. In general the method works for any two polytopes, but the complexity is quadratic in numbers of edges. The method is based on the *separating axis theorem (SAT)* for convex polyhedra which was introduced in [6].

Unlike other approaches for primitive shapes like [9] our approach does not provide an early-out strategy. It always tests all eligible separating axes to find the maximal separation. On the other hand this makes our approach suitable for parallel implementations for GPGPU or other SIMD architectures in which every thread should have the same number of operations.

2 About this document

The document is structured as follows: section (3) gives an introduction on SAT. In section (4) we describe the penetration depth computation for two intersecting boxes. This will be followed in section (5) by the pairwise distance algorithm on boxes. The MTD algorithm then is a conclusion from section (4) and section (5) and will be presented in section (6). Section (7) will give a constructive proof and a conclusion in section (9) .

3 SAT

Theorem 1 (Separating Axis Theorem (SAT)) *Given two convex objects A and B*

$$A \cap B = \emptyset \quad \Longleftrightarrow \quad \exists \text{ a plane that separates } A \text{ from } B$$

A proof for SAT can be found in ([7]). There we also learn, that for two convex polyhedra P_1 and P_2 it suffices to test a finite number of planes, namely those that are either

- parallel to one face f with $f \in \mathsf{P}_1$ or $f \in \mathsf{P}_2$ or
- parallel to two edges e_1, e_2 with $e_1 \in \mathsf{P}_1$ and $e_2 \in \mathsf{P}_2$.

If no such separating plane can be found, P_1 and P_2 have a non-empty intersection. The normal of a separating plane, if it exists, is called the *separating axis*. The *separating axis test* itself is done as follows: Project P_1 and P_2 on each eligible separating axis $\mathbf{n}$. If any two projections correspond to two disjoint line segments then $\mathbf{n}$ is a separating axis and P_1 and P_2 are disjoint. The separating axis theorem tells us that we only have to test a finite number of eligible axes namely the normals of all faces of P_1 and P_2 and the pairwise cross products of the edge directions (c.f. fig. (1)).

*Department of Computer Science, Johannes-Gutenberg-University, werth@informatik.uni-mainz.de

†Department of Computer Science, Johannes-Gutenberg-University, schoemer@informatik.uni-mainz.de

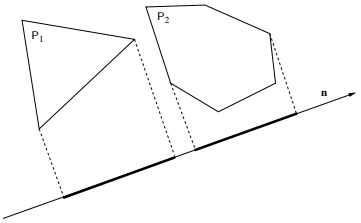


Figure 1: Two polygons in $\mathbb{R}^2$ and a separating axis.

4 Penetration depth

Definition 1 (Penetration depth of two objects) *The* penetration depth *of two (non-disjoint) objects* P_1 *and* P_2 *is the minimal distance that one object has to be moved in order to nullify the intersection.*

For convex polyhedra the separating axis theorem can be used as follows: If P_1 and P_2 intersect we will not find any separating axis as all projections correspond to pairwise overlapping line segments. Then the minimum of all overlaps is the penetration (c.f. fig (2)).

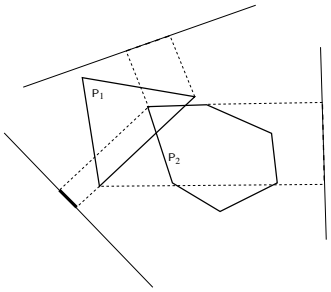


Figure 2: Two penetrating polygons in $\mathbb{R}^2$. We see 3 of the 9 possible projections including the one that gives us the penetration depth.

To compute the overlap we simply compare interval pairs and $\delta(I_1,I_2)$ gives us the penetration of two intervals I_1 and I_2:

Definition 2 *Let* $I_1 = [i_0,i_1]$ *and* $I_2 = [j_0,j_1]$ *be two (closed) intervals in* $\mathbb{R}$. *The* penetration of I_1 and I_2 *is given as*

$$\delta(I_1,I_2) \quad = \quad \frac{i_1-i_0}{2}+\frac{j_1-j_0}{2}-\left|\frac{i_1+i_0}{2}-\frac{j_1+j_0}{2}\right|$$

The penetration of two intervals indicates the minimal shift to glue two intervals together. If $\delta(I_1,I_2) < 0$ the intervals are disjoint and $-\delta(I_1,I_2)$ equals to the distance of I_1 and I_2.

For boxes, the number of axes to-be-tested is small, and hence they are applicable to this method. Only 15 axes are possible ($2\cdot 3$ for the faces of the boxes + $3\cdot 3$ pairwise cross products). We give a pseudo code for the penetration computation of two boxes here:

(1) $N = \{\mathbf{a}_i, \mathbf{b}_j, \dfrac{\mathbf{a}_i\times\mathbf{b}_j}{|\mathbf{a}_i\times\mathbf{b}_j|} \mid 1\le i,j\le 3\}$

(2) $d = \min\limits_{\mathbf{v}\in N}\{\delta([i_0,i_1],[j_0,j_1]) \mid$
$i_0 = \min\{\mathbf{v}^T\mathbf{x}\mid\mathbf{x}\in A\}, i_1 = \max\{\mathbf{v}^T\mathbf{x}\mid\mathbf{x}\in A\},$
$j_0 = \min\{\mathbf{v}^T\mathbf{y}\mid\mathbf{y}\in B\}, j_1 = \max\{\mathbf{v}^T\mathbf{y}\mid\mathbf{y}\in B\}\}$

(3) if $(d<0)$ $d=0$

In (1) we number all possible axis directions. Every box gets projected on every axis in (2) and the minimal overlap of two projection intervals defines the penetration. Step (3) is only needed if A and B do not intersect. In this case d would be negative. We may stop the computation and return 0 as soon as two intervals with negative penetration are found.

5 Pairwise distance

A modified version of the penetration depth algorithm can be used to find the minimum distance of two boxes A and B. Compared to other approaches like [3] this algorithm is easy to implement and ever easier to describe. The running time of this calculation increases slightly compared to the penetration method as we need to identify all separating axes to get the one with the *maximal separation*.

Given two disjoint boxes A and B with axis directions $\mathbf{a}_i$ and $\mathbf{b}_j$ for $1\le i,j\le 3$. In order to compute the minimal distance $d(A,B) = \min\{|\mathbf{y}-\mathbf{x}| \mid \mathbf{x}\in A, \mathbf{y}\in B\}$ we apply the following steps:

(1) $N = \{\mathbf{a}_i, \mathbf{b}_j, \dfrac{\mathbf{a}_i\times\mathbf{b}_j}{|\mathbf{a}_i\times\mathbf{b}_j|} \mid 1\le i,j\le 3\}$

(2) $\mathbf{n} = \operatorname{argmax}\limits_{\mathbf{v}\in N} \min\{|\mathbf{v}^T(\mathbf{y}-\mathbf{x})| \mid \mathbf{x}\in A, \mathbf{y}\in B\}$

(3) w.l.o.g. let $\mathbf{n}^T\mathbf{x} \le \mathbf{n}^T\mathbf{y}$ for $\mathbf{x}\in A$, $\mathbf{y}\in B$

(4) $A' = \operatorname{argmax}\limits_{\mathbf{x}\in A} \mathbf{n}^T\mathbf{x}$

(5) $B' = \operatorname{argmin}\limits_{\mathbf{y}\in B} \mathbf{n}^T\mathbf{y}$

(6) $d(A,B) = d(A',B')$

The algorithm again uses SAT. As the two boxes are disjoint there must be at least one separating axis. As there may be more than one we calculate the direction of maximal separation $\mathbf{n}$. The separation itself usually is not the distance; but it leads us to the two feature points A' and B' that are the closest extremal points of A and B in direction $\mathbf{n}$. Their distance defines the minimal distance of A and B.

An intuition why this method works correctly on boxes is given in the following figures: If $\mathbf{n}$ is the direction of one axis (e.G. A in fig. (3)) the extremal feature B' always is a vertex of B whereas A' is a point of the face with normal $\mathbf{n}$. Depending on the perpendicular dropped on the supporting plane of this face, A' either lies on the boundary of this face (vertex/edge or vertex/vertex) or in the inner part of

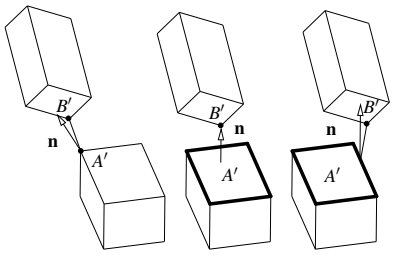


Figure 3: vertex-vertex, vertex-face and vertex-edge distance

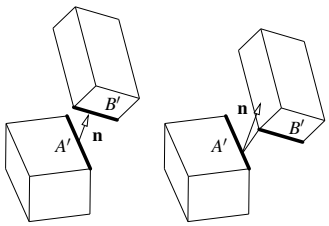


Figure 4: edge-edge and vertex-edge distance

the face (vertex/face). If $\mathbf{n}$ is the cross product of two edge directions (figure (4)) we are in the edge/edge case and the extremal features A' and B' are points on these edges. This case degenerates to a edge/vertex case if the perpendicular of the two carrier lines lies outside of one edge.

6 MTD

The close relationship between the penetration calculation and the distance computation is obvious; if the boxes are disjoint, compute the axis of maximal separation, if they intersect, compute the axis of minimal overlap, which is the maximal separation, this time with negative sign.

Having the maximal separation axis $\mathbf{n}$ on hand, we compute the maximal separation as

$$d \quad = \quad \min\{\mathbf{n}^T(\mathbf{y}-\mathbf{x}) \,|\, \mathbf{x} \in A, \mathbf{y} \in B\}.$$

For negative d we are in the intersection case, and the penetration is $-d$. Otherwise we have to compute the minimal features of A and B along $\mathbf{n}$ as described in the last section to obtain the distance. If we are looking for the MTD, which we defined as a direction vector, then it is simply given by $d * \mathbf{n}$.

7 Correctness

We gave an intuition for the correctness of the computation on boxes. SAT includes the correctness of the penetration computation for arbitrarily convex polygons. But is it correct that, if the object are disjoint, the two extremal features of A and B in direction of the maximal separation $\mathbf{n}$ define the Euclidean distance? In the following we give a proof for this. Note that this proof only relies on SAT and hence shows the correctness of the algorithm for any pair of convex polygons.

Lemma 2 *Given two polygons A and B. Assume that the minimal translational distance $MTD(A,B)$ is the distance of two points $\mathbf{a} \in A$ and $\mathbf{b} \in B$. Further let the axis of the maximal separation be $\mathbf{n}_m$. Then the maximal separation $s_m = \min\{\mathbf{n}_m^T(\mathbf{y}-\mathbf{x}) \,|\, x \in A, y \in B\}$ equals the projection of $(\mathbf{b}-\mathbf{a})$ onto $\mathbf{n}_m$.*

We will proof lemma (2) by use of some minor lemmas and helper functions. We assume that B is moving towards A along the shortest way, which of course is $(\mathbf{b}-\mathbf{a})$. Let T be the time when B reaches A. To simplify matters we will say that the time t is a scaled value in $[0,1]$ (or $t = \frac{\Delta t}{T}$). Hence $t = 1$ is the time when B collides with A. As we demand that B moves only translational, we can d fine a linear moving function for the location of point $\mathbf{b}(t) := \mathbf{b} + t(\mathbf{a}-\mathbf{b})$ which also defines the distance vector of A and B as

$$\mathbf{r}(t) \quad := \quad (1-t)(\mathbf{a}-\mathbf{b}) = (1-t)\mathbf{r}(0)$$

It is obvious hat $\mathbf{r}(0) = \mathbf{b}$ and $\mathbf{r}(1) = \mathbf{a}$. We can also define a linear function for the separation s_i of any separating axis $\mathbf{n}_i \in N$ as

$$s_i(t) \quad := \quad \mathbf{n}_i^T\,\mathbf{r}(t) - h_i$$

At this point, we do not precise the offset h_i, but we can show that it is constant over t. We also know, that the change of separation is linear in t (or rather in $(-t)$). We can further show that h_i is a non-negative value because the separation is bounded by $\mathbf{n}_i^T\,\mathbf{r}(t)$. Let us take a look at the maximal separation $s_m(t)$.

Lemma 3 *For all $t \in [0,1]$ the separation $s_m(t)$ is maximal.*

Proof. We pick any separation $s_j(t)$, $j \neq m$ and ask: At what time t' is $s_j(t') = s_m(t')$? A special case is $s_j(t) = s_m(t) \forall t$ which occurs when the (normalized and non parallel) vectors n_j and n_m lie on the same cone surface with height vector $\mathbf{r}(0)$. In the general case, the separation is a linear function over t and we know, there can be at most one t' such that $s_j(t')$ equals $s_m(t')$ which is given by means of the definition of s_j as

$$t' \quad = \quad 1 - \frac{h_i - h_m}{n_j^T\mathbf{r}(0) - n_m^T\mathbf{r}(0)}$$

A division by zero can be excluded for the general case. The nominator is always positive, because $h_j > h_m$ under the assumption that $s_m(0) > s_j(0)$. Then the denominator can be written as $(s_i(0) - s_m(0) + h_i - h_m)$ and because $s_i(0) < s_m(0)$, or $(s_i(0) - s_m(0)) = -\varepsilon$ with $\varepsilon > 0$ we get

$$t' = 1 - \frac{h_i - h_m}{h_i - h_m - \varepsilon} < 0$$

This means, we have equality only for a negative t'. But we defined $0 \leq t' \leq 1$. We know that $s_i(t)$ is continuous over t (and so is $s_m(t)$). Together with $s_m(0) > s_i(0)$ we can conclude $s_m(t) > s_i(t)$ for all $0 \leq t \leq t_0$. □

Lemma 4 *A and B collide at time t_0* iff *the maximal separation at time t_0 is zero.*

Proof. We show two directions.

"⇒" Assume that A and B collide at time t_0 and that the maximal separation is non-zero. This is a contradiction to the SAT. Therefore, $s_m(t_0)$ must be zero.

"⇐" Assume $s_m(t_0) = 0$ but A and B are disjoint. From SAT we know that there must be another non-zero separation. Let $s_k(t_0) > 0$ be this separation. From lemma (3) we know that that $s_k(t_0) < s_m(t_0)$. This is a contradiction to the assumption that A and B are disjoint. Hence A and B collide.

□

The proof of Lemma (2) now is just a conclusion of the last Lemma:

Lemma 5 *The maximal separation is given as* $s_m(t) = \mathbf{n}_m^T r(t)$.

Proof. We only have to show that $h_m = 0$. But this is obvious as s_m must have common roots with $\mathbf{r}(t)$. □

And with this conclusion, the proof of Lemma (2) is obvious.

8 Experimental results

No MTD methods specialized for boxes exist that we could compete against. Known distance algorithms for rectangles in $\mathbb{R}^3$ (c.f. [3]) already need 81 distinct tests and hence are hard to implement. Other implementation for convex polyhedra like [8] usually use costly Minkowski sums computation inspired by one of the major works on MTD for convex polyhedra ([1]). In [14] we find an efficient penetration depth and distance algorithm for polytopes and simple quadric objects like spheres, cones and cylinders using a variant of the Gilbert-Johnson-Keerthi (GJK) algorithm. Compared to the author's implementation of the GJK algorithm that is part of the SOLID library ([13]), our code runs 2.5 times faster for disjoint boxes and 5 times faster for intersecting boxes.

9 Conclusion

We gave a description for a fast and easy-to-implement MTD algorithm for boxes. It works for all convex polyhedra, whereas the complexity is linear in the number of faces and quadratic in the number of edges. But it is eligible for primitive objects and also works for mixed MTD computation, like box/triangle or other. The method presented is faster than other implementations and unlike most other implementation easy-to-implement.

References

[1] S. A. Cameron and R. K. Culley. Determining the minimum translational distance between two convex polyhedra. In *IEEE International Conference on Robotics and Automation*, pages 591–596, 1986.

[2] D. P. Dobkin and D. G. Kirkpatrick. Determining the separation of preprocessed polyhedra - a unified approach. In *ICALP '90: Proceedings of the 17th International Colloquium on Automata, Languages and Programming*, pages 400–413, London, UK, 1990. Springer-Verlag.

[3] D. H. Eberly. *3D Game Engine Design, Second Edition: A Practical Approach to Real-Time Computer Graphics (The Morgan Kaufmann Series in Interactive 3D Technology)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2006.

[4] Game Physics Simulation. *BULLET, Open Source Physics Library*. `http://bulletphysics.org/`.

[5] GAMMA research group. *RAPID, Robust and Accurate Polygon Interference Detection*. `http://gamma.cs.unc.edu/OBB/`.

[6] S. Gottschalk. *Collision Queries using Oriented Bounding Boxes*. PhD thesis, 1998.

[7] S. Gottschalk, M. C. Lin, and D. Manocha. OBB-tree: A hierarchical structure for rapid interference detection. *Computer Graphics*, pages 171–180, Aug. 1996. Proc. SIGGRAPH'96.

[8] Y. J. Kim, M. A. Otaduy, M. C. Lin, and D. Manocha. Fast penetration depth computation for physically-based animation. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 23–31, New York, NY, USA, 2002. ACM Press.

[9] E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha. Fast proximity queries with swept sphere volumes. Technical report, 1999.

[10] M. C. Lin and J. F. Canny. A fast algorithm for incremental distance calculation. In *In IEEE International Conference on Robotics and Automation*, pages 1008–1014, 1991.

[11] B. Mirtich. V-clip: fast and robust polyhedral collision detection. *ACM Trans. Graph.*, 17(3):177–208, 1998.

[12] NVIDIA. *PhysX, a powerful physics engine enabling real-time physics in leading edge PC games.* `http://www.nvidia.com/object/physx_new.html`.

[13] G. van Bergen. SOLID, software library for interference detection. `http://www.dtecta.com/`.

[14] G. van den Bergen. Proximity queries and penetration depth computation on 3d game objects. In *Game Developers Conference*, 2001.

The Tidy Set: A Minimal Simplicial Set for Computing Homology of Clique Complexes*

Afra Zomorodian†

Abstract

In this paper, we introduce the tidy set, a minimal simplicial set that captures the topology of a simplicial complex. The tidy set is particularly effective for computing the homology of clique complexes, such as the Vietoris-Rips and weak witness complexes. Our preliminary results show that tidy sets are orders of magnitude smaller than clique complexes, giving us a homology engine with small memory requirements.

1 Introduction

In this paper, we introduce the tidy set, a minimal simplicial set that captures the topology of a simplicial complex. Our method is effective in computing the homology of clique complexes without first constructing them. To inspire the reader, Figure 1 compares computing homology of a point set with a clique complex, the Vietoris-Rips complex, versus the tidy set. At the highest scale ϵ, the standard method constructs a simplicial complex with 46M simplices. Our method produces a simplicial set that has the same homology but is nearly three orders of magnitude smaller. With only 51K simplices, this tidy set is even smaller than the input point set. Our method is also five times faster than the standard method.

We are motivated by *topological data analysis*, where the goal is recovering the lost topology of sampled data [12]. The standard recovery process begins by approximating the underlying space of data using a combinatorial structure. For high-dimensional data, there are two methods popular in practice: the Vietoris-Rips complex [6] and the weak witness complex [4]. Both of these complexes are clique complexes of certain graphs, motivating our work.

Our approach is to avoid constructing the full complex by reducing it *during* its construction. We make reduction techniques effective by applying them to high-dimensional simplices only as simplices have exponential complexity. We maintain a compact representation of the structure throughout computation and postpone enumerating the structure and computing its homology as long as possible. Our methods in this paper are analogous to those by Mrozek et al. [9]. Our work was done independently from this group and applies to arbitrary-dimensional simplicial complexes, not three-dimensional cubical complexes. The implementation of our algorithms can process large datasets in arbitrary dimensions, datasets that are currently not handled by any existing software. It is also fast, takes less memory, and produces tidy sets that are orders of magnitude smaller, often sublinear in input size.

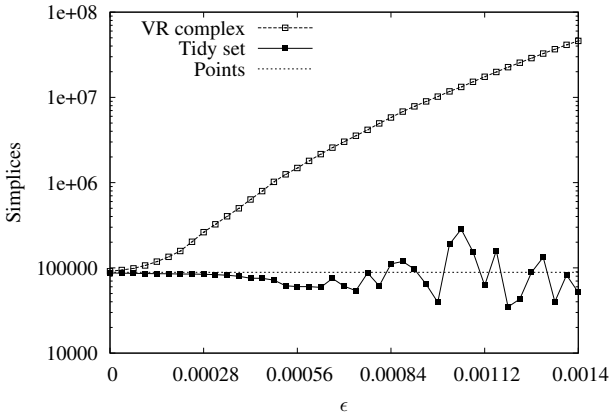


Figure 1: Size of the Vietoris-Rips complex versus our method, the tidy set, for set of 88,571 points in $\mathbb{R}^3$ at scale ϵ. Note: y-axis is log scale.

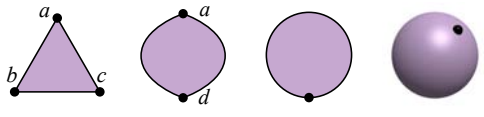
2 Background

A *simplicial complex* is a set K of finite sets, closed under the subset relation: If $\sigma \in K$ and $\tau \subseteq \sigma$, then $\tau \in K$. We then say that τ is a *face* of σ, its *coface*. A simplex is *maximal* if it has no proper coface in K. If $\sigma \in K$ has cardinality $|\sigma| = k + 1$, we call σ a *k-simplex* of *dimension* k, denoted $\dim \sigma = k$. K is d-dimensional if $d = \dim K = \max_{\sigma \in K} \dim \sigma$.

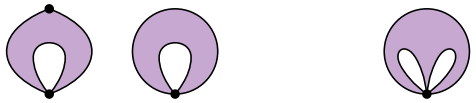
A *simplicial set* generalizes a simplicial complex to model a well-behaved topological space [3, 5, 8]. Informally, a simplicial set is like a simplicial complex where simplices may be collapsed to a point, and vertices may be identified. Simplicial sets allow these operations through *degenerate* simplices, such as an edge aa on vertex a. Figure 2 displays the seven pos-

*Research was partially supported by DARPA HR 0011-06-1-0038, ONR N 00014-08-1-0908, and NSF CAREER CCF-0845716.

†Department of Computer Science, Dartmouth College, Hanover, New Hampshire, afra@cs.dartmouth.edu



c	$(3,3,1)$	$(2,2,1)$	$(1,1,1)$	$(1,0,1)$
β	$(1,0,0)$	$(1,0,0)$	$(1,0,0)$	$(1,0,1)$



c	$(2,3,1)$	$(1,2,1)$	$(1,3,1)$
β	$(1,1,0)$	$(1,1,0)$	$(1,2,0)$

Figure 2: The 7 possible 2-simplices in a simplicial set. The triangle is the only one allowed in a simplicial complex. The rest have collapsed edges (top row) as well as identified vertices (bottom row). The vector c counts the non-degenerate simplices and the vector β holds the Betti numbers.

sible 2-simplices in a simplicial set. Simplicial sets may be constructed directly, but we are interested in sets that are derived from complexes by collapses only. Given a simplicial set X and $\sigma \in X$, the *collapse* of σ identifies it to a single point, giving us a new simplicial set $X' = X/\sigma$. Generally, we may also identify vertices in constructing simplicial sets. While we do not use vertex identifications in this paper, vertices may be identified by collapsed neighbors.

Simplicial homology extends naturally to simplicial sets. The nth homology group is $H_n(X) = \ker \partial_n / \operatorname{im} \partial_{n+1}$, where ∂_n is the nth boundary operator defined appropriately for simplicial sets. Given a subset $A \subseteq X$ that is a simplicial set, we may also define the *relative homology groups* $H_n(X, A)$, where we view the subset A as collapsed onto a single point. A *contractible* space has the homotopy type of a point, and it is often convenient for it to have trivial homology in all dimensions, including zero. For this, we define *reduced homology groups* $\tilde{H}_n(X)$, so that $H_0(X) \cong \tilde{H}_0(X) \oplus \mathbb{Z}$ and $H_n(X) \cong \tilde{H}_n(X)$ for $n > 0$. We say that a space X is *acyclic* if it has trivial reduced homology, i.e. $\tilde{H}_n(X) = 0$ for all n. Contractible spaces, such as simplices in a simplicial complex, are acyclic.

Suppose we are given a graph $G = (V, E)$. A *clique* is a set of vertices $Q \subseteq V$ that induces a complete subgraph in G. A clique is *maximal* if it cannot be made any larger. The *clique complex* $\mathcal{C}(G)$ has the maximal cliques of G as its maximal simplices [7]. Since subsets of cliques are also cliques, the clique complex is a simplicial complex. Both the Vietoris-Rips complex [6] and the weak witness complex [4], popular methods in topological data analysis, are clique complexes and motivate our work.

3 Tidy Set

In this section, we define the tidy set, a minimal simplicial complex. We begin by describing two reductions that we use in deriving the tidy set. We then describe a minimal representation for simplicial sets. We end by showing that the tidy set is minimal with respect to both reductions.

Our first reduction technique is trimming leaves. Intuitively, a leaf in a simplicial complex has an acyclic intersection with the rest of the complex, the intersection being its "stem". We generalize this notion for simplicial sets.

Definition 1 (leaf) *Let X be a simplicial set. A simplex $\sigma \in X$ is a* leaf *if for all n,*

$$H_n(Cl\,\sigma, Cl\,\sigma \cap Cl\,(X - Cl\,\sigma)) = 0.$$

Here, $\mathrm{Cl}\,\sigma$ is σ the *closure* of σ, or equivalently, σ as a simplicial complex, and $\mathrm{Cl}\,(X - \mathrm{Cl}\,\sigma)$ is the rest of the complex. In a simplicial complex, the definition simplifies to our earlier intuition.

Theorem 1 (complex leaf) *Let K be a simplicial complex and $\sigma \in K$ be a leaf. Then for all n,*

$$\tilde{H}_n(Cl\,\sigma \cap Cl\,(X - Cl\,\sigma)) = 0.$$

Leaves may be deleted without change in homology.

Theorem 2 *Let X be a simplicial set and $\sigma \in X$ be a leaf. Then for all n, $H_n(X) \cong H_n(Cl\,(X - Cl\,\sigma))$.*

The idea of removing leaves is not new. For instance, it is called *shaving* for full-dimensional cubes within cubical complexes [10].

Our second reduction technique is collapsing as defined earlier. Collapsing changes the category of the structure, from a simplicial complex to a simplicial set, but sometimes, does not change its homology.

Theorem 3 (collapse) *Let X be a simplicial set and $\sigma \in X$. If $Cl\,\sigma$ is acyclic, then for all n, $H_n(X) \cong H_n(X/\sigma)$.*

A key feature of our approach is that we use a minimal description for representing simplicial sets. We narrow our focus to sets that are complexes with collapsed maximal simplices.

Definition 2 ($\mathfrak{X}$) *Let Q and C be disjoint sets of maximal sets. Then $\mathfrak{X}(Q, C)$ is the simplicial set having the sets in Q as maximal simplices and the sets in C as collapsed maximal simplices. We use the tuple (Q, C) to denote $\mathfrak{X}(Q, C)$.*

Our representation is a natural extension of a minimum representation for simplicial complexes, with $\mathfrak{X}$ extending closure as the face enumeration operator.

Definition 3 ($\mathcal{Q}$) *Let K be a simplicial complex. We define $\mathcal{Q}(K)$ to be the set of maximal simplices.*

Theorem 4 (representation) *$\mathcal{Q}(K)$ is a unique representation of minimum size for K.*

For general simplicial sets, the representation is not unique or minimum sized, but makes reduction very easy.

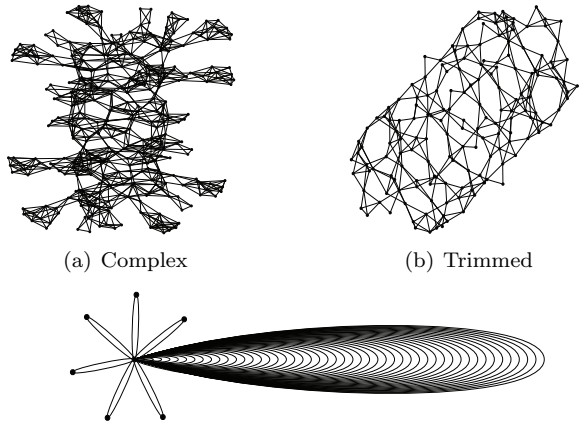
Definition 4 (trim & thin) *Let (Q, C) denote a simplicial set. For $\sigma \in Q$, we have two homology-preserving reductions:*

trim:	$(Q, C) \mapsto (Q - \{\sigma\}, C)$	*σ, a leaf*
thin:	$(Q, C) \mapsto (Q - \{\sigma\}, C \cup \{\sigma\})$	*σ, acyclic*

Note that both reductions maintain the invariant that Q and C are disjoint. Our representation also enables us to postpone enumeration of simplicial sets until we require it for computing homology.

Definition 5 (tidy set) *A* tidy set *is a trimmed, then thinned simplicial complex.*

That is, we first delete all maximal leaves in the input simplicial complex, then collapse all maximal acyclic simplices to get a simplicial set, as shown in Figure 3.



(a) Complex (b) Trimmed

(c) Tidy Set

Figure 3: 1-skeletons of homologous complexes: Complex (a), trimmed complex (b), and tidy set (c).

The order in which we do reduction is important. Indeed, it may seem reasonable to try trimming a tidy set again, as we currently trim complex leaves only. The following theorem states, however, that collapses do not grow new leaves.

Theorem 5 (minimal) *A tidy set has no leaves. That is, it is minimal with respect to trimming and thinning.*

That is, a tidy set is a minimal model for representing a simplicial complex.

4 Algorithms

In this section, we describe algorithms for computing the tidy set. Below, we assume that the procedure Betti(X) returns the reduced Betti numbers of a given simplicial set X in a vector by first computing homology over $\mathbb{Z}_2$ coefficients using the generalized persistence algorithm [13]. We use the same generic implementation of the algorithm for computing homology of both simplicial complexes and sets. We define Maximal-Sets(K) to be the function that returns the set of maximal simplices in K using a variant of an algorithm for maximal sets [11]. For clique complexes, we may compute our representation directly from the input graph G as maximal simplices are maximal cliques [1].

We define Greedy$(Q, C, \textit{Is-Reducible}, \textit{Reduce})$ to be the procedure that reduces the simplicial set denoted (Q, C) by the reduction technique specified by predicate *Is-Reducible* and action *Reduce*. The algorithm maintains a set of potentially reducible simplices, initializing it to Q. In each round, the algorithm collects reduced simplices in set and uses their neighbors in Q as candidates for the next round. We use this scheme, along with another greedy thinning algorithm, DFS-Thin, to define the procedure Tidy-Set(Q), which reduces a simplicial complex with maximal simplices Q and returns the tidy set as a tuple (Q, C).

To trim using the greedy scheme, we define the predicate Is-Leaf and action Trim. By Theorem 5, we only need to trim leaves in the simplicial complex, so we use the definition of complex leaves in Theorem 1. For this reason, Trim is exactly as in Definition 4, and $C = \emptyset$ is not used below.

```
Is-Leaf(σ, Q, C)
 1  I ← ⋃_{τ∈Q−{σ}} (σ ∩ τ)                    ▷ Intersection
 2  M ← Maximal-Sets(I)
 3  if |M| = 1                                  ▷ Single set
 4     then return TRUE
 5  elseif (max_{τ∈M} dim τ) > MAX-DIM
 6     then (Q_M, C_M) ← Tidy-Set(M) ▷ Recurse
 7          X_M = 𝒳(Q_M, C_M)
 8          if χ(X_M) ≠ 1
 9             then return FALSE
10             else  return Betti(X_M) = 0
11     else  K_M ← Cl M                         ▷ Direct
12          if χ(K_M) ≠ 1
13             then return FALSE
14             else  return Betti(K_M) = 0
```

To determine if σ is a leaf in the simplicial complex, the predicate Is-Leaf needs to check if the intersection I of σ and the rest of the complex is acyclic. The procedure computes I and represents it with maximal simplices M by using Maximal-Sets. If the intersection has only one maximal set, it immediately return

true as a maximal set corresponds to a simplex which is acyclic in a complex. If the intersection has high dimension, it recurses by using the procedure TIDY-SET as it now has a smaller instance of the original problem. It then enumerate the resulting simplicial set and checks if its reduced Betti numbers are all zero. Otherwise, when the intersection has low dimension, it directly computes homology by enumerating the simplicial complex. In both cases, it uses the Euler characteristic to skip homology computation whenever possible. Based on our experiments, we currently set MAX-DIM to 5.

We thin the trimmed complex in two phases, corresponding to complex and set thinning, respectively. Within a complex, all simplices are acyclic, but collapsing any simplex may cause a neighboring simplex to have non-trivial homology. For this reason, we attempt to find a large set of non-neighboring simplices that we may collapse at once. This idea may remind the reader of an *independent set*, a set of vertices in a graph that are pairwise non-adjacent. Indeed, vertices in an independent set of the dual graph correspond to non-intersecting maximal simplices. We may collect a larger set, however.

Theorem 6 *A simplex with one collapsed neighbor remains acyclic.*

Given this observation, we search the dual graph using *depth-first-search (DFS)* [2] to collect a set of acyclic simplices.

Having thinned the complex, we move into the category of simplicial sets. To thin using the greedy scheme, we define the predicate IS-ACYCLIC and action THIN, the latter of which is directly from Definition 4. The procedure now needs to enumerate the full simplicial set. As with trimming, we attempt to avoid computing homology by using the Euler characteristic.

IS-ACYCLIC(σ, Q, C)
1 $X_\sigma \leftarrow \mathcal{X}(\{\sigma\}, C)$
2 **if** $\chi(X_\sigma) \neq 1$
3 **then return** FALSE
4 **else return** BETTI$(X_\sigma) = 0$

5 Conclusion

In this paper, we define the tidy set, a minimal simplicial set, for computing homology of clique complexes, and give algorithms for computing it. There are a number of rich avenues for research. While we focus on clique complexes in this paper, our work applies to arbitrary simplicial complexes, provided we compute maximal simplices efficiently. We also have not applied our methods toward computing witness complexes: We may now use large sets of landmarks for massive datasets. We plan to extend our methods to filtered complexes to enable computation of persistence barcodes. Finally, almost every step of our method is parallelizable by design. Having reduced the memory requirement, parallelization is the next key step for topological analysis of massive datasets in high dimensions.

References

[1] F. Cazals and C. Karande. Reporting maximal cliques: new insights into an old problem. Research Report 5642, INRIA, 2005.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, Cambridge, MA, third edition, 2009.

[3] E. B. Curtis. Simplicial homotopy theory. *Advances in Mathematics*, 6:107–209, 1971.

[4] V. de Silva and G. Carlsson. Topological estimation using witness complexes. In *Proc. IEEE/Eurographics Symposium on Point-Based Graphics*, pages 157–166, 2004.

[5] S. Eilenberg and J. A. Zilber. Semi-simplicial complexes and singular homology. *Annals of Mathematics*, 51(3):499–513, 1950.

[6] M. Gromov. Hyperbolic groups. In S. Gersten, editor, *Essays in Group Theory*, pages 75–263. Springer-Verlag, New York, NY, 1987.

[7] D. Kozlov. *Combinatorial Algebraic Topology*. Springer-Verlag, New York, NY, 2008.

[8] J. P. May. *Simplicial Objects in Algebraic Topology*. D. Van Nostrand Co., Inc., Princeton, NJ, 1967.

[9] M. Mrozek, P. Pilarczyk, and N. Żelazna. Homology algorithm based on acyclic subspace. *Computers and Mathematics with Applications*, 55(11):2395–2412, 2008.

[10] P. Pilarczyk. Computer assisted method for proving existence of periodic orbits. *Topological Methods in Nonlinear Analysis*, 13:365–377, 1999.

[11] D. M. Yellin. Algorithms for subset testing and finding maximal sets. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*, pages 386–392, 1992.

[12] A. Zomorodian. Computational topology. In M. Atallah and M. Blanton, editors, *Algorithms and Theory of Computation Handbook*, volume 2, chapter 3. Chapman & Hall/CRC Press, Boca Raton, FL, second edition, 2010.

[13] A. Zomorodian and G. Carlsson. Localized homology. *Computational Geometry: Theory & Applications*, 41(3):126–148, 2008.

Author Index