Abstracts

21st European Workshop on Computational Geometry

Eindhoven

March 9-11, 2005

supported by

TU/e technische universiteit eindhoven

NWO

Nederlandse Organisatie voor Wetenschappelijk Onderzoek

Abstracts for the Twentyfirst
European Workshop on Computational Geometry
Technische Universiteit Eindhoven, The Netherlands
March 9–11, 2005

For information about obtaining copies of this volume contact

Mark de Berg
Department of Mathematics and Computer Science
TU Eindhoven
P.O. Box 513
5600 MB Eindhoven
The Netherlands
`m.t.d.berg@tue.nl`

Printed in the Netherlands.

Preface

This volume contains abstracts of the papers presented at the *21st European Workshop on Computational Geometry*, held at TU Eindhoven (the Netherlands) on March 9–11, 2005. There were 53 papers presented at the Workshop, covering a wide range of topics. This record number shows that the field of computational geometry is very much alive in Europe. We wish to thank all the authors who submitted papers and presented their work at the workshop. We believe that this has lead to a collection of very interesting abstracts that are both enjoyable and informative for the reader.
Finally, we are grateful to TU Eindhoven for their support in organizing the workshop and to the Netherlands Organisation for Scientific Research (NWO) for sponsoring the workshop.

Program Committee

Mark de Berg	TU Eindhoven
Joachim Gudmundsson	NICTA Sydney
René van Oostrum	Utrecht University
Bettina Speckmann	TU Eindhoven

Organizing Committee

Mark de Berg	TU Eindhoven
Joachim Gudmundsson	NICTA Sydney
Bettina Speckmann	TU Eindhoven
Micha Streppel	TU Eindhoven

Table of Contents

A Unified Algorithm for Adaptive Spacetime Meshing with Nonlocal Cone Constraints

Shripad Thite*

1 Introduction

Motivation Wave propagation is modeled by hyperbolic partial differential equations (PDEs) in both space and time variables, e.g., the wave equation $u_{tt} - \omega^2 u_{xx} = 0$ in 1D space × time. A *solution* to the PDE is a function $u(x, t)$ that satisfies the equation and the given initial and boundary conditions. The *wave velocity* ω, the velocity at which changes in physical parameters at a point (x, t) propagate to other points in the domain, may be a function of x and t as well as of u and its derivatives. The spacetime discontinuous Galerkin (SDG) finite element method is a numerical method to approximate the exact solution to the PDE. The SDG method approximates the solution within each spacetime element of a *mesh* of the spacetime domain as a linear combination of simple basis functions. The SDG method allows basis functions to be discontinuous across adjacent elements, which means that the mesh can even be nonconforming. A mesh constructed by standard techniques, such as a Delaunay triangulation, cannot be solved efficiently in general. Points in spacetime, and spacetime elements, are partially ordered by *causal dependence*—a point P influences another point Q if and only if changing the solution at P could possibly change the solution at Q. Spacetime elements must be solved in an order that respects this partial order. The efficiency of the solution technique depends on the number of elements that must be solved together because they depend on each other and are therefore *coupled*. In a Delaunay mesh, there is no guarantee on the size of a coupled system; we, on the other hand, construct an efficient mesh where this size is bounded.

Previous work Üngör and Sheffer [6] gave the first advancing front algorithm, TentPitcher, for meshing directly in 2D×Time given an initial *acute* triangulation of the space domain M. TentPitcher advances the solution over a piecewise linear triangulated *front*, a terrain over M; the initial front corresponds to $t = 0$ everywhere in space. The front at any step is *causal* which means that points on the front depend only on points in the past. At each step, the algorithm greedily advances in time a local neighborhood of a causal front τ to get a new causal front τ' and a set of new spacetime tetrahedra; causality of τ and τ' implies that the solution over the spacetime volume between τ and τ' can be computed immediately. Different parts of the front advance at different rates, depending on the local geometry, unlike with uniform time-stepping schemes. The result is a tetrahedral unstructured mesh Ω of $M \times [0, T]$ for any target time T. Erickson *et al.* [2] extended this algorithm to arbitrary spatial domains and higher dimensions, by imposing additional gradient constraints called *progress constraints* on each front.

Abedi *et al.* [1] gave an algorithm to adapt the size and—because of the causality constraint—also the duration of future spacetime elements to *a posteriori* estimates of numerical error. If the elements created at any step are too coarse, they are rejected and the front is refined by repeated bisection; the resulting smaller triangles lead to smaller spacetime elements. Coarsening or derefinement is performed when allowed by the error indicator.

Previous algorithms assumed a fixed upper bound on the wavespeed everywhere in spacetime or that the wavespeed function was Lipschitz. When the PDE is nonlinear, the wavespeed is not constant and also depends on the solution; the wavespeed at a given point in space can change discontinuously with time. Anisotropy of the medium means that waves propagate asymmetrically, with different speeds in different directions. This author [4] gave an algorithm that works even when the wavespeed increases discontinuously and in the presence of anisotropy. However, this algorithm did not adapt the spatial diameter of spacetime elements to numerical error estimates.

New results In this paper, we prove bounds on the worst-case *temporal aspect ratio* of spacetime tetrahedra constructed by TentPitcher; this ratio, together with the spatial aspect ratio, is likely to be correlated with the *quality* of the numerical solution. We also prove bounds on the size of the final mesh relative to a size optimal mesh.

Additionally, we give a unified algorithm that adapts the size of spacetime elements to error estimates while simultaneously adapting their duration to changing wavespeeds. This new algorithm meshes a given spacetime volume with many fewer tetrahedra in general than either of the previous two algorithms [1, 4].

*Department of Computer Science, University of Illinois at Urbana-Champaign; thite@cs.uiuc.edu

2 Temporal aspect ratio and mesh size

The initial front corresponds to time $t = 0$ everywhere in the space domain $M \subset \mathbb{E}^2$. Imagine time increasing upwards. At each step, TentPitcher lifts up a local minimum vertex P of a causal front τ to the vertex P' on a new causal front τ'. For every front triangle PQR incident on P, this creates a new spacetime tetrahedron $P'PQR$ in the volume between τ and τ', with a causal *inflow* facet PQR on τ and a causal *outflow* facet $P'QR$ on τ'. The volume between τ and τ' is called a *tent* and PP' is the *tentpole.* Causality implies that the solution everywhere in the tent can be computed immediately. The number of new spacetime tetrahedra is equal to the degree of P; these tetrahedra are coupled because they share non-causal facets. Only elements in a single tent are coupled; tents pitched at different local minima of τ are independent and are solved in parallel.

Suppose the wavespeed everywhere in spacetime is constant, equal to ω; let σ denote the *slope*, i.e., $\sigma = 1/\omega$. Causality means that the slope of PQR and of $P'QR$ must be less than σ. To guarantee nondegeneracy of tetrahedra, the front at each step must also satisfy so-called *progress constraints.* The causality and progress constraints limit the amount of progress made by the front at each step of the algorithm, i.e., the height of each tentpole; these constraints are functions of the slope σ and the shape of the local triangulation. The progress constraint that each front τ must satisfy depends on the causal slope encountered by the new front τ' in the *next* step which, in this case, is the same slope σ.

The *duration* of a spacetime element is the length of the shortest time interval that contains it. The *height* of a spacetime element is the length of the longest time interval contained in it. Our algorithms maximize the height of each spacetime element subject to causality and progress constraints. The progress guarantee of Erickson *et al.* [2] can be rephrased as follows: the height of each spacetime tetrahedra in the tent pitched at P is at least $\varepsilon\sigma w_p$ where $\varepsilon \in (0, \frac{1}{2}]$ is a fixed parameter and w_p denotes the distance of p from the boundary of the kernel of link(p) in the spatial projection. Thus, the height of the tentpole at P is bounded from below by a positive function of ε, the slope σ, and the shape of the triangles in star(p).

Temporal aspect ratio The *temporal aspect ratio* of a spacetime element is the ratio of the height of the element to its duration; this ratio is always in the range $(0, 1]$ with a larger value corresponding to a "better" element. The duration of the tetrahedron $P'PQR$ can be at most $2\sigma w_p$ because both facets PQR and $P'QR$ are causal. Together with the lower bound on the height of the tetrahedron, this implies the following theorem.

Theorem 1 *The temporal aspect ratio of any tetrahedron in Ω is at least $\varepsilon/2$.*

On size optimality TentPitcher constructs groups of coupled tetrahedra inside each tent such that the boundary of the tent is causal. This guarantees convergence of the DG solution [3]. Each tetrahedron constructed by TentPitcher has both a causal inflow facet and a causal outflow facet; additionally, the spatial projection of each tetrahedron $P'PQR$ is the triangle pqr in the original space mesh. Given a triangulation M of the space domain and a target time T, we say that a tetrahedral spacetime mesh of $M \times [0, T]$ is *valid* if (i) each tetrahedron has both a causal inflow facet and a causal outflow facet; and (ii) for every point x in the spatial projection Δ of each tetrahedron, the diameter of Δ does not exceed the diameter of the triangle of M containing x.

Fix an arbitrary point x in space. The *size* of a spacetime mesh of $M \times [0, T]$ is the maximum over $x \in M$ of the number of spacetime elements that intersect the temporal segment $x \times [0, T]$.

Theorem 2 *The size of the mesh constructed by TentPitcher is $\tilde{O}(1/\varepsilon^2)$ times the minimum size of any valid mesh of the spacetime volume $M \times [0, T]$.*

Proof. Let D and ρ denote the diameter and inradius respectively of the triangle pqr of M containing x. By causality, any temporal segment of duration $2\sigma D$ must intersect at least two distinct tetrahedra in a valid mesh; therefore, the number of spacetime tetrahedra in a valid mesh that intersect $x \times [0, T]$ is at least $\lfloor T/(2\sigma D) \rfloor$.

Consider a minimal sequence of tent pitching steps, called a *superstep*, in which each vertex of $\triangle pqr$ is lifted at least once. When p is pitched, the amount of progress made by x is proportional to $\mathrm{dist}(x, \overleftrightarrow{qr})$. Since $\mathrm{dist}(x, \overleftrightarrow{qr}) + \mathrm{dist}(x, \overleftrightarrow{rp}) + \mathrm{dist}(x, \overleftrightarrow{pqr}) \geq \rho$, the amount of progress made by x during a superstep is at least $\varepsilon\sigma\gamma\rho$, where $\gamma \in (0, 1]$ denotes the minimum of $w_p/\mathrm{dist}(p, \overleftrightarrow{qr})$, $w_q/\mathrm{dist}(q, \overleftrightarrow{rp})$, and $w_r/\mathrm{dist}(r, \overleftrightarrow{pq})$. Hence, after at most $\lceil T/(\varepsilon\sigma\gamma\rho) \rceil$ supersteps, the point x achieves or exceeds the target time T.

By causality, any two vertices of $\triangle pqr$ can advance in fewer than $4\sigma D$ consecutive steps without also advancing the third vertex. Therefore, the number of steps in each superstep is at most $\lfloor (4\sigma D)/(\varepsilon\sigma w) \rfloor$ where $w = \min\{w_p, w_q, w_r\}$. It follows that the number of tetrahedra in the spacetime mesh constructed by TentPitcher intersected by $x \times [0, T]$ is at most $\lceil T/(\varepsilon\sigma\gamma\rho) \rceil \cdot \lfloor (4\sigma D)/(\varepsilon\sigma w) \rfloor$.

The ratio of the upper bound to the lower bound on the size is $O\left(\frac{1}{\varepsilon^2} \frac{1}{\gamma} \frac{D^2}{\rho w}\right)$. □

3 Nonlocal cone constraints

The *cone of influence* of a point P is the set of points that depend on P. This cone has its apex at P and its slope in any spatial direction is the reciprocal of the wavespeed at P in that direction; fast waves correspond to cones with smaller slope. A front τ is *causal* if and only if τ lies below the *cone of influence* of every point P on τ; each such cone is a *causal cone constraint*. When the medium is *anisotropic*, the cones are asymmetric, e.g., with elliptical cross-sections. When the PDE is nonlinear or when the medium is anisotropic, a distant but fast wave, i.e., a *nonlocal* cone, can overtake a slower wave and hence limit the duration of new elements. Therefore, maximizing the progress of P, and thus the duration of new tetrahedra, requires querying the lower hull of the cones of influence. After the solution is computed on the new front, we obtain a new set of cone constraints. Maintaining the entire arrangement of cones of influence is expensive and unnecessary for our purpose; it suffices to obtain a cone that bounds (tightly) the actual cone of influence at P, i.e., to estimate a lower bound $\tilde{\sigma}(P)$ on the actual slope $\sigma(P)$. We assume the *absence of focusing*, which means that the cone of influence of any point P is contained in the cone of influence of every other point Q such that P depends on Q. Thus, the slope $\sigma(P)$ at P is bounded so that $0 < \sigma_{\min} \le \sigma(P) \le \sigma_{\max} < \infty$.

When the wavespeed at a point in space can increase discontinuously, cone constraints must be enforced on the front at every step that depend on the front in the *next* step. We give an algorithm that looks ahead k steps of the algorithm to estimate the slope on future fronts. The lookahead parameter k can be fixed or chosen adaptively. When $k = 0$, we assume that the minimum slope $\sigma_{\min}$ occurs on the front in the next step, so our estimate of future wavespeed is $\tilde{\sigma} = \sigma_{\min}$. When $k > 0$, we can use the current estimate to compute the next front and the actual slope on this new front to refine our previous estimate. We repeat this process either until subsequent iterations fail to improve the estimate $\tilde{\sigma}$ of future slope or until sufficient progress has already been ensured by the current estimate. (See [4] for the case $k = 1$.) To efficiently query the arrangement of cones, we use a discrete *bounding cone hierarchy* induced by a balanced partition of the triangular front, similar to a bounding volume hierarchy used in collision detection.

We will prove a minimum progress guarantee of $T_{\min} > 0$, a function of the local shape of the triangulation and the global minimum slope $\sigma_{\min}$, similar to that proved by Abedi *et al.* [1].

Definition 1 (k-progressive) *Let $\triangle ABC$ be a given triangle with A as its lowest vertex. We inductively define $\triangle ABC$ as k-progressive if*

(1) $\triangle ABC$ is causal;
(2) Let $\triangle A'BC$ denote the triangle after lifting A by T_{min} to A'. Then, $\triangle ABC$ must satisfy progress constraint $\sigma(A'BC)$ and $\triangle A'BC$ must be $\max\{0, k-1\}$*-progressive.*

***Base case** $k = 0$**:** $\triangle ABC$ is 0-progressive iff it satisfies progress constraint σ_{min}.*

4 Adaptive refinement and coarsening

Abedi *et al.* [1] gave an adaptive algorithm by strengthening the progress constraints due to Erickson *et al.* [2]. (This author later [5] corrected an oversight in their proofs, also slightly improving Tent-Pitcher.) The adaptive algorithm refines a triangle on the front using newest-vertex bisection; repeated bisection of a single triangle gives rise to at most 8 predictable homothetic shapes of front triangles. The front is coarsened by undoing previous refinement. The adaptive algorithm enforces a progress constraint on $\triangle PQR$ at every step that anticipates all the different shapes that can be obtained from $\triangle pqr$ by refinement. However, the algorithm of Abedi *et al.* [1] does not anticipate changes in the slope and assumes the minimum slope at every step.

The crucial observation we make here is that a cone of influence that limits the progress of $\triangle PQR$ may not limit the progress of a smaller triangle $\triangle ABC$, a descendant of $\triangle PQR$ by refinement. (The converse is also true.) Specifically, we observe that $\tilde{\sigma}(ABC) \ge \tilde{\sigma}(PQR)$. We obtain a unified algorithm by relaxing the progress constraints of Abedi *et al.* to allow a *child* triangle after by newest-vertex bisection to satisfy a potentially weaker progress constraint than its larger *parent* triangle. A potential drawback is that coarsening two *sibling* triangles coplanar on a front may require both triangles to satisfy a progress constraint *stricter* than their individual progress constraints. This can make coarsening requests harder to satisfy in practice; however, refinement can always be performed when necessary.

A key property we use in deriving the new progress constraint is that the boundary of a cone is a *ruled surface*; if any triangle $\triangle ABC$ intersects a given cone this intersection is a line segment in the plane of $\triangle ABC$. Therefore, if the bisector edge AD intersects a cone of influence then so do at least two of the edges of $\triangle ABC$ (Figure 1). Therefore, if a fast wavespeed in the future intersects AD but not an edge of $\triangle ABC$, then it can do so only if $\triangle ABC$ has been bisected at least once.

Fix ε, φ such that $0 < \varepsilon < \varphi < (1+\varepsilon)/2 < 1$. For any triangle $\triangle abc$ with newest-vertex a, we define the *diminished width* of $\triangle abc$, denoted by $\tilde{w}(abc)$, as the minimum of $(1-\varepsilon)\operatorname{dist}(a, \overleftrightarrow{bc})$, $(1-\varphi)\operatorname{dist}(b, \overleftrightarrow{ac})$, and $(1-\varphi)\operatorname{dist}(c, \overleftrightarrow{ab})$.

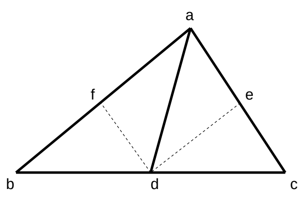


Figure 1: $\triangle abc$ after newest-vertex bisection.

Definition 2 (Progress constraint σ) *A triangle $\triangle ABC$ (Figure 1) satisfies* progress constraint σ *if and only if the applicable constraint from the following list is satisfied:*

If a is the lowest vertex:	$\lvert t(b) - t(c)\rvert \le 4\tilde{w}(fda)\sigma$
If b is the lowest vertex:	$\lvert t(a) - t(c)\rvert \le 2\tilde{w}(dab)\sigma$
If c is the lowest vertex:	$\lvert t(a) - t(b)\rvert \le 2\tilde{w}(dca)\sigma$

Definition 3 ((k,l)-progressive) *We inductively define a triangle $\triangle PQR$ as (k,l)-progressive if it is k-progressive (Definition 1) and any child $\triangle ABC$ obtained by newest-vertex bisection is $(k, \max\{0, l-1\})$-progressive.*

***Base case** $l = 0$: $\triangle PQR$ is $(k,0)$-progressive if an arbitrary* descendant *$\triangle ABC$ obtained by newest-vertex bisections satisfies progress constraint $\tilde{\sigma}(PQR)$.*

A front is *progressive* if every triangle on the front is (k,l)-progressive for some $k, l \ge 0$. Our unified algorithm greedily maximizes the progress such that each front is (k,l)-progressive for some choice of k and l. The algorithm can be as complicated as desired. Definition 3 stresses the fact that our algorithm can optimize the choice of k and l, likely doing better than the theoretical guarantee; however, a simple choice of $k = l = 1$ may suffice in practice.

Lemma 3 *(1) If a front τ is progressive, then the front after bisecting a triangle of τ is also progressive. (2) If a front τ is progressive, then for any local minimum vertex P the front τ', obtained from τ by advancing P in time by T_{min}, is progressive.*

Proof. (1) By Definition 3, if a triangle PQR of the front τ is (k,l)-progressive, then either of the two smaller triangles after bisecting $\triangle PQR$ is (k,l')-progressive for $l' = \max\{l-1, 0\}$.

(2) This part was essentially proven by Abedi *et al.* (see [5]) when each triangle PQR on the front τ satisfies progress constraint $\sigma(P'QR)$. Our algorithm ensures $\triangle PQR$ satisfies progress constraint $\tilde{\sigma}(PQR)$, where $\tilde{\sigma}(PQR) \le \sigma(P'QR)$. Because the progress constraint is monotonic in the slope σ, $\triangle PQR$ automatically satisfies progress constraint $\sigma(P'QR)$. The algebraic details are straightforward [5] and are omitted here for lack of space, to appear in a forthcoming paper. $\square$

By induction on the number of tent pitching and refinement steps, we have the following theorem.

Theorem 4 *Given a triangular mesh $M \in \mathbb{E}^2$ and a target time value T, our algorithm builds a finite tetrahedral mesh of the spacetime domain $M \times [0,T]$, provided each triangle is refined only a finite number of times.*

5 Conclusion and open problems

We gave an advancing front spacetime meshing algorithm that unifies previous algorithms [1, 5, 4] which tackled nonlinearity and adaptivity separately. The unified algorithm constructs a 2D$\times$Time mesh with provable guarantees, for the efficient solution of nonlinear and anisotropic problems. We will report experimental results in the near future.

We would like to extend adaptivity to 3D and higher dimensions. The space domain often changes with time; for instance, in the simulation of rocket fuel combustion, the shape of the residual fuel changes with time. We propose to include other front modification operations, such as edge flips, in addition to pitching inclined tentpoles, into a new meshing algorithm. The new algorithm will track features in spacetime, such as phase and domain boundaries and shock fronts, by aligning mesh facets along such features. The SDG method accurately captures the discontinuous solution when mesh facets align exactly with such singular surfaces.

Acknowledgments The author thanks the other members of the CPSD spacetime group, especially Jeff Erickson and Robert Haber.

References

[1] R. Abedi, S.-H. Chung, J. Erickson, Y. Fan, M. Garland, D. Guoy, R. Haber, J. M. Sullivan, S. Thite, Y. Zhou. Spacetime meshing with adaptive refinement and coarsening. In *Proc. Symp. Comput. Geom. (SoCG)*, pp. 300–309, 2004.

[2] J. Erickson, D. Guoy, J. M. Sullivan, A. Üngör. Building space-time meshes over arbitrary spatial domains. In *Proc. 11th Int'l. Meshing Roundtable (IMR)*, pp. 391–402, 2002.

[3] K. Jegdic, R. Jerrard, R. Haber. Convergence of a causal spacetime discontinuous Galerkin method for Temple 2×2 systems. Preprint.

[4] S. Thite. Efficient spacetime meshing with nonlocal cone constraints. In *Proc. 13th Int'l. Meshing Roundtable (IMR)*, pp. 47–58, 2004.

[5] S. Thite. Spacetime meshing for discontinuous Galerkin methods. Ph.D. thesis proposal. Online at http://www.uiuc.edu/~thite/thesis, 2004.

[6] A. Üngör, A. Sheffer. Tent-Pitcher: A meshing algorithm for space-time discontinuous Galerkin methods. In *Proc. 9th Int'l. Meshing Roundtable (IMR)*, pp. 111–122, 2000.

Quality Triangulations Made Smaller

Alper Üngör*

Abstract

We study alternative types of Steiner points (to circumcenters) for computing quality guaranteed Delaunay triangulations in three dimensions. We show through experiments that their effective use results in smaller (in the number of tetrahedra) triangulations than the output of the traditional circumcenter refinement methods.

1 Introduction

We consider the following optimization problem: *Compute the smallest size triangulation of a given domain such that all the simplices in the triangulation are of good quality.* Quality constraint is motivated by the numerical methods used in many engineering applications. A simplex is said to be good if its radius-edge ratio (circumradius over shortest edge length) is bounded from above. Under the quality constraint, our objective is to make the triangulation size as small as possible for their efficient use in the applications. There has been quite a few solutions for this problem [1, 2, 6, 8, 9]. Earliest algorithms that provide both size optimality (within a constant factor) and quality guarantee used balanced quadtrees to generate first a nicely spread point set and then the Delaunay triangulation of these points [1]. Subsequently, Delaunay refinement techniques are developed based on an incremental point insertion strategy and provide the same theoretical guarantees [8]. Delaunay refinement has become much more popular than the quadtree-based algorithms mostly due to its superior performance in generating smaller triangulations. Due to its importance in a wide range of applications, this problem is frequently revisited and several versions of the Delaunay refinement is suggested [2, 6, 8, 9].

Delaunay refinement method involves first computing an initial Delaunay triangulation of the input domain, and then iteratively adding points called *Steiner points* to improve the quality of the triangulation. Traditionally, circumcenters of bad simplices are used as Steiner points [8, 9]. We recently introduced a new type of Steiner points, called *off-centers*, as an alternative to circumcenters and propose a new variant of the Delaunay refinement algorithm in two dimensions [11]. We showed that the off-center insertion algorithm generates size-optimal quality-guaranteed triangulations. Moreover, experimental study indicates that our refinement algorithm with off-centers inserts fewer Steiner points than the circumcenter insertion algorithms and results in smaller triangulations. This implies substantial reduction not only in triangulation time, but also in the running time of the subsequent application algorithms. In this extended abstract, we present recent research progress on off-center based Delaunay refinement. We extend the off-center definition to three dimensions and present preliminary experimental results.

*Dept. of Computer & Information Science & Engineering, University of Florida, ungor@cise.ufl.edu

2 Quality Triangulations in 2D

Replacing the circumcenters with off-centers enabled us to make progress both on theoretical and practical fronts. In theory, using off-centers, we first improved the earlier parallel complexity results [10], then designed the first time-optimal Delaunay refinement algorithm [5]. In practice, off-center insertion algorithm results in significant reduction in the output size (see Figure 1). It is now used in the popular Delaunay refinement software `Triangle`[1].

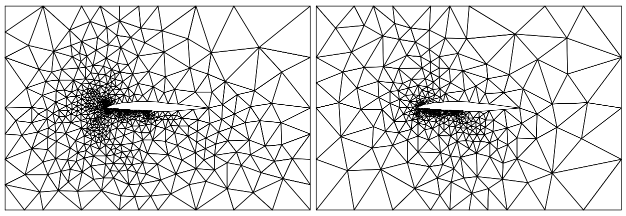
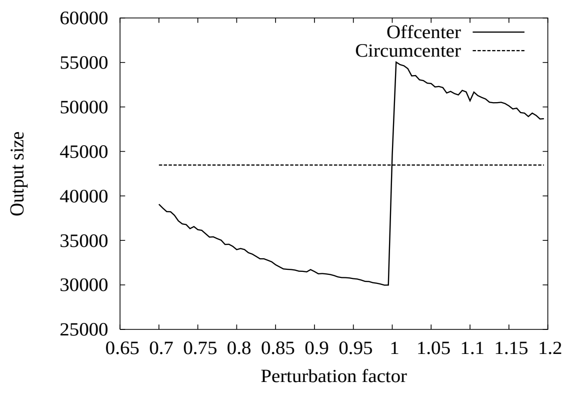


Figure 1: Airfoil mesh. Smallest angle in both output triangulations is 31°. Circumcenter insertion introduces 624 Steiner points resulting a mesh with 1222 triangles (left). Off-center insertion introduces only 359 Steiner points resulting a mesh with 699 triangles (right).

Off-center, c, of a bad triangle pqr is defined as the closest point to the circumcenter of pqr on the bisector of the shortest edge, say pq, such that pqc is (barely) a good triangle [11]. In our experiments we observed that, a perturbation from this theoretical definition gives the best results. We control the amount of perturbation by a parameter called α_1, which rescales the distance between the off-center and the shortest edge. While $\alpha_1 = 1$ means that there is no perturbation,

[1]Available at `http://www-2.cs.cmu.edu/~quake/triangle.html`



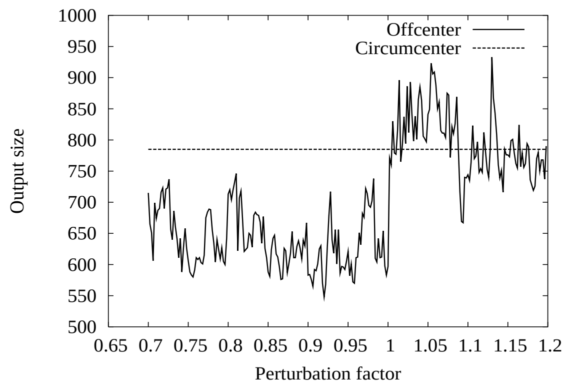


Figure 2: Impact of α_1 on the output size for random point (top) and airfoil (bottom) data sets.

$\alpha_1 < 1$ perturb the off-center on the bisector towards the shortest edge, and $\alpha_1 > 1$ move it away.

While the best choice for α_1 varies as we change the radius-edge ratio threshold and the data set, there is a clear pattern in the performance behavior. There is a sudden large shift in the output size from small to large as α_1 becomes larger than 1 (Figure 2). Best performance is usually observed when α_1 is in the interval (0.95, 1). Note that with a perturbation we not only make sure that the new triangle formed by the shortest edge points and the off-center is of good quality but also potentially fix more bad triangles at the same iteration.

3 Quality Triangulations in 3D

An extension of the circumcenter insertion algorithm to three dimensions is given by Shewchuk [9]. We briefly review this algorithm below and refer to [3, 9] for details.

3.1 Delaunay Refinement with Circumcenters

In three dimensions, a collection Ω of vertices, segments, and facets is called a *piecewise linear complex* (PLC) if (i) all lower dimensional elements on the boundary of an element in Ω also belong to Ω, and (ii) if any two elements intersect, then their intersection is a lower dimensional element in Ω [7]. We first compute the Delaunay triangulation of the set of vertices of the input PLC Ω. Then, we add new points (i) to recover the edges and facets that are not conformed by the Delaunay triangulation and (ii) to improve the quality of the triangulation. A point is said to *encroach* upon a simplex if it is inside the smallest sphere that contains the simplex. A tetrahedron is considered *bad* if its radius-edge ratio is larger than a pre-specified constant $\beta \geq 2$. We maintain the Delaunay triangulation as we add new points using the following rules.

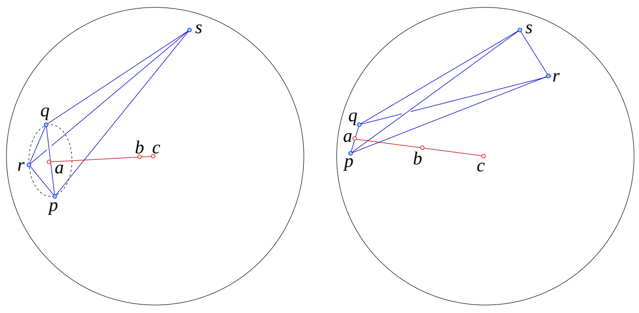
1. If a segment is encroached upon, we add its midpoint.
2. If a facet is encroached upon, we add its circumcenter unless Rule 1 applies.
3. If a tetrahedron is of bad quality, we add its circumcenter unless Rule 1 or 2 applies.

3.2 Delaunay Refinement with Off-centers

Here, we describe two new types of off-centers as Steiner points for three dimensional refinement.

3.2.1 Off-center on triangle bisector

Let pqr be the face of $pqrs$ with the smallest circumradius. Let a be the circumcenter of the triangle pqr, and c be the circumcenter of the tetrahedron $pqrs$. We call the ray that starts from a and goes through c, the bisector of the triangle pqr. We define the TYPE I *off-center* to be the circumcenter of $pqrs$ if the radius-edge-ratio of $pqrc$ is smaller than or equal to β. Otherwise, the TYPE I *off-center* is the point on the bisector of pqr, which makes the radius-edge ratio of the triangle based on p, q, r and the off-center itself exactly β (shown as b in Figure 3 (a)).



(a) (b)

Figure 3: Off-center on triangle and edge bisectors.

Let pq be the shortest edge of pqr and t be the circumradius of pqr. We compute the location of b by rescaling the length of the vector $c - a$ to $|ab|$:

$$|ab| = \alpha_2 \sqrt{\frac{2\beta^2|pq|^2 - t^2 + 2\sqrt{\beta^2|pq|^2\left(\beta^2|pq|^2 - t^2\right)}}{\|c-a\|^2}},$$

where $\alpha_2 \leq 1$ is the perturbation factor, similar to the one described in Section 2 for two dimensional off-center insertion. The choice of $\alpha_2 = 1$ means that the tetrahedron $pqrb$ is just good. Our experiments show that a good choice for α_2 is 0.9.

Note that we use this type of off-centers only if $\beta^2|pq|^2 > t^2$. Otherwise, the radius-edge ratio β cannot be satisfied with the location of b.

3.2.2 Off-center on edge bisector

The line that goes through the midpoint of an edge of a tetrahedron and its circumcenter is called the *bisector* of the edge. Given a bad tetrahedron $pqrs$, suppose that its shortest edge is pq. Let c denote the circumcenter of $pqrs$. We define the TYPE II *off-center* to be the circumcenter of $pqrs$ if the radius-edge-ratio of pqc is smaller than or equal to β. Otherwise, the TYPE II *off-center* is the point on the bisector (and inside the circumsphere), which makes the radius-edge ratio of the triangle based on p, q and the off-center itself exactly β (shown as b in Figure 3 (b)). We compute the length of ab as follows:

$$|ab| = \alpha_3 \left(\beta + \sqrt{\beta^2 - 1/4}\right) |pq|,$$

where α_3 is the perturbation factor.

When $\alpha_3 \leq 1$, diametral sphere of pqb has radius $\beta|pq|$, hence tetrahedra formed by p, q, b, and a fourth point x can be a good tetrahedron. As the value of α_3 approaches to 1, the chances of $pqbx$ being a good tetrahedron converges to 0. Experimentally, we found that a good choice for α_3 is 0.6. Note that the factor multiplying $|pq|$ above can be precomputed.

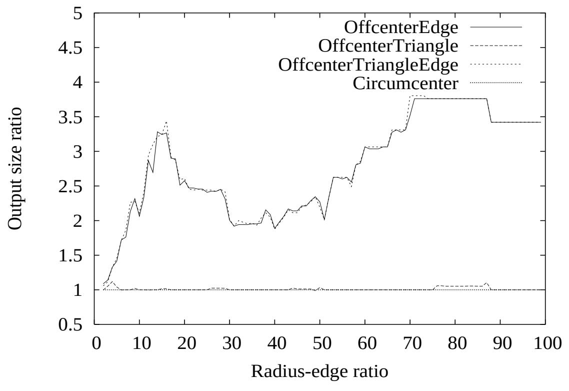
3.2.3 Algorithm

The structure of the Delaunay refinement algorithm as presented in Section 3.1, remains the same. We just replace the type of Steiner points used. The two types of off-centers give us the opportunity to explore several versions of the algorithm. We can use a single (either) type of off-center, or both. We give a comparison of these three approaches in the next section. When facets on the boundary are to be split, we use the two-dimensional off-center insertion algorithm.

3.2.4 Experiments

We implemented the Delaunay refinement with off-centers by replacing the circumcenter procedure in the `Pyramid` software. Computing off-centers and circumcenters are very similar and take roughly the same time. Hence, savings in the number of Steiner points reflects the amount of savings in triangulation time.

It is known that the insertion order of the Steiner points has an impact on the output mesh size. In this study, for fairness of comparison, we use the same ordering strategy (larger radius-edge ratio first) for both the circumcenter and the off-center insertion schemes. We shall note that there is room for further improvement by using a more appropriate ordering strategy for the off-center insertion method.



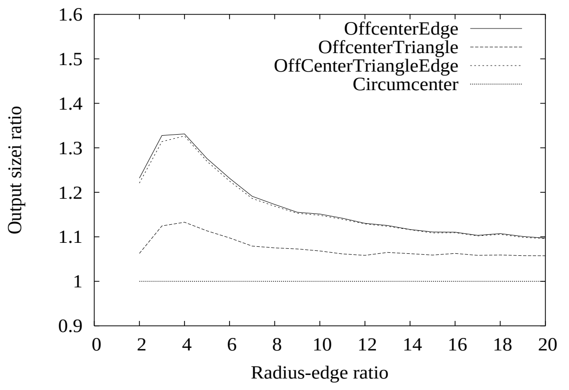


Figure 4: Output size ratio with respect to radius-edge ratio constraint for the tiny feature in the middle of a box (top) and ten thousand points on an ellipsoid (bottom) data sets..

Figure 4 presents a summary of our experiments on two data sets. First data set consists of a tiny feature (two vertices within a distance of 10^{-4}) located at the center of a unit box. Second data set consists of 10,000 points randomly located on the surface of an ellipsoid, which is contained inside a bounding box. We report the ratio of the output size M_c/M_o, where M_c and M_o are the number of elements generated by the circumcenter and the off-center insertion methods, respectively. We ran experiments on various data sets. In most cases, the difference in the output is visible (see Figure 5). We summarize our observations as follows:

- We get significant size improvements with the use of off-centers, especially when there is grading in the mesh (due to relatively small input features with respect to the domain size).
- Use of both type of off-centers or the use of TYPE II off-centers alone outperforms the use of TYPE I off-centers alone, which in turn outperforms the use of circumcenters.

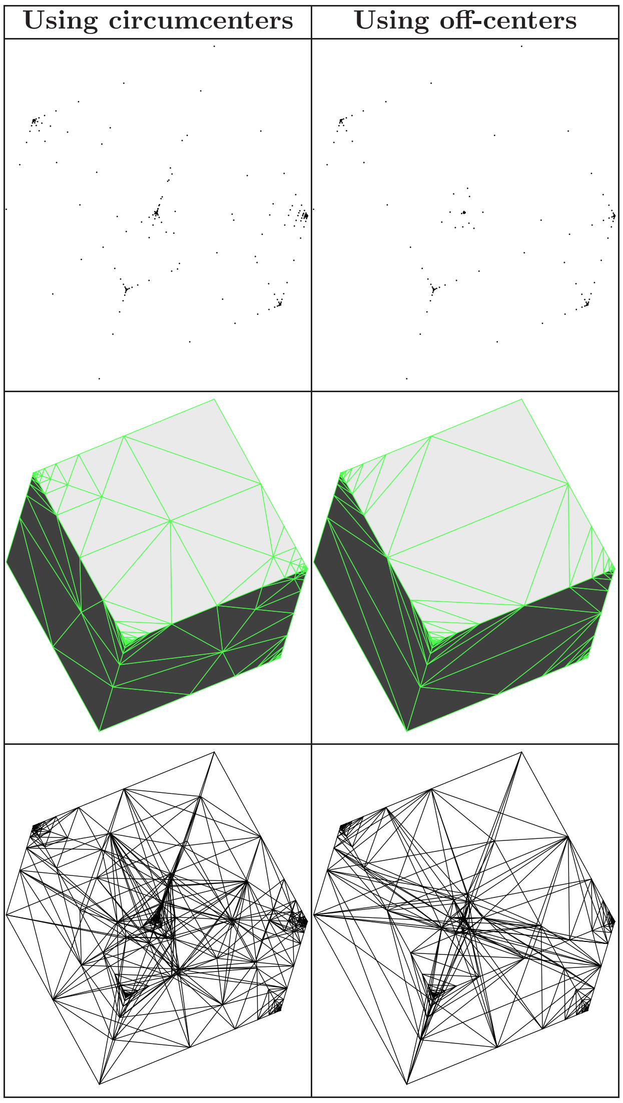


Figure 5: Input consists of 20 points and 6 facets. Largest radius-edge ratio in both triangulations is 15. The `Pyramid` software inserted 785 circumcenters resulting 1343 edges and 392 tetrahedra (left). Our algorithm inserted only 322 off-centers resulting 797 edges and 219 tetrahedra (right).

- Output size ratio M_c/M_o varies largely (more so than in two dimensions) as we change data sets.
- Performance behavior with respect to radius-edge ratio constraint (Figure 4) is somewhat different than that pattern in two dimensions [11], where we got the best size improvements for the smallest radius-edge ratio values.

4 Discussions

Our experimental study of the off-center insertion algorithm in three dimensions is by no means complete. Here, we described two types of off-centers as Steiner points and present how effective off-center insertion can be for computing small size quality-guaranteed triangulations. We should note that, off-center insertion do not always output smaller triangulations than the output of circumcenter insertion, especially when the perturbation factors α_1, and α_2 are not carefully chosen. We believe that it is worth to explore a perturbation strategy based on the local point distribution. In fact, our goal is to combine the off-center insertion algorithm with the perturbation based sliver removal approach presented in [4] to compute small size sliver-free triangulations in three dimensions.

Acknowledgements

I am thankful to Jonathan Shewchuk for using the off-center insertion algorithm in the `Triangle` software and for providing the `Pyramid` software. I also gratefully acknowledge the preliminary implementations Ethan Eade has done while taking my course at Duke University.

References

[1] M. Bern, D. Eppstein, and J. Gilbert. Provably good mesh generation. *J. Comp. System Sciences* 48:384–409, 1994.

[2] L.P. Chew. Guaranteed-quality triangular meshes. TR-89-983, Cornell Univ., 1989.

[3] H. Edelsbrunner. *Geometry and Topology for Mesh Generation.* Cambridge Univ. Press, 2001.

[4] H. Edelsbrunner, X. Li, G.L. Miller, A. Stathopoulos, D. Talmor, S.-H. Teng, A. Üngör, and N. Walkington. Smoothing and cleaning up slivers. *Proc. 32nd ACM Symp. on Theory of Computing*, 273–277, 2000.

[5] S. Har-Peled and A. Üngör. A time-optimal Delaunay refinement algorithm in two dimensions. *Submitted. Available at* `arXiv:cs.CG/0501007`.

[6] G.L. Miller. A time efficient Delaunay refinement algorithm. *Proc. ACM-SIAM Symp. on Disc. Algorithms*, 400–409, 2004.

[7] G.L. Miller, D. Talmor, S.-H. Teng, N. Walkington, and H. Wang. Control volume meshes using sphere packing: Generation, refinement, and coarsening. *Proc. 5th Int. Meshing Roundtable*, 47–61, 1996.

[8] J. Ruppert. A new and simple algorithm for quality 2-dimensional mesh generation. *Proc. 4th ACM-SIAM Symp. on Disc. Algorithms*, 83–92, 1993.

[9] J.R. Shewchuk. *Delaunay Refinement Mesh Generation.* Ph.D. thesis, Carnegie Mellon University, 1997.

[10] D.A. Spielman, S.-H. Teng, and A. Üngör. Parallel Delaunay refinement with off-centers. *Proc. EUROPAR*, LNCS 3149, 812–819, 2004.

[11] A. Üngör. Off-centers: A new type of Steiner points for computing size-optimal quality-guaranteed Delaunay triangulations. *Proc. LATIN*, LNCS 2976, 152–161, 2004.

The Relative Neighbourhood Graph is a part of every 30°−Triangulation

J. Mark Keil[*] Tzvetalin S. Vassilev[†]

Abstract

We study sets of points in the two-dimensional Euclidean plane. The relative neighbourhood graph (**RNG**) of a point set is a straight line graph that connects two points from the point set if and only if there is no other point in the set that is closer to both points than they are to each other. A triangulation of a point set is a maximal set of nonintersecting line segments (called **edges**) with vertices in the point set. We introduce angular rectrictions in the triangulations. Using the well-known method of exclusion regions, we show that the relative neighbourhood graph is a part of every triangulation all of the angles of which are greater than or equal to 30°.
Keywords: triangulation, relative neighbourhood graph, angular restriction, exclusion region

1 Introduction and basic definitions

Definition 1 *For a given planar set of points S, the* ***relative neighbourhood graph*** *of S, denoted by $RNG(S)$ consists of all edges AB, where $A, B \in S$, such that there is no point from S that is closer to both A and B than the distance AB.*

This definition is equivalent to saying that the region formed by the intersection of the circles with radii $|AB|$ centered at A and B is empty of points of S. This region is known in the literature as a **lune** of the edge AB. The lune is illustrated in Figure 1.
The relative neighbourhood graph has been extensively studied with relation to optimal triangulations. The RNG is a subsgraph of the Delaunay triangulation and the Gabriel graph of a point set, and it is a supergraph of the Minimum Spanning Tree of the point set [1]. The RNG is therefore a connected graph, with linear number of edges in n – the size of the point set S. The RNG is a subgraph of the MinMax Length triangulation as shown in [5]. Another interesting result there is that the RNG subdivides the convex hull of the point set into simple polygons, and each of these polygons contains at most one convex hull edge. The connectivity is a very important property of RNG as it automatically implies a polynomial time computability of any optimal triangulation of which it is a part. To be more exact, this can be done in $O(n^3)$ time and $O(n^2)$ space by Klincesk's algorithm (dynamic programming) [4].

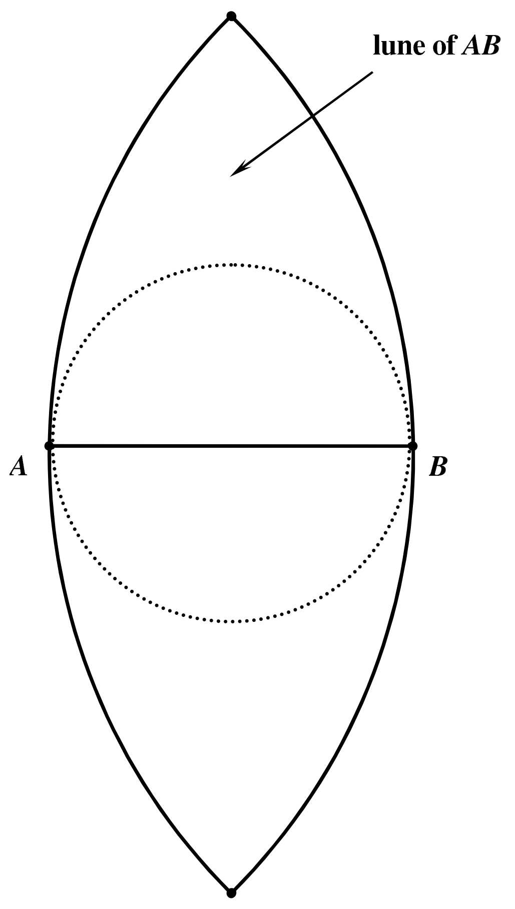


Figure 1: The lune of the edge AB

Definition 2 (α-triangulation) *Given a planar point set S and an angle α such that $0^\circ < \alpha \leq 60^\circ$, a triangulation T of S is called α-triangulation if and only if all the angles in the triangles of T are greater than or equal to α.*

Angular constraints are sometimes dicated by the application. In general it is considered that "fat" triangulations, i.e. triangulation with no small angles are more suitable for specific purposes as mesh generation, for example. Interesting experimental results on angle-constrained triangulations are presented in [2].

[*]Department of Computer Science, University of Saskatchewan, keil@cs.usask.ca
[†]Department of Computer Science, University of Saskatchewan, tsv552@mail.usask.ca

Definition 3 (Exclusion region) *Given a planar point set S and two points A and B of S, the exclusion region of the edge AB is a closed planar region with the property that its interior is empty of other points of S when the edge AB is a part of a specific (optimal) triangulation.*

Thus, the exclusion region is defined with reference to some quality measure or a specific property of the triangulation. The boundary of the exclusion region is usually a chain of line segments, circular arcs, etc. One classical example is the diamond-shaped exclusion region for the Minimum Weight Triangulation [3]. Conversely, we are interested in the property (which originally gave the name) that if the named region for an edge contains point(s) from the point set, the edge can be excluded from consideration as it will not be a part of any triangulation with the desired property. Next we discuss the exclusion region that results from introducing angular constraints, we call it a **forbidden zone**.

2 Angular constraints and forbidden zones

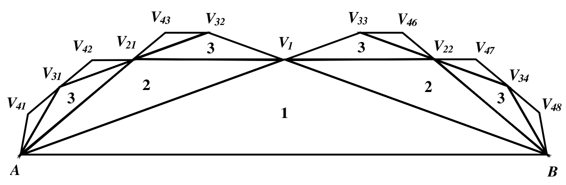


Figure 2: Construction of the i-th (for $i \leq 4$) order triangles for $\alpha = 20°$

Definition 4 (Forbidden zone) *Given a planar point set S and an angle α such that $0° < \alpha \leq 60°$ we call an edge AB, $A, B \in S$* ***internal*** *if there are points from S on both sides of the line AB. For an internal edge AB we define the isosceles triangle $\triangle AV_1B$ with angles $\angle V_1AB = \angle ABV_1 = \alpha$ as a* ***first order triangle*** *with respect to the edge AB. We call the point V_1 a* ***first order vertex*** *with respect to the edge AB. Similarly, we call the edges AV_1 and V_1B* ***first order edges*** *with respect to the edge AB. Note that the point V_1 might not be, and generally is not a point from the set S. It is just a part of an auxiliary construction. Recursively, on each i-th order edge we can build an isosceles triangle with base angles of α and it will be $(i+1)$-th order triangle with respect to the original edge AB. As it is clear from the construction method, for $i > 1$ there are multiple i-th order triangles, edges and vertices. The vertices in particular can be enumerated by double indexing V_{ik} meaning that V_{ik} is the k-th vertex of i-th order, where $i = 2, 3, \ldots, k = 1, 2, \ldots, 2^{i-1}$. The construction is illustrated in Figure 2. For each triangle of i-th order we can define its* ***free wedge*** *as the interior of the angle opposite of its internal angle of $180° - 2\alpha$. The union over all values of $i, i = 1, 2, \ldots$ of all free wedges for all i-th order triangles of the edge AB is the* ***free zone*** *of the edge AB. Note that some of the i-th order triangles lie entirely in the free wedges of triangles of lower order, thus not contributing to the free zone. Also some of the wedges overlap. The complement of the free zone of the edge AB is the* ***forbidden zone*** *of the edge AB.*

Lemma 1 *If there is a point of the set S in the forbidden zone of the edge AB, then this edge is not a part of any α-triangulation of S.*

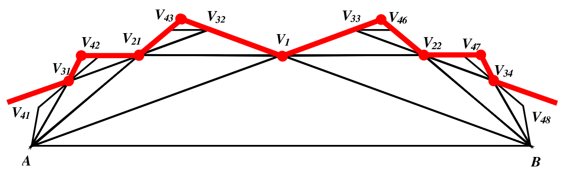


Figure 3: The forbidden zone of the edge AB, the boundary line (red) up to 3rd order

Proof. In fact the forbidden zone was built so as to ensure this property. To see that it is valid, consider the location of a point $X \in S$ inside the forbidden zone. Remember that by our assumption the edge AB is in some α-triangulation of S. By construction X is in the closure of some i-th order triangle of the edge AB. Suppose $i = 1$, i.e. the point X is in the first order triangle $\triangle AV_1B$. Then X is either connected directly to the edge AB, which forms an illegal triangle (both angles at A and B will be less than α), or there is another edge of the α-triangulation that intersects the interior of the triangle $\triangle AXB$. In the latter case, consider the "closest" to AB edge with this property, one of the points A or B has to be connected to an endpoint of this edge, thus violating the angular constraint. Therefore the first order triangle $\triangle AV_1B$ is empty of points of S. Let now $i = 2$ and assume that X is inside $\triangle V_1V_{22}B$. X cannot be connected directly to A as this will immediately violate the angular constraint. Thus there is an edge intersecting the interior of $\triangle V_1V_{22}B$. Similarly to the previous case there will be an edge emanating form B that either connects to X or to an endpoint of the "closest" edge. However the edge AB is connected to some point in the triangulation. Therefore we have two edges emanating from B inside an angle of less than 2α, and this violates the angular constraint. Inductively, by considering the edges of some triangulation emanating from the points A and B, and their intersection with the interiors of the triangles of order up to i we will be able to show that any triangulation containing the edge AB contains angle(s) smaller than

α which shows the validity of the argument.
In Figure 3 the boundary line of the forbidden zone is shown in red (bold). The forbidden zone extends on both sides of the edge AB, although in the construction we implicitly referred only to one of the half-planes defined by the line AB. □

Corollary 2 *Suppose that the edge AB has length $2a$, and that $\alpha = 30°$. The forbidden zone of the edge AB includes a rectangle with base AB and height $\frac{a}{\sqrt{3}}$. On top of this rectangle the forbidden zone includes two right triangles $\triangle V_{21}V_LV_1$ and $\triangle V_1V_RV_{22}$ with bases of $\frac{2a}{3}$ and angles of $30°$ at V_1. Refer to Figure 4.*

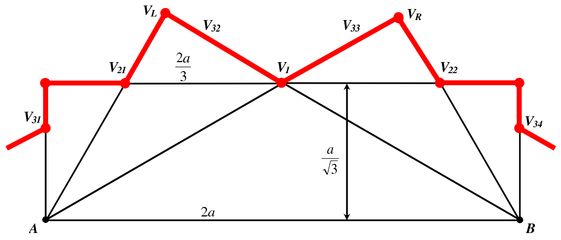


Figure 4: Parameters of the forbidden zone of the edge $AB = 2a$ for $\alpha = 30°$

3 Main result

We are going to show that the RNG is a part of every 30°–triangulation by showing that the forbidden zones of all possible edges that could intersect a given edge AB of the RNG contain at least one of the points A or B in their interior. Thus, in an 30°–triangulation, if such exists, the RNG edges cannot be intersected by other legal edges.
Keeping the assumptions of the previous section, let us denote the midpoint of the considered edge AB by O. Further, let us place the edge AB on the x–axis of a coordinate system with an origin in its midpoint O. The points A and B have coordinates $(-a, 0)$ and $(a, 0)$, respectively, where a is a positive real number.

Lemma 3 *Let PQ be a segment that goes through O and the points P and Q are on the boundary of the lune of AB. Then the point B is in the forbidden zone of the segment PQ.*

Proof. Without loss of generality we can assume that the point P lies in the I quadrant. Let $P(x, y)$, denote the orthogonal projection of the point B onto the segment PQ by H_B. The idea is to compute the distance BH_B and the depth of the forbidden zone of PQ at H_B and show that the first is less than or equal to the second quantity, thus establishing the claim. It is easy to see that H_B is a point that lies in the segment OP: the angles of triangle $\triangle OBP$ are all acute and BH_B is an altitude in this triangle. Using the fact that P lies on a circle centered at A with a radius of $2a$, and the law of sines we obtain $OH_B = \sqrt{\frac{a}{3a-2x}} \cdot x$ and $BH_B = \sqrt{\frac{a(a-x)(3a+x)}{3a-2x}}$. Denote the distance between O and P by p, and the distance between O and H_B by d. We can write the depth of the forbidden zone $B(d)$ as a function of d and p as follows:

$$B(d) = \begin{cases} (p+d)/\sqrt{3} & \text{for } 0 \le d \le p/2 \\ \sqrt{3}(p-d) & \text{for } p/2 \le d \le 2p/3 \\ p/\sqrt{3} & \text{for } 2p/3 \le d \le p \end{cases} \tag{1}$$

We have to verify, therefore, that $BH_B \le B(d)$ for all values of x and a. This is rather lengthy, but only involves basic mathematics and is omitted here. The result of this lemma was suggested by computational experiments done by the first author using the software package *Cinderella* ©. □

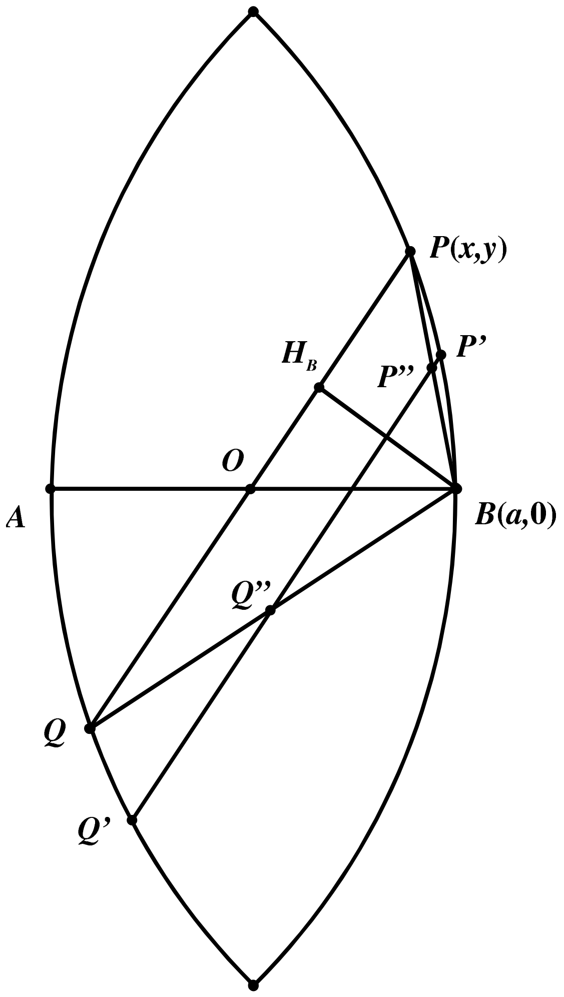


Figure 5: Edges PQ and $P'Q'$ intersecting the RNG edge AB in Lemmas 3 and 5

Corollary 4 *If a segment $P'Q'$ properly contains another segment $P''Q''$ then the forbidden zone of $P'Q'$ properly contains the forbidden zone of $P''Q''$.*

Proof. From the analytical geometry approach taken in the proof of Lemma 3, we can use Equation 1 which is valid in general, i.e. it describes the (left) part of the forbidden zone of a segment of length $2p$ with respect to a point that is at a distance d from the midpoint of that segment. Now, consider extension of the segment to the left by a length of $2h$. In analytical form it corresponds to the substitution:

$$\{p \leftarrow p - h, d \leftarrow d - h\}$$

It is now easily verified by Equation 1 that each point that was in the forbidden zone of the original segment is also in the forbidden zone of the extended segment. Similarly, an extension of the segment to the right by a length of $2h$ is equivalent to the substitution:

$$\{p \leftarrow p + h, d \leftarrow d + h\}$$

Again, Equation 1 verifies that any point that was in the forbidden zone of the segment before its extension is still in the forbidden zone after the extension. Figure 6 presents an illustration. □

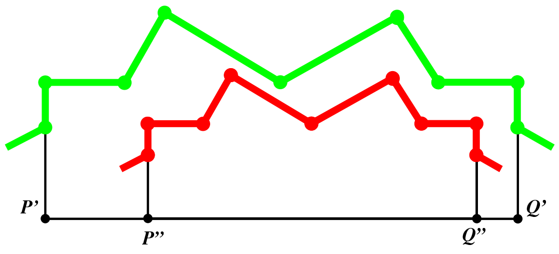


Figure 6: Edges $P'Q'$ and $P''Q''$ and their respective forbidden zone boundaries

Further, we have to consider possible edges that cross AB outside of its midpoint.

Lemma 5 *Let $P'Q'$ be a segment such that the points P' and Q' are on the boundary of the lune of AB. Let $P'Q'$ intersect AB at a point R such that R is between O and B. Then the point B is in the forbidden zone of the segment $P'Q'$.*

Proof. As in the proof of Lemma 3, assume that the point P' is in the I quadrant, further assume that the edge PQ through O is parallel to the edge $P'Q'$. The situation is illustrated in Figure 5. Construct the segment BP, by construction and the properties of the lune/circles, BP intersects $P'Q'$ in an internal point, which we denote by P''. Similarly, if we construct the segment BQ it will intersect $P'Q'$ at an internal point which we denote by Q''. By construction $\triangle PBQ \sim \triangle P''BQ''$ because of the fact that PQ and $P''Q''$ are parallel. Because of the similarity of the two triangles and the scaling property the fact that B is in the forbidden zone of PQ, established in Lemma 3, implies that B is also in the forbidden zone of $P''Q''$. Thus, we have two segments, namely $P''Q''$ and $P'Q'$ that satisfy the premises of Corollary 4. We conclude that B is in the forbidden zone of the edge $P'Q'$. Since any edge crossing AB is parallel to an edge crossing AB and going through its midpoint, the claim of this lemma is established. □

Theorem 6 *The relative neighbourhood graph of a planar set of points is part of every 30°–triangulation of this set (if such a triangulation exists).*

Proof. It is evident form Lemmas 3 and 5 that the edges of the relative neighbourhood graph of a planar point set S cannot be intersected by any other edge in a 30°–triangulation, if such a triangulation exists. Therefore, they must be in every 30°–triangulation, if such a triangulation exists. □

4 Conclusion

The result presented in this paper is tight. In other words, there is no guarantee that for an angle $\alpha < 30°$ the relative neighbourhood graph will be part of every (or any) α–triangulation. A four-point example can be constructed that shows this. Consider a pair of points A and B as per the notation used throughout this paper, and two other points C and D "slightly" outside the lune of AB, placed on the right side of the perpendicular bisector of AB infinitesimally close to it. Analysis shows that we can make the angles $\angle DCB$ and $\angle CDB$ as close to 30° as we want, while keeping the point B outside of the forbidden zone of the edge CD.

References

[1] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications, 2nd Edition.* Springer-Verlag, February 2000.

[2] Y. Dai, N. Katoh, and S.-W. Cheng. LMT-skeleton Heuristics for Several New Classes of Optimal Triangulations. *Computational Geometry: Theory and Applications.* Vol. 17, 2000, pages 51–68.

[3] R. L. Drysdale, S. McElfresh, and J. S. Snoeyink. On Exclusion regions for Optimal Triangulations. *Discrete Applied Mathematics.* Vol. 109, 2001, pages 49–65.

[4] G. T. Klincsek. Minimal Triangulations of Polygonal Domains. *Annals of Discrete Mathematics.* Vol. 9, 1980, pages 121–123.

[5] T. S. Tan. *Optimal Two-dimensional Triangulations.* Ph.D. Thesis, University of Illinois at Urbana-Champaign, Urbana, 1993.

Bounds on Optimally Triangulating Connected Subsets of the Minimum Weight Convex Partition

Magdalene Grantson   Christos Lecvopoulos [*]

Abstract

Given a set S of n points, we show that the length of
1) the minimum weight triangulation (MWT) of the minimum weight convex partition ($MWCP$) of S (T_{MWCP}) is at most $\Theta(n)$ longer than the MWT of S if collinearity of two or more edges is allowed and $\Theta(\log n)$ otherwise,
2) the MWT of the minimum spanning tree (MST) of the $MWCP$ of S ($T_{mst(MWCP)}$) is at most $\Theta(n)$ longer than the MWT of S if collinearity of two or more edges is allowed and $\Theta(\log n)$ otherwise,
3) the MWT of any connected subset G of the $MWCP$ of S ($T_{MWCP(G)}$) is at most $\Theta(n)$ longer than the MWT of S if collinearity of two or more edges is allowed.

1 Introduction

A triangulation of a set S of n points in the plane is a maximal set of non-intersecting edges connecting the points in S. The minimum weight triangulation MWT of S is a triangulation of minimum total edge length. It is unknown whether the MWT problem is NP-complete or solvable in polynomial time [2].

However, since the MWT of a simple polygon can be found in $O(n^3)$ time [3], it sounds reasonable to approximate the MWT of a point set by first connecting the set of points into a single component (a polygon). If the polygon is convex and no three vertices are collinear, a triangulation of weight $O(\log n)$ times the polygon's perimeter can be found by the *ring heuristic* of repeatedly connecting every second vertex [8]. Using this heuristic and a complicated method to partition the input into convex polygons, it was shown in [9] that a triangulation of $O(\log n)$ times the MWT length can be achieved.

Notation: We use the following abbreviations:

$MWCP$:	minimum weight convex partition
T_{MWCP}:	MWT of the $MWCP$
$mst(MWCP)$:	minimum spanning tree (MST) of the $MWCP$
$T_{mst(MWCP)}$:	MWT of $mst(MWCP)$
$MWCP(G)$:	a connected subset of the $MWCP$
$T_{MWCP(G)}$:	MWT of the $MWCP(G)$

*Dept. of Computer Science, Lund University, Box 118, 221 Lund, Sweden {magdalene,christos}@cs.lth.se

New Results:
1) The length of the T_{MWCP} of S is at most $\Theta(n)$ greater than the MWT of S if collinearity of two or more edges is allowed and $\Theta(\log n)$ otherwise.
2) The length of the $T_{mst(MWCP)}$ of S is at most $\Theta(n)$ greater than the MWT of S if collinearity of two or more edges is allowed and $\Theta(\log n)$ otherwise.
3) The length of the $T_{MWCP(G)}$ of S is at most $\Theta(n)$ greater than the MWT of S if collinearity of two or more edges is allowed.

2 Tight Bounds on T_{MWCP} and MWT of S

Theorem 1 *For any $n \geq 9$, there is a set S of n points in the plane, such that the T_{MWCP} of S can be $\Theta(n)$ longer than the MWT of S if collinearity of three or more vertices is allowed.*

Proof. For the lower bound we consider the set S of n points in Figure 1. S is symmetric and compressed w.r.t. the y-axis by a larger factor than shown in Figure 1 s.t. each diagonal between the convex hull pieces from v_7 to v_* and from v_{*+1} to v_n is of length at most $\frac{1}{n^2}$. The length of the diagonal connecting v_3 to v_4 is 1 and the length of the diagonals between (v_1, v_3), (v_2, v_3), (v_4, v_5), (v_4, v_6) are $\frac{1}{n}$. Consequently, the diagonals between (v_3, v_7), (v_3, v_{*+1}), (v_*, v_4), (v_4, v_n) have a length of about $\frac{1}{2}$ each for larger n.

The *only single* diagonal that can eliminate concavity at v_3 and v_4 after the insertion of diagonals between (v_1, v_3), (v_2, v_3), (v_4, v_5), (v_4, v_6) is the diagonal from v_3 to v_4. Let C be the convex hull piece from v_7 to v_*, and C' be the convex hull piece from v_{*+1} to v_n. (C and C' are straight lines.) An alternative elimination of the concavity at v_3 (resp. v_4) after the insertion of the diagonals (v_1, v_3), (v_2, v_3) (resp. (v_4, v_5), (v_4, v_6)) is to insert two diagonals, one from v_3 (resp. v_4) to a vertex on C, and the other from v_3 (resp. v_4) to a vertex on C'.

An $MWCP$ algorithm will always choose the diagonal between (v_3, v_4) of length 1 and the diagonals between (v_1, v_3), (v_2, v_3), (v_4, v_5), (v_4, v_6), since they give the minimum edge length convex partition. Including the convex hull CH of length about 2. This the total length of this convex partition is approximately $3 + \frac{4}{n}$. Any alternative convex partition which inserts two edges incident to v_3 and to v_4 results in an edge length of $(4 \pm \epsilon) + \frac{4}{n}$.

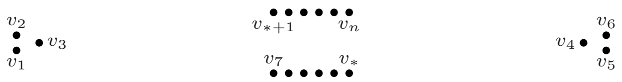


Figure 1: An approximate illustration of the set of points which shows the lower bound.

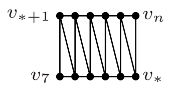


Figure 2: Approximate illustration of an optimal triangulation of area bounded by C and C'.

The T_{MWCP} of S includes the optimal triangulation of the sub-polygon q containing the vertices $(v_1, v_3, v_4, v_5, v_*, \ldots, v_7)$ and its symmetric counterpart q' containing vertices $(v_2, v_3, v_4, v_6, v_n, \ldots, v_{*+1})$. The sub-polygon q is triangulated by adding edges between v_3 and vertices on C and/or edges between v_4 and vertices on C. Each of these edges has a length of approximately $\frac{1}{2}$. For larger n there are about $\frac{n}{2}$ vertices on C (there are at least $\frac{2n}{9}$ vertices on C, since $n \geq 9$). Thus the total length of the edges needed to triangulate q is $\frac{1}{2} \cdot \frac{n}{2} = \frac{n}{4}$, and since q' is symmetric to q, the total length of the edges needed to triangulate both q and q' is $2\frac{n}{4} = \frac{n}{2}$. Adding the total edge length $3 + \frac{4}{n}$ for the $MWCP$ of S obtained above, we have that the T_{MWCP} of S has a total edge length of approximately $\frac{n}{2}$, for larger n.

The MWT of S, however, includes the diagonals between the convex hull CH, (v_1, v_3), (v_2, v_3), (v_4, v_5), (v_4, v_6), (v_3, v_7), (v_3, v_{*+1}), (v_*, v_4), (v_4, v_n) and diagonals going between C and C'. The optimal triangulation T of the area bounded by C and C' approaches zero for larger n, because each edge going between C and C' in T has length at most $\frac{1}{n^2}$ and there are $O(n)$ edges (see Figure 2). The MWT of S thus has a total edge length of about 4 for larger n. Hence $\frac{|T_{MWCP}|}{|MWT|} \approx \frac{n}{8}$.

For the upper bound, we draw on a result in [4], where it was shown that for a point set S any triangulation achieves a total edge length $O(n)$ times the MWT of P. Therefore the $\Theta(n)$ bound is tight. □

Theorem 2 *For any n, there is a set S of n points in the plane, such that the T_{MWCP} of S can be $\Theta(\log n)$ longer than the MWT of S if collinearity of three or more vertices is disallowed.*

Proof. To show the lower bound, we modify the set S of points in Figure 1 such that (1) on the convex hull piece C from v_7 to v_* the vertices lie on a circular arc so that no three vertices are collinear, likewise on the convex hull piece C' from v_{*+1} to v_n; (2) each edge between adjacent vertices on C and C' has length $\frac{1}{n}$. C and C' are both of length about 0.3; (3) the distance from each v on C (resp. C') to the closest vertex on C' (resp. C) is at most $\frac{1}{n^2}$; (4) the diagonals between (v_3, v_7), (v_3, v_{*+1}), (v_*, v_4), (v_4, v_n) have length of about 0.35 each.

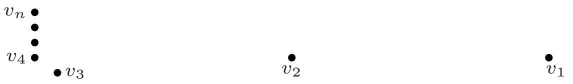


Figure 3: An approximate illustration of a point set S of points showing the lower bound.

An $MWCP$ algorithm always chooses the diagonal between (v_3, v_4) of length 1 and the diagonals between (v_1, v_3), (v_1, v_3), (v_4, v_5), (v_4, v_6), since they give the minimum edge length convex partition (similar explanation as in the proof of Theorem 1).

The T_{MWCP} of S includes the triangulation of the convex sub-polygon q containing vertices $(v_1, v_3, v_4, v_5, v_*, \ldots, v_7)$ and its symmetric counterpart q' containing vertices $(v_2, v_3, v_4, v_6, v_n, \ldots, v_{*+1})$. From [5, 6] we know that the *greedy triangulation*[1] of a convex polygon P is an $O(1)$ approximation of the MWT of P. The greedy triangulation of the $MWCP$ of S adds the diagonals between (v_7, v_*) and (v_{*+1}, v_n) before the diagonals (v_3, v_7) and (v_3, v_{*+1}) (resp. (v_*, v_4) and (v_4, v_n)) in q (resp. q'). The sub-polygon containing the circular arc C (resp. C') and the diagonal between (v_*, v_7) (resp. (v_{*+1}, v_n)) is referred to as a *semi-circular polygon* in [7]. [7] showed that the MWT of such semi-circular polygons has length $\Theta(\log n)$ times its perimeter. Thus the triangulations of such resulting sub-polygons have length $\Theta(\log n)$ plus the length of the perimeters of the sub-polygons. The length of the MWT of the two semi-circular polygons of S is $\Theta(\log n)$ (since the greedy triangulation of P is $O(1)$ of the MWT of P). Thus the total edge length of the T_{MWCP} of S is $\Omega(\log n)$.

A much shorter triangulation of S includes the diagonals between the vertices stated for the MWT in the proof of Theorem 1, giving a total edge length of at most $O(1)$. Using results from [9, 8] it can be deduced that given a set S (disallowing collinearity) partitioning the region of the plane enclosed by the CH of S into convex polygons one can achieve an $O(\log n)$ approximation to the MWT by triangulating the convex polygons. Therefore the $\Theta(\log n)$ bound is tight. □

3 Tight Bounds on $T_{mst(MWCP)}$ and MWT of S

Theorem 3 *For any $n > 0$, there exists a set S of n points for which the length of the $T_{mst(MWCP)}$ of S can be $\Theta(n)$ times the length of the MWT of P if collinearity of three or more vertices is allowed.*

[1]The greedy triangulation is obtained by repeatedly adding the shortest edge that does not lead to an intersection.

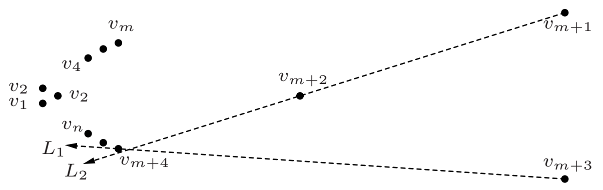


Figure 4: An approximate illustration showing the lower bound for the $T_{mst(MWCP)}$ and MWT ratio.

Proof. Consider the set S of $n \geq 10$ points in Figure 3 and let the distances between pairs of vertices be $d(v_1, v_2) = 1.2$, $d(v_2, v_n) = 1.2$, $d(v_1, v_n) = 2.4$, $d(v_2, v_4) = 1$, $d(v_1, v_3) = 2.3$, $d(v_2, v_3) > 1$, $d(v_3, v_4) \leq \frac{1}{n+2}$, $d(v_n, v_4) \leq \frac{1}{n+1}$, $d(v_n, v_3) \leq \frac{1}{n}$. We observe that the $MWCP$ includes the convex hull of S and the edges (v_1, v_2), (v_2, v_4), since the total edge length of this partition is minimum (concavity at v_2 is removed, since v_1, v_2, v_4 are collinear).

The $mst(MWCP)$ includes all edges in the $MWCP$ of S except (v_1, v_n) and (v_1, v_3). There are at most $n-5$ vertices between v_n and v_4, and each (including v_n) is connected to the vertex v_2 by an edge of length about 1.2 in the $T_{mst(MWCP)}$.

However, the MWT includes edges (v_2, v_n), (v_2, v_3) and edges from v_3 to each of the vertices from v_5 and v_n. Each of the $n-5$ edges from v_3 to vertices $(v_5, v_6, \ldots, v_n)$ has length at most $\frac{1}{n}$. The total length of the MWT of S is $O(1)$. Thus the ratio of the lengths of the $T_{mst(MWCP)}$ and the MWT is $\Omega(n)$. This proves a lower bound for the above problem.

For the upper bound we know that every triangulation has length $O(n)$ times the optimum (MWT) [4, 1]. Therefore the $\Theta(n)$ bound is tight. $\square$

Theorem 4 *For any $n > 0$, there exists a set S of n points for which the length of the $T_{mst(MWCP)}$ of S can be $\Theta(\log n)$ times the length of the MWT of S if collinearity of three or more vertices is disallowed.*

Proof. We construct a set S of n points, $n \geq 15$, which is sketched in Figure 4. We assume that S is compressed w.r.t. the y-axis s.t. the y-coordinate of each point is multiplied by $\frac{1}{n^2}$ and S has the following properties: (1) All vertices except v_3 and v_{m+2} (which lie on the x-axis) lie on the convex hull CH. (2) On the CH the vertices $v_{m+4}, v_{m+5}, \ldots, v_n$ lie on a circular arc. (3) Let $\delta(u, v)$ denote the *vertical distance* between any two given vertices u and v, and $d(u, v)$ the *distance* between u and v. Then $d(v_1, v_2) = \frac{1}{2n}$, $d(v_1, v_3) = \frac{2}{n}$, $\delta(v_3, v_n) = 0.2$, $\delta(v_n, v_{m+4}) = 0.1$ $\delta(v_{m+4}, v_{m+2}) = 0.7$, $d(v_3, v_{m+2}) = \delta(v_3, v_{m+2}) = 1$ $\delta(v_{m+2}, v_{m+3}) > 1$, $d(v_{m+3}, v_{m+1}) = 2$. The shortest diagonal from any vertex on the convex hull piece (v_n, v_{m+4}) to any vertex on the convex hull piece (v_4, v_m) is no longer than 1. (4) Let L_1 (see Figure 4) be the half-line extension of v_{m+1} to v_{m+2} and L_2 the half-line extension of v_{m+3} to v_{m+4}. The vertices v_{m+4} to v_n lie above the intersection of L_1 and L_2.

In any convex partition of S: The two diagonals (v_{m+2}, v_{m+1}) and (v_{m+2}, v_{m+3}) must both be present to remove the concavity at v_{m+2}, because even adding all remaining diagonals incident to v_{m+2} does not remove the concavity at v_{m+2} (this follows from property 4 above). To remove the concavity at v_3, if the diagonal (v_3, v_{m+2}) is added, at least two other diagonals incident to v_3 are needed. However, if the diagonal (v_3, v_{m+2}) is not added, either (v_1, v_3) or (v_2, v_3) together with at least two other diagonals are needed, namely one incident to v_3 and going to the left of v_3 and one incident to v_3 and going to the right of v_3.

To find the $MWCP$ of S, inserting the diagonal (v_3, v_{m+2}) removes the concavity at v_{m+2} and requires inserting either (v_1, v_3) and (v_2, v_3) to remove the concavity at v_3 giving a total edge length of $1 + \frac{4}{n}$. If we do not, however, add the diagonal (v_3, v_{m+2}), a possible solution is the insertion of either (v_{m+2}, v_{m+4}) or (v_{m+2}, v_m) at v_{m+2} and (v_n, v_3), (v_3, v_4) and either (v_1, v_3) or (v_2, v_3) to remove the concavity at v_3 giving a total edge length of $0.7 + 0.2 + 0.2 + \frac{2}{n}$. We conclude that the $MWCP$ of S includes the convex hull of S and edges (v_1, v_3), (v_2, v_3), (v_3, v_{m+2}), (v_{m+2}, v_{m+3}) and (v_{m+2}, v_{m+1}).

An $mst(MWCP)$ of S includes the edges in the $MWCP$ of S with the exception of (v_{m+4}, v_{m+3}), (v_m, v_{m+1}) and either (v_{m+2}, v_{m+1}) or (v_{m+2}, v_{m+3}).

The greedy triangulation of any $mst(MWCP)$ of S includes the triangulation of the so-called *semi-circular polygon* [7] bounded by the convex hull piece C from v_n to v_{m+4} and the edge (v_n, v_{m+4}), as well as its symmetric counterpart the convex hull piece C' from v_4 to v_m and the edge (v_4, v_m). In [7] it was shown that the MWT of such regular semi-circular polygons has length $\Theta(\log n)$ times its perimeter. Since the greedy triangulation is an $O(1)$ approximation of the MWT for any convex polygon [5, 6], the total edge length of the T_{msp} of S must be also $\Omega(\log n)$.

A shorter triangulation T of S includes the convex hull of S, the edges (v_1, v_3), (v_2, v_3), (v_3, v_4), (v_3, v_n), (v_{m+2}, v_{m+4}), (v_{m+2}, v_m), (v_{m+2}, v_{m+4}), (v_{m+2}, v_{m+1}), and edges from the triangulation of the area bounded by C, C', (v_4, v_n), (v_{m+4}, v_m) (see Figure 2 for a similar triangulation). In all there is a linear number of edges going between the area bounded by edges (v_4, v_n), (v_{m+4}, v_m), C and C', each of which has length at most $O(\frac{1}{n^2})$ giving a total edge length of $O(\frac{1}{n})$. Thus the total edge length in T is $O(1)$.

The set S considered above has an even number of points. For the case when n is odd we add a *dummy vertex* v_d in the area bounded by the triangle with corners v_1, v_2, and v_3 to maintain the symmetric nature of S. The introduction of the dummy vertex gives the

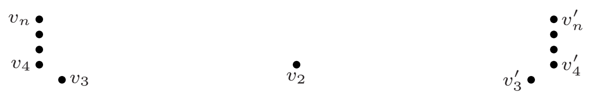


Figure 5: An sketch showing a modification of the point set S in Figure 3. The point sets are symmetric along the x-axis.The figure is not to scale.

same lower bound as shown for the even case since the concavity at v_d can be removed by inserting edges from v_d to v_1, v_2, and v_3.

The $\Theta(\log n)$ bound is tight. We can show this by starting with polygonal regions formed by combining the convex hull with the MST of the point set, and then computing the MWT of these regions using the *ring heuristics* proposed by Lingas [8]. The ring heuristics achieves a $O(\log n)$ approximation to the MWT of polygons. □

Generalization

Theorem 5 *For any n, there exists a point set S for which $\frac{|T_{MWCP(G)}|}{|MWT|} = \Omega(n)$.*

Proof. We show that Theorem 5 holds by modifying the point set S in Figure 3 to be symmetric w.r.t. the y-axis: the point v_2 lies on the y-axis, the points v_3 to v_n are at the same positions relative to v_2, and we add corresponding points v'_3 to v'_n at symmetric positions (see Figure 5). Any connected subset G in the $MWCP$ of S includes either the edge (v'_3, v_2) or (v_2, v_3). Any of these two edges prevents us from getting triangulation edges having length of at most $\frac{1}{n}$ as shown in the proof of Theorem 3. □

Observations

Definition 1 *A vertex of a polygon is strictly convex if its internal angle is strictly less than 180 degrees. Every vertex of a strictly convex polygon is also strictly convex. Similarly every polygon of a strictly convex partition is also strictly convex.*

Observation 1 *For the case where* strictly convex partitions *are required, the $T_{mst(MWCP)}$ of S is of length $\Theta(\log n)$ times the length of the MWT of S if collinearity of three or more vertices is allowed. We can prove the $\Omega(\log n)$ lower bound part of the $\Theta(\log n)$ bound using the same proof as for Theorem 4 (for the non collinear case), since a strictly convex partition means a strictly convex polygon, collinearity of three or more vertices is not allowed in the convex polygons formed from the strictly convex partitions. To show the $O(\log n)$ upper bound part of the $\Theta(\log n)$ bound, the ring heuristics [9] can be used to optimally triangulate all the strictly convex polygons derived from the strict $MWCP$ of S.*

References

[1] D. Eppstein. Approximating the Minimum Weight Triangulation *2nd ACM-SIAM Symposium on Discrete Algorithms* 48–57. 1992.

[2] M.R. Garey and D.S. Johnson. *Computers and Intractibility: A Guide to Theory of NP-Completeness.* W.H. Freeman, 1979.

[3] P.D. Gilbert. New Results in Planar Triangulations. *Report R-850.* University of Illinois Coordinated Science Lab 1979.

[4] D.G. Kirkpatrick. A Note on Delaunay and Optimal Triangulations. *Inform. process. Lett. 10*, 127-128. 1980.

[5] C. Levcopoulos and A. Lingas. Fast Algorithms for Greedy Triangulation. *Proc. 2nd Scand. Worksh. Algorithm Theory*, 238-250. Springer-Verlag, LNCS 447 1990.

[6] C. Levcopoulos and A. Lingas. On Approximation Behavior of the Greedy Triangulation for Convex Polygons. *Algorithmica 2*, 175-193. 1987.

[7] C. Levcopoulos, A. Lingas, and J. Sack. Heuristics for Optimum Binary Search Trees and Minimum Weight Triangulation Problems. *Theoretical Comp. Sci.*, 181–203. North-Holland, Amsterdam, Netherlands 1989.

[8] A. Lingas. A Linear Time Heuristic for Minimum Weight Triangulation of Convex Polygons. *Proc. 23rd Allerton Conference on Communication, Control and Computing*, 480-485. 1985.

[9] D. Plaisted and J. Hong. A Heuristic Triangulation Algorithm. *Journal of Algorithms* 8:405–437. Academic Press, San Diego, CA, USA 1987.

Incremental Construction along Space-Filling Curves

Kevin Buchin*

Abstract

For the incremental construction of a Delaunay triangulation, we prove that inserting points in rounds and walking along a space-filling curve in each round yields an algorithm running in linear expected time for uniformly distributed points. We complement this result by a simpler incremental construction running in linear expected time in any dimension.

1 Introduction

Motivation When devising an insertion order for the incremental construction of the Delaunay triangulation there are two seemingly conflicting goals: Inserting points randomly from the data avoids creating artificial triangles during the construction. In contrast, inserting points nearby allows taking advantage of geometric locality and locality of reference.

Randomized incremental construction follows the first approach. It is asymptotically optimal but performs poorly with modern memory hierarchies when used for large data sets as observed by Amenta, Choi and Rote [1]. They showed how randomness can be reduced without changing the asymptotic performance by a *biased randomized insertion order*: Points are randomly assigned to rounds of insertion of increasing sizes, and within a round the order of insertion can be chosen freely.

This allows us to use locality within the rounds by traversing the points of a round in an order along a space-filling curve [11]. We chose a space-filling curve order because it combines locality of reference with geometric locality by linearizing space, adapts well to irregularities of the point distribution, is fast to compute, is applicable in higher dimensions, and gives a good bound on the length of the resulting tour.

Related Algorithms Some linear expected time algorithms for constructing the Delaunay triangulation of uniformly distributed points from a bounded convex area in the plane are known [2, 6, 8]. Dwyer [6] gives an algorithm running in linear expected time for points from a sphere in any fixed dimension.

*Freie Universität Berlin, Institut für Informatik, buchin@inf.fu-berlin.de. This research was supported by the Deutsche Forschungsgemeinschaft within the European graduate program 'Combinatorics, Geometry, and Computation' (No. GRK 588/2).

Two incremental constructions running in linear time in practice on uniformly distributed points are known [9, 12]. In both cases, the analysis does not treat the irregularities near the boundary. The boundary case can be avoided by considering points from a Poisson point process.

Inserting near the Boundary For algorithms based on incremental construction, points near the boundary seem difficult to handle, because long and thin triangles slow down the point location. Figure 1 shows a typical case of this: Near the boundary, triangles with a large circumcircle are likely to occur in the triangulation, because a large part of the circumcircle may lie outside the region with points.

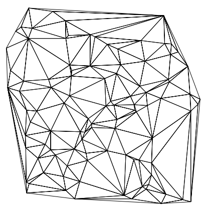


Figure 1: Delaunay triangulation of points in a square

Our main effort is to prove that the boundary case does not change overall linearity. While the analysis is done for our algorithm it seems possible to adapt the analysis to treat the algorithms mentioned above.

Surprisingly, we found another simple incremental construction which has no problems near the boundary and constructs the Delaunay triangulation in linear expected time in any fixed dimension. We include an analysis of this algorithm.

Contributions Our main contribution is to prove that a biased randomized insertion order together with a local insertion scheme runs in linear expected time on uniform points in a bounded convex region. This result complements the good practical performance of biased randomized insertion orders and resolves an open problem posed by Amenta et al. [1] This algorithm is the first completely analyzed linear expected time incremental construction algorithm for Delaunay triangulations.

The main technical contribution is the explicit analysis of point location near the boundary. Furthermore, we present an incremental construction running in linear expected time in any fixed dimension.

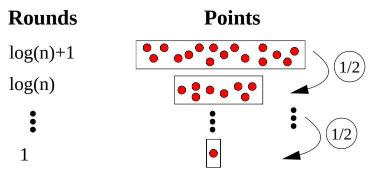


Figure 2: Assigning points to rounds

2 Walking along a Space-Filling Curve

Incremental Construction The basic concept of incremental construction is simple to state: Insert the points into the Delaunay triangulation one by one, updating the data structure after each insertion step. The time needed to insert a point consists of the time needed for locating the point in the current triangulation and the time for updating the triangulation. If the points are inserted in random order the expected total time needed for updating is in $O(n)$ and for point location is in $O(n \log n)$.

Biased Randomized Insertion Orders The order of insertion is allowed to deviate from a random order as long as randomness dominates. Sufficient randomness can be introduced to the insertion order by assigning the points independently at random to rounds as illustrated in Figure 2: A point is independently assigned to the last round with the probability of 1/2. Each of the remaining points is assigned to the next to last round with the probability of 1/2, and so on [1].

After a logarithmic number of rounds an expected constant number of points remain, and we can stop sampling and assign the remaining points to the first round. The points are inserted round by round. In a round points can be inserted in an arbitrary order.

Biased randomized insertion orders were originally introduced to reduce random memory access. We make use of the fact that they do not change the update cost, which, in our case, is linear. Therefore we can focus on the point location time.

Space-Filling Curves Within a round we construct a short tour through the points by the space-filling curve heuristic for the traveling salesman [10]. To see how the tour is constructed, consider the first steps of the geometric construction of the Hilbert Curve shown in Figure 3. The space is successively subdivided. The cells are ordered in such a way that consecutive cells in the order are adjacent.

The limit of this process yields a *space-filling curve*, i. e. a surjective mapping from the unit interval to the unit square or, more generally, to the d–dimensional unit cube. Formally, the space-filling curve heuristic sorts the points by selecting a preimage for every point and by sorting the points according to the preimages.

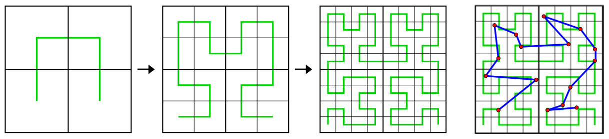


Figure 3: First steps in the construction of the Hilbert curve and a space-filling curve tour

In practice, the process can be stopped after a finite number of subdivisions. The maximal number of subdivisions necessary is the number of bits of precision. In order to achieve an $O(\sqrt{n})$ bound on the tour length a subdivision with as many cells as points is sufficient. Points within a cell can then be ordered in an arbitrary order.

Walking We traverse the tour and insert the points along the way. The next point is located by *walking* [5], i. e. by a local search starting at the current point and traversing the triangles stabbed by the line segment between the two points. This point location scheme does not need a point location data structure.

The heuristic can be used not only for points in the unit square but also in an arbitrary rectangle. The bound on the tour length changes by a factor of the length of the longer side. For points in a bounded convex region a bounding rectangle is used.

3 Analysis

Space-Filling Curve Heuristic The heuristic constructs a tour through a given set of points in the unit square by visiting them in the order of their preimages under a space-filling curve ψ. The order of preimages is not unique since ψ cannot be injective [11]. For the heuristic to be effective the images of nearby points on the side of the preimages should be near to each other in space. For space-filling curves this follows from their Lipschitz continuity of order 1/2, i. e. that for any s, t in the unit interval $|\psi(s) - \psi(t)| \leq c_\psi |s-t|^{1/2}$.

The space-filling curve heuristic was popularized by Platzman and Bartholdi [10]. A general treatment and probabilistic analysis is given by Gao and Steele [7]. We summarize the result we need in the following lemma:

Lemma 1 *For a space-filling curve that is Lipschitz of order 1/2 and can be generated by subdivision, an order on the points can be computed in linear time in such a way that for any k-subset of points the length of the tour through these points along the order is bounded by $O(k^{1/2})$.*

For this lemma no assumption on the point distribution is used. A stronger bound holds for points distributed uniformly in the unit square [7]. In d dimensions the bound generalizes to $O(n^{(d-1)/d})$.

Counting Intersections To analyze the running time it is sufficient to analyze the time required in the last round using an induction. Assume $m + n$ points distributed independently and uniformly at random in a bounded convex region C of area 1, where n points are already inserted in the Delaunay triangulation. To insert the m remaining points, a tour through the points is constructed using the space-filling curve heuristic.

The points are located by traversing the triangulation along the tour. Therefore, the time needed for locating the points is proportional to the number of intersections between the tour and the triangulation.

A bound on the expected number of intersections is obtained by considering *exclusion regions* for possible edges of the triangulations, i. e. if the region contains points on both sides of the possible edges the edge cannot be in the triangulation. For Delaunay triangulations the disc with the edge as diameter is an exclusion region. For uniformly distributed points the edges of a triangulation with exclusion regions typically are expected to be either short or near to the boundary. This can be strengthened to the following:

Lemma 2 (Devroye, Mücke and Zhu [5]) *The expected number of intersections between a Delaunay triangulation of points distributed independently and uniformly in a compact convex area C and a fixed line segment L that is at least distance $c_0\sqrt{\log n/n}$ from the boundary of C is bounded by*

$$c_1 + c_2|L|\sqrt{n},$$

where c_0 is a constant, and c_1 and c_2 depend only on the geometrical properties of C.

The bound on the tour length and the bound on the number of intersections together give a linear bound for all line segments that have a distance of at least $c_0\sqrt{\log n/n}$ from the boundary ∂C. The expected number of points near the boundary is bounded by $m' := c_0|\partial C|m\sqrt{\log n/n}$ and the number of line segments by $2m'$, and therefore, by Jensen's inequality and Lemma 1, the total length of these line segments by $c\sqrt{2m'}$ for suitable c.

To treat these segments we quantify what it means that the edges of the triangulation are likely to be short or near to the boundary:

Lemma 3 *Let T be the Delaunay triangulation of n points distributed independently and uniformly in a convex area C. Denote by $D_{w,l}$ the event that a Delaunay edge with an endpoint with a distance of at least w to the boundary of C is longer than l.*
For $c > 1$ and $l \geq cw$

$$\Pr(D_{w,l}) \leq n^2 e^{-(n-2)wl\sqrt{1-1/c^2}/2}.$$

In particular, if $l \geq 3w$ and $wl \geq 6\sqrt{2}\log n/n$, then $\Pr(D_{w,l}) \in o(1/n)$.

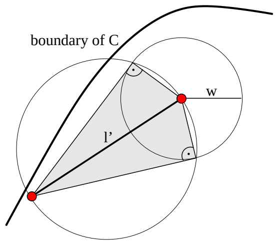


Figure 4: Exclusion region for a Delaunay edge that is contained in C

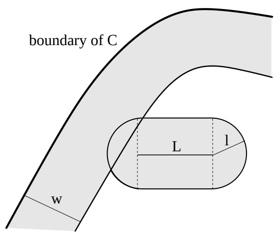


Figure 5: Area for endpoints of Delaunay edges intersecting L

Proof. Consider the edge in Figure 4 with length $l' \geq l$ and a vertex with a distance of more than w to the boundary of C. The two rectangular triangles form an exclusion region for the edge that is contained in C. The area of a triangle is bounded by $1/2 \cdot w\sqrt{l^2 - w^2} \geq wl\sqrt{1 - 1/c^2}/2$. There are $\binom{n}{2}$ possible edges and therefore

$$\begin{aligned}\Pr(D_{w,l}) &\leq \binom{n}{2} 2(1 - wl\sqrt{1-1/c^2})^{n-2}\\ &\leq n^2 e^{-(n-2)wl\sqrt{1-1/c^2}/2}\end{aligned}$$

□

This gives us a bound on the number of Delaunay edges that can intersect the line segments of the tour:

Lemma 4 *The expected number of intersections of a Delaunay triangulation and a tour along a Lipschitz-1/2 space-filling curve with a total number of N points which are distributed independently and uniformly in a convex area is linear in N.*

Proof. Assume $l \geq 3w$ and $wl \geq 6\sqrt{2}\log n/n$. With high probability only edges with endpoints with distance of at most l to one of the line segments, or with distance at most w from the boundary can intersect. For a single line segment this area is shown in Figure 5. The expected number of endpoints of edges that intersect a line segment L is therefore bounded by $n(|\partial C|w + \pi l^2 + 2l|L|) + o(1)$. Because of planarity there are at most three times that many edges intersecting L.

For k line segments of total length λ this yields a $3n(|\partial C|kw + \pi l^2 k + 2l\lambda) + o(k)$ bound on the number of intersecting edges. In our case, we have $k = c_0|\partial C|m\sqrt{\log n/n}$ and $\lambda = \sqrt{k}$. Choosing $w := \max(k^{-1/4}\sqrt{\log n/n}, (\log n/n)^{2/3})$ and $l := 6\sqrt{2}\log n/(nw)$ the number of intersections can be bounded by $O(n^{1/8}m^{3/4}\log^{7/8} n + n^{-1/6}m\log^{7/6} n)$. Adding up the bound for segments near the boundary and far away from the boundary yields a linear bound on the expected number of intersections. □

Inserting Points We now extend the analysis to the case where the triangulation changes during a tour because points are inserted. The points of the triangulation occur in two different roles in the analysis: They may contribute to the number of intersections as an endpoint of an intersecting edge but they may also block other edges because they lie in their exclusion region. The analysis can be extended by taking all points as possible endpoints but only the points of the original triangulation as blocking points.

For a fixed line segment of the tour the remaining points of the tour are not independent of this segment but their density can be bounded if we use a *bi-measure preserving* curve, which allows us to work on uniform distributions on the preimage and the image exchangeably [7]. The cost resulting from the fact that the starting point of a line segment is a vertex of the triangulation can be bounded by the update cost. In total this yields the following theorem:

Theorem 5 *Using a biased randomized insertion order and, in each round, walking along a Lipschitz-$1/2$, bi-measure preserving space-filling curve, the incremental construction algorithm runs in linear expected time for points distributed independently and uniformly in a bounded, convex area.*

4 Seeking a Conflict with the Neighbor

The main problem in the average case analysis of incremental constructions seems to be the boundary. Here we give an algorithm for which this is not so.

The points are inserted in random order. The algorithm maintains a dynamic bucketing scheme. This allows us to find the nearest neighbor in the triangulation for a new point in constant expected time using spiral search [2]. Now a d–simplex incident to the nearest neighbor is found which conflicts with this point. From this triangle all conflicting d–simplices are found as in the Bowyer-Watson algorithm.

Theorem 6 Seeking a Conflict with the Neighbor *constructs the Delaunay triangulation in linear expected time when the points are distributed independently and uniformly in a d–dimensional bounded convex open region for which the expected complexity of the Delaunay triangulation is linear. In particular, this is the case for the unit d–ball.*

Proof. The expected time required for searching the nearest neighbor and for updating the triangulation is linear [2]. It remains to bound the expected number of d–simplices of the triangulation containing the nearest neighbor of a new point. The difficulty is that the nearest neighbor is not a random point of the triangulation but a constant bound can be obtained by using that the in-degree of the nearest neighbor graph is bounded in any fixed dimension.

The special case of the d–ball follows directly from the linear expected complexity [6]. $\square$

5 Discussion

We presented two incremental construction algorithms for the Delaunay triangulation. The first algorithm constructs a spatial order of the points. Ideally, the Delaunay triangulation should be stored in the same order to make use of the locality of reference. A possible way to achieve this is presented by Blandford et al. [3]. For this, it is important to use one ordering for all points.

Two advantages of the first algorithm which we have not addressed in the analysis are its good performance on surface points and on large data sets. Furthermore, the algorithm runs in higher dimensions.

References

[1] N. Amenta, S. Choi, and G. Rote. Incremental constructions con BRIO. In *Proc. 19th Annual Symposium on Computational Geometry*, pages 211–219. ACM Press, 2003.

[2] J. L. Bentley, B. W. Weide, and A. C.-C. Yao. Optimal expected-time algorithms for closest point problems. *ACM Trans. Math. Softw.*, 6(4):563–580, 1980.

[3] D. K. Blandford, G. E. Blelloch, D. E. Cardoze, and C. Kadow. Compact representations of simplicial meshes in two and three dimensions. In *Proc. 12th Int. Meshing Roundtable*, 2003.

[4] L. Devroye, C. Lemaire, and J.-M. Moreau. Expected time analysis for Delaunay point location. *Comput. Geom. Theory Appl.*, 22:61–89, 2004.

[5] L. Devroye, E. Mücke, and B. Zhu. A note on point location in Delaunay triangulations of random points. *Algorithmica*, 22:477–482, 1998.

[6] R. A. Dwyer. Higher-dimensional Voronoi diagrams in linear expected time. *Discrete Comput. Geom.*, 6(4):343–367, 1991.

[7] J. Gao and J. M. Steele. General spacefilling curve heuristics and limit theory for the traveling salesman problem. *J. Complex.*, 10(2):230–245, 1994.

[8] J. Katajainen and M. Koppinen. Constructing Delaunay triangulations by merging buckets in quadtree order. *Fundam. Inform.*, 11:275–288, 1988.

[9] T. Ohya, M. Iri, and K. Murota. Improvements of the incremental method for the Voronoi diagram with a comparison of various algorithms. *J. Operations Res. Soc. Japan*, 27:306–337, 1984.

[10] L. K. Platzman and J. J. Bartholdi, III. Spacefilling curves and the planar travelling salesman problem. *J. ACM*, 36(4):719–737, 1989.

[11] H. Sagan. *Space-Filling Curves.* Springer, 1994.

[12] P. Su and R. L. S. Drysdale. A comparison of sequential Delaunay triangulation algorithms. *Comput. Geom. Theory Appl.*, 7(5-6):361–385, 1997.

Region Counting Graphs

Jean Cardinal[*] Sébastien Collette[†] Stefan Langerman[‡§]

Abstract

A new family of proximity graphs, called *region counting graphs (RCG)* is presented. The RCG for a finite set of points in the plane uses the notion of region counting distance introduced by Demaine et al. to characterize the proximity between two points p and q: the edge pq is in the RCG if and only if there is less than or exactly k vertices in a given geometric neighborhood defined by a region. These graphs generalize many common proximity graphs, such as k-nearest neighbor graphs, β-skeletons or Θ-graphs. This paper concentrates on RCGs that are invariant under translations, rotations and uniform scaling. For $k = 0$, we give conditions on regions R that define an RCG to ensure a number of properties including planarity, connectivity, triangle freeness, cycle freeness, bipartiteness, and bounded degree. These conditions take form of what we call *tight regions*: maximal or minimal regions that a region R must contain or be contained in to satisfy a given monotone property.

1 Introduction

We consider here *proximity graphs* [12], also called *neighborhood graphs*. Those graphs are defined on a finite set V of vertices in the plane and there exists an edge between any two vertices if they are "close" in some sense. The proximity can be measured for instance by the Euclidean distance between those vertices, the distance to other vertices of the graph, or the number of other vertices in a given neighborhood. Those graphs are well-studied and have numerous applications in computer graphics and classification; a survey of Jaromczyk and Toussaint [7] discusses many of them, such as Relative Neighborhood Graphs [7, 11], Gabriel Graphs [5, 10], β-skeletons [9], Rectangular Influence graph [6], and Θ-graphs [8, 14].

Previous work on proximity graphs traditionally consisted in the introduction of one or more graphs, followed by different contributions analyzing their properties. Surprisingly, the natural opposite approach does not seem to have been considered: to start from a set of desired graph properties to construct the definition of the proximity graph. For this, we have to define a class of proximity graphs general enough to encompass many useful graphs, but simple enough to be analyzed.

The simplest form of proximity graph is a *distance graph* that connects a point $p \in V$ to every point in V whose distance to p is at most some specified value D. Our class of graphs is a variant on this definition using the discrete *region counting distances* defined by Demaine, Iacono and Langerman [2]. These distance functions are parameterized by the finite point set V and the distance between two points is the count of items of V inside a region surrounding those points. In a *k-region counting graph* or *k-RCG* (respectively ($\leq k$)-RCG), two vertices are adjacent if and only if the region counting distance between them is equal to k (respectively at most k).

One of the motivations of our work was to design proximity graphs that are invariant under translations, rotations and uniform scaling. It can be shown that this property is satisfied if and only if the region defining the region counting distance between two points is obtained by translating, rotating and uniformly scaling a template region. In this paper, we concentrate on the case $k = 0$, where two vertices are adjacent if the region does not contain any other point of the set. We further focus on symmetric and convex regions and symmetric distances (undirected graphs).

The properties of those graphs are determined by the choice of the template region. More specifically, we say that a given template region satisfies a property when for all point sets V the graph generated using that region satisfies the property.

Graph properties that are monotone with respect to either edge removal or addition are good candidates for investigation, because monotone properties that are satisfied by a template region are satisfied by all template regions included in it in the case of edge addition, or containing it in the case of edge removal. This naturally raises the issue of template regions that are extremal with respect to the inclusion partial order. We show for instance that the lune, defined as the intersection of two disks of radius $||pq||$ and respective centers p and q, is the unique maximal region ensuring the connectivity of the graph. We call these extremal regions *tight regions*. However, because the inclusion relation is not a total order, tight regions need not be unique. Tight regions can somehow be seen as a deterministic geometric analogue to thresholds for monotone properties studied in random graph theory [4]. Table 1 summarizes our findings related to tight regions and their uniqueness for various graph properties.

[*] jcardin@ulb.ac.be
[†] Aspirant du F.N.R.S., sebastien.collette@ulb.ac.be
[‡] Chercheur qualifié du F.N.R.S., stefan.langerman@ulb.ac.be
[§] Computer Science Department, Université Libre de Bruxelles, CP212, Boulevard du Triomphe, 1050 Bruxelles, Belgium.

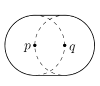
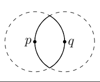
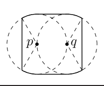
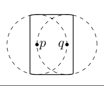
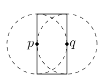
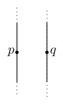
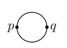
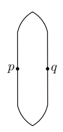
Region	Name	Property
$\mathbb{R}^2$	Plane	no edge $\star$ (unless $\lvert V\rvert < 3$)
	Mastercard	no chain $\star$, no cycle (Thm 10)
	Lune	no 3-cycle $\star$ (Thm 9), connected $\star$ (Thm 12)
	Pacman $P_{4\pi/3}$	no 3-star $\star$ (Thm 8: $P_{4\pi/k}$ for no k-star $\star$)
	Pacman $P_{6\pi/5}$	no 5-cycle (Thm 9)
	Pacman P_{π}	no 4-star $\star$ (Thm 8), no 4-cycle $\star$ (Thm 9)
	Slab	no cycle (Thm 10), bipartite (Thm 13)
	Ball	planar (Thm 11)
	Truncated Slab	no 5-cycle (Thm 9)

Table 1: Regions which are tight for various properties. Unique tight regions are marked with a $\star$.

In Section 2, we define the k- and $(\leq k)$-RCG and prove several facts, including how to combine tight regions for conjunction of properties. Section 3 is about geometric properties, which depend on the position of the vertices. We consider the planarity of the embedding and prove that no region counting graph invariant under translation, rotation and uniform scaling can guarantee a constant spanning ratio. This is interesting in light of known bounds on the spanning ratio of Θ-graphs [8], which are not rotation invariant. In section 4 we study the property of not having a given graph as subgraph, and how sets of tight regions can be constructed for forbidden combinations of graphs. Then we specifically consider the properties of not having a k-star or a k-cycle as subgraph. Finally, section 5 presents tightness results for planarity, cycle-freeness, connectivity and bipartiteness.

2 Region Counting Graphs

Definition 1 *An* influence region *R is a function mapping a pair (p, q) of points in $\mathbb{R}^2$ to a subset of $\mathbb{R}^2$ such that inclusion in $R(p, q)$ can be computed in $O(1)$ time.*

Definition 2 *An* anchored region *R is an influence region parameterized by a triple (a, b, D), where a and b are points in $\mathbb{R}^2$ and D is a subset of $\mathbb{R}^2$ such that inclusion in D can be computed in $O(1)$ time. The set $R(p, q)$ is the subset of $\mathbb{R}^2$ obtained by translating, rotating and uniformly scaling D so that a maps to p and b maps to q.*

Definition 3 *A* region counting distance *[2] $\mathrm{d}_R = \mathrm{d}_R^S(p, q)$ parameterized by a finite point set $S \subseteq \mathbb{R}^2$ and an influence region R, is defined by $\mathrm{d}_R(p, q) = \lvert(S \setminus \{p, q\}) \cap R(p, q)\rvert$.*

Definition 4 *A* symmetric region counting distance *is a region counting distance satisfying $\mathrm{d}_R(p, q) = \mathrm{d}_R(q, p)$.*

For the region counting distances using an anchored region as influence region, the symmetry of the region counting distance implies that the region $R(p, q)$ is symmetric with respect to the center of the line segment pq.

Definition 5 *A k-region counting graph $RCG_R^k(V) = (V, E)$ (respectively $(\leq k)$-region counting graph $RCG_R^{\leq k}(V)$) parameterized by an influence region R and an integer k is a graph where V is a finite subset of $\mathbb{R}^2$ and*

$$\forall p, q \in V : pq \in E \Leftrightarrow \mathrm{d}_R(p, q) = k \text{ (respectively } \leq k).$$

If the region counting distance is not symmetric, then the graph is defined as a directed graph, and as an undirected graph otherwise.

We denote by $RCG_R(V)$ the 0-RCG using the influence region R, which is the region counting graph where the edge pq exists if no other point is included in the region $R(p, q)$. Many previously known proximity graphs such as nearest neighbor graphs [3], β-skeletons [9] and Θ-graphs [8] can be defined as 0-RCG.

2.1 Assumptions

In what follows, we are mainly concerned with 0-RCG, and refer to them as region counting graphs or simply RCG. We restrict ourselves to using anchored regions parameterized by triples (a, b, D) where D is closed, convex and symmetric with respect to segment ab. Using only anchored regions is necessary and sufficient to guarantee the invariance of the graph structure under translation, rotation and uniform scaling of the set of points. We further restrict ourselves to symmetric region counting distances, hence undirected graphs.

The regions presented in Table 1 are of particular interest. The pacman $P_\Theta(p, q)$ is bounded by the convex hull of two pie-wedges of angle Θ, with apex in p and in q facing each other, such that $P_0(p, q)$ is the segment pq. The lune $L(p, q)$ is defined as $P_{2\pi/3}(p, q)$, while the mastercard $M(p, q)$ is $P_{2\pi}(p, q)$. The slab $S(p, q)$ is the infinite strip perpendicular to the line segment pq.

2.2 Properties

Lemma 1 *$\forall k, G = RCG_{R}^{\leq k}(V), G' = RCG_{R'}^{\leq k}(V)$: $R(p,q) \subseteq R'(p,q) \Rightarrow G' \subseteq G$.*

Definition 6 *A* graph property *$\mathcal{P}$ on a family of graphs $\mathcal{G}$ is a subset $\mathcal{P} \subseteq \mathcal{G}$. A* graph *$G$* has property *$\mathcal{P}$ if $G \in \mathcal{P}$.*

Definition 7 *A graph property $\mathcal{P}$ is* monotone with respect to edge addition (respectively to edge removal) *if and only if $\forall G = (V,E), G' = (V,E'), E \subseteq E'$ (respectively $E \supseteq E'$): $G \in \mathcal{P} \Rightarrow G' \in \mathcal{P}$.*

Our definition of monotonicity is slightly different from the one commonly used in graph theory. Usually, this is stated as follows: a property is monotone if and only if it is closed upon taking subgraphs. We add the symmetrical definition with properties monotone upon taking supergraphs.

Definition 8 *An anchored region R* satisfies *a graph property $\mathcal{P}$ if and only if for all $V \in \mathbb{R}^2$ finite, $RCG_R(V) \in \mathcal{P}$.*

Definition 9 *An anchored region R* is tight *for a graph property $\mathcal{P}$ monotone with respect to edge addition (respectively to edge removal) if and only if R satisfies $\mathcal{P}$ and for all anchored region $R' \supset R$ (respectively $R' \subset R$), R' does not satisfy $\mathcal{P}$.*

Note that the monotonicity of the property with respect to edge removal implies that any region containing a tight region as subset satisfies the property as well. On the other hand, for regions that are strictly contained in a tight region, one can always find a set of points generating a graph that does not have the property. A similar observation holds the other way around for properties that are monotone with respect to edge addition. Another definition of a tight region for a monotone property is a region that satisfies the property and is extremal for the inclusion partial order.

Knowing tight regions for useful properties is important in practice, because it allows to check quickly if the properties we wish to obtain are satisfied or not. The uniqueness of a tight region is even more important, because knowing a single tight region R does not, in general, give any information on the properties guaranteed by regions that simultaneously do not contain R and are not contained in R. If the tight region R is unique, any region that does not contain R does not satisfy the property, even if it is not strictly included in R.

Lemma 2 *Let $\mathcal{P}$ be a monotone property with respect to edge removal and R be the unique tight region satisfying that property. Every region $R' \not\supseteq R$ does not satisfy $\mathcal{P}$.*

The same lemma holds for edge addition, where every region $R' \not\subseteq R$ does not satisfy $\mathcal{P}$.

Now given a set of compatible properties we wish to have on the graph, we can easily construct a region guaranteeing these properties. In the case of convex and symmetric regions and properties that are monotone with respect to edge removal, this is achieved by taking the convex hull of the union of the tight regions for each property. In some cases, this region can be proved to be extremal with respect to the inclusion ordering among the considered type of regions.

Lemma 3 *Let $\mathcal{P}$ and $\mathcal{P}'$ be two monotone properties with respect to edge removal, and R and R' two regions satisfying $\mathcal{P}$ and $\mathcal{P}'$ respectively. Then $R \cup R'$ satisfies $\mathcal{P} \cap \mathcal{P}'$. Furthermore, if R and R' are the unique tight regions for $\mathcal{P}$ and $\mathcal{P}'$, then the convex hull of $R \cup R'$ is tight and unique for $\mathcal{P} \cap \mathcal{P}'$.*

A similar Lemma holds for properties that are monotone with respect to edge addition as well.

In the class of all possible properties, we can identify various families. For instance the class of properties corresponding to graphs not containing any of the graphs in a given set as subgraph. Another way used to describe properties is to express them as a set of forbidden minors.

Our study showed that there is not always a unique tight region satisfying a property expressed as a set of forbidden subgraphs, or as a set of forbidden minors. However, we do not know whether every monotone property can be expressed as a *finite* set of symmetric, convex and closed tight regions.

3 Geometric Properties

Here we consider geometric properties, which are properties depending on the position of the vertices.

The following theorem shows that the graph embedding obtained by linking adjacent points by straight line segments is planar if and only if the region contains the ball of diameter pq.

Theorem 4 *The ball B is the unique tight region ensuring a planar embedding.*

In the following, $\mathrm{d}_G(u,v)$ is the minimum Euclidean length of a path between u and v.

Definition 10 *A graph G in the plane is a t*-spanner *for $t \in [1,\infty)$, if and only if $\forall u,v \in V$: $\mathrm{d}_G(u,v)/||uv|| \leq t$, where t is called the spanning ratio.*

The spanning ratio is also called the dilation. We say that the spanning ratio is unbounded whenever it cannot be bounded by a constant independent of n, i.e. it can be made arbitrarily large for sufficiently large n.

Theorem 5 *For every convex anchored region R with non-empty interior, there exists a real number $\alpha > 0$ such that we can find a set S of n vertices for any n for which the spanning ratio of $RCG_R(S)$ is $\Omega(n^\alpha)$.*

The proof uses recent results on the spanning ratio of β-skeletons [1, 13]. When we introduced the region counting distances, one of our motivations was to find an anchored region such that the corresponding graph would be a t-spanner not affected by rotations of the set of points. The well-known Θ-graph, which is not invariant to rotations, exhibits a constant spanning ratio. The theorem above shows that it is not possible to find an anchored region corresponding to a constant spanning ratio other than the segment or the empty region if the anchored region is convex.

4 Forbidden Subgraphs

A property $\mathcal{P}$ defined by a forbidden subgraph F is a set of graphs not having any subgraph isomorphic to F. We denote by $F \subseteq F'$ the fact that F' has a subgraph isomorphic to F. The union $F \cup F'$ of two graphs $F = (V, E)$ and $F' = (V', E')$ with $V \cap V' = \emptyset$ is $(V \cup V', E \cup E')$.

Lemma 6 *If the region R is tight for a forbidden subgraph F and for a forbidden subgraph $F' \supseteq F$, then it is tight for all forbidden subgraph G which satisfies $F \subseteq G \subseteq F'$.*

Theorem 7 *If $\mathcal{R}$ is the set of tight regions for a forbidden subgraph F and if $\mathcal{R}'$ is the set of tight regions for a forbidden subgraph F', then the set of tight regions for the forbidden subgraph $F \cup F'$ is $\{R \in \mathcal{R} \cup \mathcal{R}' | \forall R' \in \mathcal{R} \cup \mathcal{R}' : R \not\supseteq R'\}$.*

We will now study properties which can be explained as forbidden subgraphs.

Theorem 8 *The pacman $P_{4\pi/k}$ is the unique tight region forbidding a k-Star, which is equivalent to bounding the maximum degree by $k - 1$.*

There are many regions corresponding forbidding a k-cycle, depending on the parameter k. The tight regions for 3-cycle and 4-cycle are unique, while there are at least two regions for $k \geq 5$.

Theorem 9
1. *The Lune L is the unique tight region forbidding a 3-cycle.*
2. *The pacman P_π is the unique tight region forbidding a 4-cycle.*
3. *There are at least two tight regions forbidding a 5-cycle: the truncated slab and $P_{6\pi/5}(p, q)$.*

We show in the next sections that cycle freeness, which corresponds to forbidding a k-cycle for every k, has also two tight regions, and that the tight regions for the 5-cycle are subregions of those for cycle freeness.

5 Other Properties

Properties discussed in theorems 10 and 11 correspond to forbidding graph minors: cycle freeness corresponds to forbidding a triangle as minor; Kuratowski's theorem says that planarity corresponds to forbidden minors $K_{3,3}$ and K_5.

Theorem 10 *There are at least two tight regions for cycle freeness: the slab S and the mastercard M.*

Theorem 11 *The ball B is a tight region for planarity.*

Theorem 12 *The lune L is the unique tight region for connectivity.*

Theorem 13 *There are at least two tight regions for bipartiteness: the slab S and the mastercard M.*

References

[1] P. Bose, L. Devroye, W. Evans, and D. Kirkpatrick. On the spanning ratio of gabriel graphs and beta-skeleton. *Siam Journal of Discrete Math*, 2004. accepted.

[2] E. D. Demaine, J. Iacono, and S. Langerman. Proximate point searching. In *Proceedings of the 14th Canadian Conference on Computational Geometry (CCCG)*, 2002.

[3] D. Eppstein, M. Paterson, and F. Yao. On nearest-neighbor graphs. *Discrete and Computational Geometry*, 17:263–282, 1997.

[4] E. Friedgut and G. Kalai. Every monotone graph property has a sharp threshold. In *Proceedings of the AMS*, pages 2993–3002, 1996.

[5] K. Gabriel and R. Sokal. A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18:259–278, 1969.

[6] M. Ichino and J. Sklansky. The relative neighborhood graph for mixed feature variables. *Pattern Recognition*, 18:161–167, 1985.

[7] J. Jaromczyk and G. Toussaint. Relative neighborhood graphs and their relatives. *Proceedings of the IEEE*, 80(9):1502–1571, 1992.

[8] J. Keil and C. Gutwin. Classes of graphs which approximate the complete euclidean graph. *Discrete and Computational Geometry*, 7(1):13–28, 1992.

[9] D. Kirkpatrick and J. Radke. A framework for computational morphology. *Computational Geometry*, pages 217–248, 1985.

[10] T. Su and R. Chang. The k-gabriel graphs and their applications. In *Proceedings of the International Symposium SIGAL'90*, pages 66–75, 1990.

[11] T. Su and R. Chang. Computing the k-relative neighborhood graphs in euclidean plane. *Pattern Recognition*, 24:231–239, 1991.

[12] G. Toussaint. Some unsolved problems on proximity graphs. In *Proceedings of the First Workshop on Proximity Graphs*, 1991.

[13] W. Wang, X. Li, K. Moaveninejad, Y. Wang, and W. Song. The spanning ratio of beta-skeletons. In *Proceedings of the Canadian Conference on Computational Geometry (CCCG)*, 2003.

[14] A. Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM Journal on Computing*, 11(4):721–736, 1982.

Homotopic Spanners*

Sergio Cabello[†] Bojan Mohar[‡] Arjana Žitnik[§]

Abstract

We introduce the concept of homotopic spanners in the plane with obstacles and show lower bounds on the number of edges that they require. We also provide a construction based on Θ-graphs for constructing homotopic spanners.

1 Introduction

Spanners have become a basic tool for the design of networks: they are graphs connecting a given set of sites with the property that the distances between sites along the graph is similar to the straight-line distance between the sites. As a basic requirement, spanners have to be sparse, that is, they need to have few edges. Typically, we are interested on spanners that have additional properties, such as bounded degree, small total length, small spanning diameter, etc.

In applications like robot motion planning, we often deal with the scenario where the sites are in the plane and we also have a set of obstacles to be avoided. This naturally leads to the problem of computing spanners under the influence of polyhedral obstacles, already considered by Clarkson [6] and Das [7].

We consider here the construction of spanners in the plane with point-obstacles, but with the additional condition that between each pair of sites there is a short path in the spanner which is homotopically equivalent to the straight-line segment that joins the sites. Although much work has been done on spanners with additional properties, we are not aware of any research on constructing spanners with topological properties.

In the next section we introduce the basic notation and topological background; we also define precisely the concept of homotopic spanners. In Section 3 we show a modification of Θ-graphs that can be used to construct homotopic spanners. In Section 4 we discuss the computational issues related to the construction. In Section 5 we present lower bounds on the number of edges that any homotopic spanner needs.

2 Notions and problem statement

Topological background. A finite set of points $\mathcal{K} \subset \mathbb{R}^2$ will be called *point-obstacles*. If $x, y \in \mathbb{R}^2 \setminus \mathcal{K}$, a *path* from x to y is a continuous mapping $\alpha : [0,1] \to \mathbb{R}^2 \setminus \mathcal{K}$ such that $\alpha(0) = x$ and $\alpha(1) = y$. If β is a path from y to z, then the concatenation $\alpha + \beta$ of paths α and β is a path from x to z defined as $(\alpha+\beta)(u) = \alpha(2u)$ if $0 \le u \le \frac{1}{2}$ and $(\alpha+\beta)(u) = \beta(2u-1)$ if $\frac{1}{2} \le u \le 1$. Two paths α, β joining the same pair of points in $\mathbb{R}^2 \setminus \mathcal{K}$ are said to be *homotopic*, denoted $\alpha \sim_{\mathbb{R}^2\setminus\mathcal{K}} \beta$ if the loop $\alpha - \beta$ (α concatenated with the reverse of β) is a contractible curve in $\mathbb{R}^2 \setminus \mathcal{K}$. The reader is referred to [8], where also the following standard results can be found:

Lemma 1 *Homotopy of paths has the following properties:*

1. *The relation $\alpha \sim_{\mathbb{R}^2\setminus\mathcal{K}} \beta$ is an equivalence relation.*
2. *If $\alpha \sim_{\mathbb{R}^2\setminus\mathcal{K}} \beta$, $\alpha' \sim_{\mathbb{R}^2\setminus\mathcal{K}} \beta'$, and $\alpha(1) = \beta(1) = \alpha'(0) = \beta'(0)$, then $(\alpha+\alpha') \sim_{\mathbb{R}^2\setminus\mathcal{K}} (\beta+\beta')$.*
3. *If the paths α, β share endpoints and are contained in a convex subset of $\mathbb{R}^2\setminus\mathcal{K}$, then $\alpha \sim_{\mathbb{R}^2\setminus\mathcal{K}} \beta$.*

Homotopic Spanners. Let S be a point set in $\mathbb{R}^2$, and let $G = (S, E)$ be a graph on S. The graph is represented in the plane with each vertex represented by the point itself and with straight-line edges. We use ss' to denote both, the edge of G and the straight-line segment joining s and s'. We associate with each edge $ss' \in E$ the length $|ss'|$ of the straight-line segment joining its vertices. The length of a path α in G is the sum of the lengths of its edges; we denote it by $|\alpha|_G$.

For $t \in \mathbb{R}, t \ge 1$, a path in G from $s \in S$ to $s' \in S$ is a *t-path* if its length, is at most $t\,|ss'|$. A graph G is a *t-spanner* if, for each pair of points $s, s' \in S$, there exists a t-path in G from s to s'. We consider the following generalization.

Definition 1 *Given a set of points $S \subset \mathbb{R}^2$ and a set of point-obstacles $\mathcal{K}$, a $\mathcal{K}$-*homotopic t-spanner *of S is a graph $G = (S, E)$ such that, for any $s, s' \in S$, there is a t-path α in G such that α and the segment ss' are homotopic in $\mathbb{R}^2 \setminus \mathcal{K}$.*

*S.C. partially supported by the European Community Sixth Framework Programme under a Marie Curie Intra-European Fellowship. B.M. partially supported by grant L1-5014-0101-04.

[†]Department of Mathematics, IMFM, Ljubljana, Slovenia. sergio.cabello@imfm.uni-lj.si

[‡]Department of Mathematics, FMF, University of Ljubljana, Slovenia. bojan.mohar@fmf.uni-lj.si

[§]Department of Mathematics, FMF, University of Ljubljana, Slovenia. arjana.zitnik@fmf.uni-lj.si

We consider the following problem: given a fixed number $t > 1$, construct homotopic t-spanners as a function of S and $\mathcal{K}$ such that the number of edges is the spanner is not too large. We let $n = |S|$ and $k = |\mathcal{K}|$. We assume that no obstacle in $\mathcal{K}$ is aligned with two points of S, as otherwise it may be that the desired spanner does not exist.

3 Construction of homotopic spanners

The idea is to modify the construction of Θ-spanners introduced by Keil and Gutwin [9]. We use a notation similar to Arya, Mount, and Smid [2] and Bose, Gudmundsson, and Morin [3]. Consider an angle $\theta = \frac{2\pi}{T}$ for some integer $T > 8$ such that it holds $t_\theta = \frac{1}{\cos\theta - \sin\theta} \leq t$. For a point s in S, consider the set of rays $\mathcal{R}_{s,\theta} = \{ray_j(s) \mid j \in \{0, \ldots, T-1\}\}$, where $ray_j(s)$ is the straight ray from s with angle $j\theta$ with a horizontal line, and $\mathcal{R}_{s,\mathcal{K}} = \{ray(s,o) \mid o \in \mathcal{K}\}$, where $ray(s,o)$ is the straight ray starting at s with direction towards o. Let $\mathcal{R}_s = \mathcal{R}_{s,\theta} \cup \mathcal{R}_{s,\mathcal{K}}$.

All the rays in $\mathcal{R}_s$ have s as starting point, and therefore they divide the plane into a set of cones, which we denote by $\mathcal{C}_s$. Since t is a fixed constant, also T is a constant. Hence, $\mathcal{C}_s$ consists of $O(1+k)$ cones. Any cone $C \in \mathcal{C}_s$ has angle at most θ and it contains no obstacles in its interior. For a cone $C \in \mathcal{C}_s$, consider any ray r from s contained in C and let j be the largest value such that $j\theta$ is smaller than the angle of r; we use $ray(C)$ for the ray $ray_j(s) \in \mathcal{R}_{s,\theta}$. Observe that the angle between r and $ray(C)$ is at most θ.

Let the graph $\Theta(S, \mathcal{K}, T)$ be defined as follows:

- The set of vertices of $\Theta(S, \mathcal{K}, T)$ is S;
- For each point $s \in S$, for each cone $C \in \mathcal{C}_s$ such that $C \cap (S \setminus \{s\}) \neq \emptyset$, we put an edge connecting s and a point s_C in $C \cap S \setminus \{s\}$ that has the orthogonal projection onto $ray(C)$ closest to s. If there are more than one candidate for s_C, we select one which is closest to $ray(C)$.

Observe that $\Theta(S, \mathcal{K}, T)$ has $O(nk)$ edges.

Theorem 2 *The graph $\Theta(S, \mathcal{K}, T)$ is a $\mathcal{K}$-homotopic t_θ-spanner of S with $O(nk)$ edges.*

Proof. Consider two points $s, s' \in S$, and let C be the cone of $\mathcal{C}_s$ that contains s'. By construction, we know that there is a point $s_c \in C$ such that ss_c is an edge in $\Theta(S, \mathcal{K}, T)$. Using that $ray(s, s')$ and $ray(s, s_c)$ form an angle at most θ, the same argument that is used for the standard Θ-graph [2] implies

$$t_\theta\,|s_cs'| \leq t_\theta\,|ss'| - |ss_c|. \tag{1}$$

We show by induction on the rank of the interpoint distances that for any pair of points $s, s' \in S$ there is a t_θ-path in $\Theta(S, \mathcal{K}, T)$ that is homotopic to ss'. If the pair s, s' is a closest pair, then it holds that $s_c = s'$ and therefore the segment ss' is in $\Theta(S, \mathcal{K}, T)$.

Consider a pair of points $s, s' \in S$. If $s' = s_c$, then the segment ss' is in $\Theta(S, \mathcal{K}, T)$ and there is nothing to show. Otherwise, $s' \neq s_c$. Because of (1), we have $|s_cs'| < |ss'|$, and by induction hypothesis there is a t_θ-path α in $\Theta(S, \mathcal{K}, T)$ from s_c to s' that is homotopic to the segment s_cs' in $\mathbb{R}^2 \setminus \mathcal{K}$, that is $\alpha \sim_{\mathbb{R}^2\setminus\mathcal{K}} s_cs'$. Let $\beta = ss_c + \alpha$. We have

$$|\beta|_G = |ss_c| + |\alpha|_G \leq |ss_c| + t_\theta\,|s_cs'| \leq t_\theta\,|ss'|,$$

where the last inequality follows from equation (1). This means that β is a t_θ-path from s to s'.

We next show that $\beta \sim_{\mathbb{R}^2\setminus\mathcal{K}} ss'$, which finishes the proof. Since $\beta = ss_c + \alpha$ and $\alpha \sim_{\mathbb{R}^2\setminus\mathcal{K}} s_cs$, we have $\beta \sim_{\mathbb{R}^2\setminus\mathcal{K}} ss_c + s_cs$ because of property 2 in Lemma 1. Because the triangle $\triangle ss's_c$ is contained in the cone $C \in \mathcal{C}_s$ we have $\mathcal{K} \cap \triangle ss's_c = \emptyset$, and by property 3 in Lemma 1 we conclude that $ss_c + s_cs' \sim_{\mathbb{R}^2\setminus\mathcal{K}} ss'$. Since $\sim$ is an equivalence relation we get $\beta \sim_{\mathbb{R}^2\setminus\mathcal{K}} ss'$. $\square$

For any value $t > 1$ we can take a constant $T \in \mathbb{N}$ large enough such that $t \geq \frac{1}{\cos(2\pi/T) - \sin(2\pi/T)}$, and we conclude that for any fixed t we can construct a $\mathcal{K}$-homotopic t-spanner with $O(nk)$ edges.

4 Efficient construction

Consider a set of n sites S and k obstacles $\mathcal{K}$. We assume that $k \leq n$, as otherwise we can just consider the complete graph as a spanner and we are within the bound of $O(nk)$ edges for a spanner that we are aiming to. For a fixed value T, the graph $\Theta(S, \mathcal{K}, T)$ can be constructed in $O(n^2 \log k)$ time as follows:

1. for each site $s \in S$
   (a) split the sites $S \setminus \{s\}$ into the cones of $\mathcal{C}_s$. This can be done by making a tree-like structure for the boundary rays $\mathcal{R}_s$ of $\mathcal{C}_s$ in $O(k \log k)$ and locating each point of $S \setminus \{s\}$ in the appropriate cone in $O(\log k)$ time per point. This takes $O(k \log k + n \log k) = O(n \log k)$ time.
   (b) for each cone $C \in \mathcal{C}_s$, scan the points and choose the one that s gets connected to, according to the criteria in Section 3. This takes $O(n)$ time overall because each point appears at most in two cones of $\mathcal{C}_s$.

We discuss how the graph $\Theta(S, \mathcal{K}, T)$ can be constructed in a more efficient way. The idea is to consider all the cones as range spaces and use the standard trade-offs for simplex range queries; see Matoušek [10] or the survey by Agarwal and Erickson [1]. The main result to be used is the following (we use the notation $\tilde{O}(f(n)) = O(f(n)n^\varepsilon)$ for any $\varepsilon > 0$, where the constant in $\tilde{O}(f(n))$ may depend on ε).

Lemma 3 *For any set S of n points in the plane and any value $n \leq m \leq n^2$ there is a family $\mathcal{F}(S) = \{F_1, \dots, F_p\}$ of subsets of S and a data structure $\mathcal{D}(S)$ such that*

- *$p = O(m)$, that is, $\mathcal{F}(S)$ has $O(m)$ members;*
- *$\sum_{i=1}^{p} |F_i| = \tilde{O}(m)$;*
- *for any triangle Δ in the plane, there is a group $\mathcal{F}(\Delta)$ of $\tilde{O}(n/\sqrt{m})$ elements of $\mathcal{F}(S)$ such that $\Delta \cap S = \bigcup_{F \in \mathcal{F}(\Delta)} F$;*
- *$\mathcal{D}(S)$ has size $O(m)$ and can be constructed in $\tilde{O}(m)$ time;*
- *for a query triangle Δ, the data structure $\mathcal{D}$ provides $\mathcal{F}(\Delta) \subset \mathcal{F}$ in $\tilde{O}(n/\sqrt{m})$ time.*

For a point set S and an angle α let $Point(S, \alpha)$ denote a point in S such that the line passing through it with angle $\alpha + \pi/2$ has all the points of S to its right; that is, $Point(S, \alpha)$ is a point with minimum x-coordinate after rotating S with angle $-\alpha$.

We extend the data structure of the previous lemma as follows: for each set $F \in \mathcal{F}$ and each value $j = 0, \dots, T-1$ we store $Point(F, j\frac{2\pi}{T})$. For any triangle Δ we have

$$Point(S \cap \Delta, j\tfrac{2\pi}{T}) \in \{Point(F, j\tfrac{2\pi}{T}) \mid F \in \mathcal{F}(\Delta)\},$$

and using the previous lemma we conclude that we can find the point $Point(S \cap \Delta, j\frac{2\pi}{T})$ in $\tilde{O}(n/\sqrt{m})$ time per triangle Δ.

The augmented data structure can be constructed by considering each $j = 0, \dots, T-1$ and scanning each $F \in \mathcal{F}$. Since we regard T as a constant, and each $F \in \mathcal{F}$ is considered T times, we need $O(T \sum_{F \in \mathcal{F}} |F|) = \tilde{O}(m)$ time to construct the augmented data structure.

Consider the construction of $\Theta(S, \mathcal{K}, T)$ given in Section 3. For a cone C with apex s, we have to find a point in C with the orthogonal projection onto $ray(C)$ closest to s. Since $ray(C)$ has an angle of the form $j_C \frac{2\pi}{T}$ for some j_C, it follows that this point is $Point(C \cap S, j_C \frac{2\pi}{T})$. Since a cone is a special case of a triangle, we can use the previous discussion to conclude that we can find the edge that the cone C contributes to $\Theta(S, \mathcal{K}, T)$ in $\tilde{O}(n/\sqrt{m})$ time.

By setting $m = n^{4/3}k^{2/3}$ we can find the edge corresponding to a cone in $\tilde{O}(n/\sqrt{n^{4/3}k^{2/3}}) = \tilde{O}(n^{1/3}k^{-1/3})$ time. This makes sense since we are assuming $k \leq n$. The preprocessing of the data structure for this case takes $\tilde{O}(n^{4/3}k^{2/3})$ time. Since we have to consider $O(nk)$ cones for the construction of $\Theta(S, \mathcal{K}, T)$, we can find all the edges in time $O(nk) \cdot \tilde{O}(n^{1/3}k^{-1/3}) = \tilde{O}(n^{4/3}k^{2/3})$ time. We summarize.

Theorem 4 *If $k \leq n$, we can construct $\Theta(S, \mathcal{K}, T)$ in $O(n^{4/3+\varepsilon}k^{2/3})$ time, for any fixed $\varepsilon > 0$.*

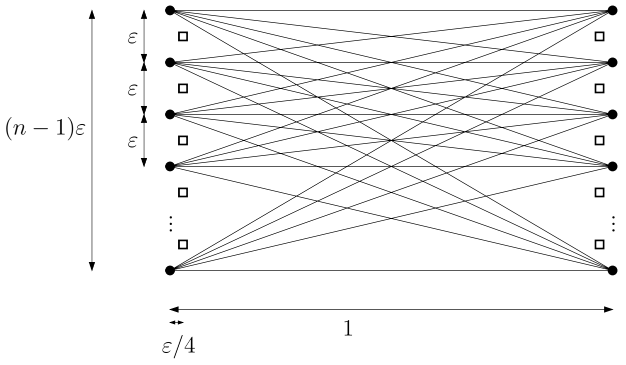


Figure 1: Lower bound for homotopic spanners when $k = \Theta(n)$. The dots are sites and the squares are obstacles.

This result improves the $O(n^2 \log k)$ time construction given above whenever $k = O(n^{1-\varepsilon})$ for any fixed $\varepsilon > 0$.

5 Lower bounds

The homotopic spanner that we have constructed above has $O(nk)$ edges, where n is the number of points and k is the number of point-obstacles. In contrast, the standard spanners have only $O(n)$ edges. It is natural to wonder if $\Omega(nk)$ edges are indeed necessary for constructing a homotopic spanner. We have the following construction for the case $k = \Theta(n)$.

Lemma 5 *For any value of t, $1 < t < 3$, and any value of n, there is a set S of $O(n)$ points and a set $\mathcal{K}$ of $O(n)$ point obstacles such that any $\mathcal{K}$-homotopic t-spanner of S needs $\Omega(n^2)$ edges.*

Proof. Take $\varepsilon = \frac{3-t}{3n}$ and consider the configuration in Figure 1; we have sites

$$L = \{(0, j\varepsilon) \mid j \in [n]\}, \quad R = \{(1, j\varepsilon) \mid j \in [n]\},$$

and obstacles

$$\mathcal{K}_L = \{(\tfrac{\varepsilon}{4}, \tfrac{1}{2} + j\varepsilon) \mid j \in [n-1]\}, \tag{2}$$
$$\mathcal{K}_R = \{(1 - \tfrac{\varepsilon}{4}, \tfrac{1}{2} + j\varepsilon) \mid j \in [n-1]\}, \tag{3}$$

where we use the notation $[n] = \{1, \dots, n\}$. This configuration has $2n$ points and $2n-2$ obstacles. It remains to argue that any homotopic t-spanner has $\Omega(n^2)$ edges.

The key observation is that any homotopic t-path from a site $l \in L$ to a site $r \in R$ has to use the segment lr. Note that if a path α from l to r is homotopic to lr in $\mathbb{R}^2 \setminus (\mathcal{K}_L \cup \mathcal{K}_R)$ and only "crosses" from L to R once, then the segment lr has to be part of α.

Assume for the contrary that there is a $(\mathcal{K}_L \cup \mathcal{K}_R)$-homotopic t-spanner G of $L \cup R$ that does not contain the edge lr for some $l \in L, r \in R$. Let α be the

t-path in G from l to r that is homotopic to lr in $\mathbb{R}^2 \setminus (\mathcal{K}_L \cup \mathcal{K}_R)$. Since α does not contain the segment lr, it has to "cross" from L to R (or vice versa) at least three times. We conclude that $|\alpha|_G \geq 3$. Using that $|lr| \leq 1 + n\varepsilon$ it follows that $|\alpha|_G/|lr| > t$, and G cannot be a $(\mathcal{K}_L \cup \mathcal{K}_R)$-homotopic t-spanner.

Since each segment from L to R has to be in a homotopic t-spanner, then any homotopic t-spanner has at least $n^2 = \Omega(n^2)$ edges. □

The above construction for the lower bound generalizes for general values of n, k as $\Omega(n + \min\{n^2, k^2\})$. Given n and k, if $k \geq n$ then we can take the construction of the previous result and add $k - n$ extra obstacles; we need $\Omega(n^2) = \Omega(n + \min\{n^2, k^2\})$ edges in any homotopic t-spanner. If $k < n$ then take the construction above with $n = k$ and add the extra $n - k$ sites far enough not to influence the construction; we need $\Omega(k^2)$ edges to make a t-spanner of the first part, and we need $n - k - 1$ edges to connect all the sites added afterwards, which adds to $\Omega(k^2 + n - k) = \Omega(k^2 + n) = \Omega(n + \min\{n^2, k^2\})$ because $k < n$. We summarize:

Theorem 6 *For any value t, $1 < t < 3$, and any values of n, k, there is a set S of $O(n)$ points and a set $\mathcal{K}$ of $O(k)$ point obstacles such that any $\mathcal{K}$-homotopic t-spanner of S needs $\Omega(n + \min\{n^2, k^2\})$ edges.*

6 Discussion

We have introduced the concept of homotopic spanners in the plane with point-obstacles. It is not clear how this concept generalizes to higher dimensions, where all paths are homotopic with respect to point-obstacles, neither how it generalizes to polyhedral obstacles, where a straight-line segment connecting two sites may intersect obstacles.

For n sites and k point-obstacles, we have presented a construction for homotopic spanners that uses $O(nk)$ edges. However, we can only provide an example showing that a homotopic spanner may need $\Omega(n + \min\{n^2, k^2\})$ edges. Our construction is based on Θ-graphs. The most natural alternative to consider is the Well Separated Pairs Decomposition of Callahan and Kosaraju [5, 4], but it does not seems easy to handle the homotopy classes induced by the obstacles in a better way than with Θ-graphs.

As with normal spanners, we can also be interested on homotopic spanners with additional properties, such as small maximum degree, small spanner diameter, small total weight, etc. As for the maximum degree D, the construction given above shows that in the worst case $D = \Omega(k)$, and so we cannot aim to get bounded degree. Adapting the ordered Θ-spanners of Bose, Gudmundsson, and Morin [3] to handle point-obstacles, it is possible to construct spanners with $O(nk)$ edges and maximum degree $O(k \log n)$. As for the spanner diameter, a randomized construction similar to Arya, Mount, and Smid [2], where we keep all the obstacles at each stage, will lead to randomized algorithms for constructing homotopic spanners with $O(nk)$ edges and $O(\log n)$ spanner diameter.

References

[1] P. Agarwal and J. Erickson. Geometric range searching and its relatives. In B. Chazelle, J. E. Goodman, and R. Pollack, editors, *Advances in Discrete and Computational Geometry*, volume 223 of *Contemporary Mathematics*, pages 1–56. Ame. Math. Soc., Providence, RI, 1999.

[2] S. Arya, D. Mount, and M. Smid. Dynamic algorithms for geometric spanners of small diameter: Randomized solutions. *Comput. Geom.: Theory and Applications*, 13:91–107, 1999.

[3] P. Bose, J. Gudmundsson, and P. Morin. Ordered theta graphs. *Comput. Geom.: Theory and Applications*, 28:11–18, 2004.

[4] P. Callahan and S. Kosaraju. Faster algorithms for some geometric graph problems in higher dimensions. In *Proc. 4th ACM-SIAM Sympos. Discrete Algorithms*, pages 291–300, 1993.

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

[6] K. Clarkson. Approximation algorithms for shortest path motion planning. In *Proc. 19th Ann. ACM Symp. Theory Comput.*, pages 56–65, 1987.

[7] G. Das. The visibility graph contains a bounded-degree spanner. In *Proc. 9th Canad. Conf. Comput. Geom.*, pages 70–75, 1997.

[8] A. Hatcher. *Algebraic Topology*. Cambridge University Press, 2001. Available at `http://www.math.cornell.edu/~hatcher/`.

[9] J. Keil and C. Gutwin. Classes of graphs which approximate the complete Euclidean graph. *Discrete Comput. Geom.*, 7:13–28, 1992.

[10] J. Matoušek. Range searching with efficient hierarchical cuttings. *Discrete Comput. Geom.*, 10:157–182, 1993.

Finding the Best Shortcut in a Geometric Network [*]

Mohammad Farshi[†] Panos Giannopoulos[‡] Joachim Gudmundsson[§]

Abstract

Given a Euclidean graph G in $\mathbb{R}^d$ with n vertices and m edges we consider the problem of adding a shortcut such that the stretch factor of the resulting graph is minimized. Currently, the fastest algorithm for computing the stretch factor of a Euclidean graph runs in $\mathcal{O}(mn + n^2 \log n)$ time, resulting in a trivial $\mathcal{O}(mn^3 + n^4 \log n)$ time algorithm for computing the optimal shortcut. First, we show that a simple modification yields the optimal solution in $\mathcal{O}(n^4)$ time using $\mathcal{O}(n^2)$ space. To reduce the running times we consider several approximation algorithms. Our main result is a $(2 + \varepsilon)$-approximation algorithm with running time $\mathcal{O}(nm + n^2(\log n + 1/\varepsilon^{3d}))$ using $\mathcal{O}(n^2)$ space.

1 Introduction

Consider a set V of n points in $\mathbb{R}^d$. A network on V can be modeled as an undirected graph G with vertex set V and an edge set E of size m where every edge $e = (u, v)$ has a weight $wt(e)$. A Euclidean network is a geometric network where the weight of the edge $e = (u, v)$ is equal to the Euclidean distance between its two endpoints u and v. Let $t > 1$ be a real number. We say that G is a *t-spanner* for V, if for each pair of points $u, v \in V$, there exists a path in G of weight at most t times the Euclidean distance between u and v. The minimum t such that G is a t-spanner for V is called the stretch factor, or dilation, of G.

Complete graphs represent ideal communication networks, but they are expensive to build; sparse spanners represent low-cost alternatives. The weight of the spanner network is a measure of its sparseness; other sparseness measures include the number of edges, the maximum degree, and the number of Steiner points. Spanners for complete Euclidean graphs as well as for arbitrary weighted graphs find applications in robotics, network topology design, distributed systems, design of parallel machines, and many other areas and have been a subject of considerable research. Recently spanners found interesting practical applications in areas such as metric space searching [6] and broadcasting in communication networks [1]. The problem of constructing spanners has received considerable attention from a theoretical perspective, see the surveys [3, 7].

Apx. factor	Time complexity	Space	Sec.
1	$\mathcal{O}(n^3 m + n^4 \log n)$	$\mathcal{O}(n)$	2
1	$\mathcal{O}(n^4)$	$\mathcal{O}(n^2)$	2
3	$\mathcal{O}(nm + n^2 \log n)$	$\mathcal{O}(n)$	3
$2 + \varepsilon$	$\mathcal{O}(nm + n^2(\log n + 1/\varepsilon^{3d}))$	$\mathcal{O}(n^2)$	4

Table 1: Complexity bounds for the algorithms presented in the paper.

Most known algorithms either construct a spanner given a point set or prunes a given graph, but in many applications the geometric network is already given, and the problem at hand is to extend the network with an additional edge, or edges, while minimizing the stretch factor of the resulting graph. Surprisingly this problem has not been studied previously, to the best of the authors' knowledge. In this paper we study the following problem:

Problem. Given a Euclidean graph G construct a graph G' by adding an edge to G such that the stretch factor of G' is minimized.

We present one exact algorithm and several approximation algorithms. The results presented in this paper are summarized in Table 1.

We will denote by $|uv|$ the Euclidean distance between u and v, and $\delta_G(u, v)$ denotes the shortest path between u and v in G with length $d_G(u, v)$. Finally, $G_{\mathcal{P}}$ will denote the optimal solution, while $t_{\mathcal{P}}$ and t denotes the stretch factor of $G_{\mathcal{P}}$ and G respectively.

2 Finding an optimal solution

We consider the problem of computing an optimal solution $G_{\mathcal{P}}$. That is, we are given a t-spanner $G = (V, E)$, and the aim is to compute a $t_{\mathcal{P}}$-spanner $G_{\mathcal{P}} = (V, E \cup \{e\})$.

A naïve approach to decide which edge to add is to test every possible candidate edge. The number of

[*]M.F. supported by Ministry of Science, Research and Technology of I. R. Iran.

[†]Department of Mathematics and Computing Science, TU Eindhoven, 5600 MB, Eindhoven, the Netherlands. m.farshi@tue.nl

[‡]Department of Computer Science, Utrecht University, Utrecht, the Netherlands. panos@cs.uu.nl

[§]NICTA, Sydney, Australia. National ICT Australia is funded by the Australian Government's Backing Australia's Ability initiative, in part through the Australian Research Council. Joachim.Gudmundsson@nicta.com.au such edges is obviously $\left(\frac{n(n-1)}{2} - m\right) = \mathcal{O}(n^2)$. Testing a candidate edge e entails computing the stretch factor of the graph $G' = (V, E \cup \{e\})$, therefore we briefly consider the problem of computing the stretch factor of a given Euclidean graph.

A trivial upper bound is obtained by computing the All-Pairs-Shortest-Path for the given graph G. Running Dijkstra's algorithm – implemented using Fibonacci heaps – gives the stretch factor of G in time $\mathcal{O}(mn + n^2 \log n)$ using linear space. This algorithm is quite slow and we would like to be able to compute the stretch factor more efficiently, but no faster algorithm is known for any graphs except planar graphs, paths, cycles and trees [4, 5].

Applying the above bounds for computing the exact stretch factor of a Euclidean graph gives us that $G_{\mathcal{P}}$ can be computed in time $\mathcal{O}(n^3(m + n \log n)))$ using linear space.

An improvement can be obtained by observing that when an edge (u, v) is about to be tested we do not have to check all possible shortest paths between two vertices $x, y \in V$ again, it suffices to check if there is a shorter path using the edge (u, v). That is, we only have to check the length of the paths $\delta_G(x, u) + |uv| + \delta_G(v, y)$ and $\delta_G(x, v) + |vu| + \delta_G(u, y)$, which can be done in constant time since $\delta_G(x, u)$ and $\delta_G(v, y)$ already have been computed (provided that we store this information). Hence by first computing all-pair-shortest paths of G we obtain:

Lemma 1 *Given a Euclidean graph G, an optimal solution $G_{\mathcal{P}}$ can be computed in time $\mathcal{O}(n^4)$ using $\mathcal{O}(n^2)$ space.*

3 Adding the bottleneck edge

In this section we study the approach of adding an edge between a pair of vertices in G that decides the stretch factor of G.

Consider an optimal solution $G_{\mathcal{P}}$ and denote by x and y the two endpoints of the edge added to G to obtain $G_{\mathcal{P}}$. Assume that a pair of vertices deciding the stretch factor of G is (u, v), i.e., the length of the path between u and v in G is exactly $t \cdot |uv|$. We call this edge a bottleneck edge of G. Let $G_{\mathcal{B}}$ be the graph obtained from G by adding the bottleneck edge, and let $t_{\mathcal{B}}$ be the stretch factor of $G_{\mathcal{B}}$.

Lemma 2 *Given a Euclidean graph G in $\mathbb{R}^d$ it holds that $t_{\mathcal{B}} < 3t_{\mathcal{P}}$.*

The main result of this section is:

Theorem 3 *Given a Euclidean graph $G = (V, E)$ one can in $\mathcal{O}(mn + n^2 \log n)$ time, using $\mathcal{O}(n)$ space, compute a $t_{\mathcal{B}}$-spanner $G' = (V, E \cup \{e\})$ where $t_{\mathcal{B}} < 3t_{\mathcal{P}}$.*

We end this section by giving a lower bound for the bottleneck approach.

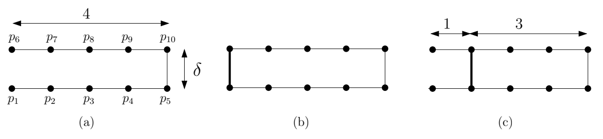


Figure 1: (a) The input graph G. (b) The bottleneck solution compared to (c) the optimal solution.

Observation 1 *There exists a Euclidean graph G such that, $(2 - \varepsilon) \cdot t_{\mathcal{P}} \leq t_{\mathcal{B}}$, for any $\varepsilon > 0$.*

Proof. Consider the graph G, as in Fig. 1(a). More specifically, G is a graph with ten vertices $p_i = ((i-1) \bmod 5, \lfloor i/5 \rfloor \cdot \delta)$, $1 \leq i \leq 10$, and nine edges (p_5, p_{10}) and (p_j, p_{j+1}), for $1 \leq j \leq 4$ and $6 \leq j \leq 9$. If we assume that δ is a very small positive real value then (p_1, p_6) is the bottleneck in G and $t_{\mathcal{B}} = \frac{4+\delta}{\delta}$, see Fig. 1(b).

In the case when edge (p_2, p_7) is added to G, as shown in Fig. 1(c), the resulting graph has stretch factor $(2 + \delta)/\delta$. Combining the upper and lower bounds gives $\frac{t_{\mathcal{B}}}{t_{\mathcal{P}}} \geq \frac{4+\delta}{2+\delta} = (2 - \varepsilon)$, where the last inequality follows if we set $\delta = \frac{2\varepsilon}{1-\varepsilon}$. $\square$

Hence, we have an upper bound of 3 and a lower bound of $(2 - \varepsilon)$ when adding the bottleneck edge to the input graph.

4 A $(2 + \varepsilon)$-approximation

In this section we will present a fast approximation algorithm which guarantees an approximation factor of $(2 + \varepsilon)$. The algorithm is similar to the algorithm presented in Section 2 in the sense that it tests candidate edges. Testing a candidate edge entails computing the stretch factor of the graph. The main difference is that we will show, in Section 4.1, that only a linear number of candidate edges needs to be tested to obtain a solution that gives a $(2 + \varepsilon)$-approximation, instead of a quadratic number of edges.

Moreover, Section 4.2 shows that the same approximation bound can be achieved by performing only a linear number of shortest path queries for each candidate edge. The candidate edges are selected by using the well-separated pair decomposition (WSPD) defined by Callahan and Kosaraju (see [2]). They showed that a WSPD of size $m = \mathcal{O}(s^d n)$ can be computed in $\mathcal{O}(s^d n + n \log n)$ time (s is called the separation constant of the WSPD).

4.1 Linear number of candidate edges

In this section we show how to obtain a $(2 + \varepsilon)$-approximation in cubic time.

The approach is straight-forward. First the algorithm computes the length of the shortest path in G between every pair of points in V. The distances are saved in a matrix M. Next, the well-separated pair decomposition is computed. Note that, in Step 5, the candidate edges will be chosen using the well-separated pair decomposition. Finally, steps 4–9, each candidate edge is tested by computing the stretch factor of the candidate graph.

Algorithm ExpandGraph(G, ε)
Input: Euclidean graph $G = (V, E)$ and a real constant $\varepsilon > 0$.
Output: Euclidean graph $G' = (V, E \cup \{e\})$.

```
1.  $M \leftarrow$All-Pairs-Shortest-Path dist. matrix of $G$.
2.  $\{(A_i, B_i)\}_{i=1}^k \leftarrow$WSPD of $V$ with $s = \frac{256}{\varepsilon^2}$.
3.  $t' \leftarrow \infty$.
4.  for $i \leftarrow 1$ to $k$
5.          Select a point $a_i \in A_i$ and a point $b_i \in B_i$.
6.          $G_i \leftarrow G = (V, E \cup (a_i, b_i))$.
7.          $t_i \leftarrow$StretchFactor$(G_i, M)$.
8.          if $t_i < t'$
9.              then $t' \leftarrow t_i$ and $e \leftarrow (a_i, b_i)$
10. return $G' = (V, E \cup \{e\})$.
```

Lemma 4 *Algorithm* ExpandGraph *requires* $\mathcal{O}(n^3/\varepsilon^{2d})$ *time and* $\mathcal{O}(n^2)$ *space.*

It remains to analyze the quality of the solution obtained from algorithm ExpandGraph. Let $\Delta(p, q)$ denote the set of point pairs in V such that $u, v \in V$ belongs to $\Delta(p, q)$ if and only if $(p, q) \in \delta_{G\cup\{(p,q)\}}(u, v)$. That is, the set of point pairs for which the shortest path between them in $G \cup \{(p, q)\}$ passes through (p, q).

Lemma 5 *For any given constant* $0 < \lambda \leq 1$*, there exists a point pair* $p, q \in V$ *such that for every pair* $(u, v) \in \Delta(p, q)$ *it holds that* $|uv| \geq \frac{\lambda}{2}|pq|$*, and the stretch factor of* $G \cup \{(p, q)\}$ *is bounded by* $(2 + \lambda) \cdot t_{\mathcal{P}}$.

Note that algorithm ExpandGraph might not test (p, q) stated in Lemma 5. However, in the following lemma it will be shown that algorithm ExpandGraph will test an edge (a, b) that is almost as good as (p, q).

Lemma 6 *For any given constant* $0 < \varepsilon \leq 1$ *it holds that the graph* G' *returned by algorithm* ExpandGraph *has stretch factor at most* $(2 + \varepsilon) \cdot t_{\mathcal{P}}$.

Proof. According to Lemma 5 there exists an edge (p, q) such that for every pair $(u, v) \in \Delta(p, q)$ it holds that $|uv| \geq \frac{\lambda}{2}|pq|$, and the stretch factor t_H of $H = G \cup \{(p, q)\}$ is bounded by $(2 + \lambda) \cdot t_{\mathcal{P}}$. Let $\{A_i, B_i\}$ be the well-separated pair computed in step 2 of the algorithm such that $p \in A_i$ and $q \in B_i$. Next consider the candidate edge (a_i, b_i) tested by the algorithm, such that $a_i, p \in A_i$ and $b_i, q \in B_i$. For simplicity of writing we will use a and b to denote a_i and b_i respectively.

Our claim is that the stretch factor t' of $G' = G \cup \{(a, b)\}$ is bounded by $(1+\varepsilon/4) \cdot t_H$. Thus setting $\lambda = \varepsilon/4$ would then prove the lemma since $(2 + \varepsilon/4)(1 + \varepsilon/4) < (2 + \varepsilon)$, for $\varepsilon \leq 1$.

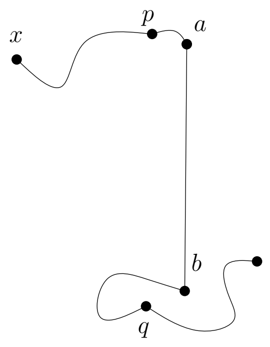


Figure 2: Illustrating the proof of Lemma 6.

Now we are ready to prove the claim. If for all pairs x, y, $(x, y) \notin \Delta(p, q)$ then the claim is obviously true, thus we only have to consider the pairs x, y for which it holds that $(x, y) \in \Delta(p, q)$, see Fig 2. It holds that:

$$d_G(a, p) = d_H(a, p) \quad \text{and} \quad d_G(b, q) = d_H(b, q). \qquad (1)$$

This follows from the fact that the closest pair x', y' for which it holds that $(x', y') \in \Delta(p, q)$ has inter point distance at least $|x'y'| \geq \frac{\varepsilon}{8}|pq|$, according to Lemma 5. It holds that $|ap|$ and $|bq|$ are bounded by $\frac{2}{s}\ |pq| \leq \frac{\varepsilon^2}{128}\ |pq|$ which is less than $\frac{\varepsilon}{8}|pq|$ since $\varepsilon < 1$. As a consequence $(p, q) \notin \delta_H(a, p)$ and $(p, q) \notin \delta_H(b, q)$. Hence, claim (1) holds, which we will need below.

Next, we consider the length of the path in G' between x and y as illustrated in Fig. 2. Recall that x and y are two arbitrary points of V for which it holds that $(x, y) \in \Delta(p, q)$.

$$\begin{aligned}
d_{G'}(x, y) &\leq d_G(x, p) + d_G(p, a) + |ab| + d_G(b, q) \\
&\quad + d_G(q, y) \\
&\leq d_G(x, p) + d_H(p, a) + |ab| + d_H(b, q) \\
&\quad + d_G(q, y) \qquad \text{(from (7))} \\
&< d_G(x, p) + (1 + 4/s) \cdot |pq| + d_G(q, y) \\
&\quad + \frac{4t_H}{s} \cdot |pq| \qquad \text{(WSPD property)} \\
&\leq d_H(x, y) + \frac{64t_H}{\varepsilon s} \cdot |xy| \qquad \text{(Lemma 5)} \\
&= d_H(x, y) + \frac{\varepsilon}{4} \cdot t_H \cdot |xy|
\end{aligned}$$

The stretch factor of the path in G' between x and y is:

$$\frac{d_{G'}(x, y)}{|xy|} \leq \frac{d_H(x, y)}{|xy|} + \frac{\frac{\varepsilon}{4} t_H |xy|}{|xy|} \leq \left(1 + \frac{\varepsilon}{4}\right) \cdot t_H.$$

□

We may now conclude this section with the following theorem.

Theorem 7 *Given a Euclidean graph $G = (V, E)$ in $\mathbb{R}^d$ one can in time $\mathcal{O}(n^3/\varepsilon^{2d})$, using $\mathcal{O}(n^2)$ space, compute a t'-spanner $G' = (V, E \cup \{e\})$, where $t' \leq (2+\varepsilon) \cdot t_{\mathcal{P}}$.*

4.2 Speed-up the algorithm

In the previous section we showed that a $(2+\varepsilon)$-approximate solution can be obtained by testing a linear number of candidate edges. Testing each candidate edge entails $\mathcal{O}(n^2)$ shortest path queries. One way to speed up the computation is to compute the approximate stretch factor. The problem of computing the approximate stretch factor was considered by Narasimhan and Smid in [5]. They showed the following fact:

Fact 1 *([5]) Given a Euclidean graph G and a real value $\varepsilon > 0$, a $(1+\varepsilon)^2$-approximative stretch factor of G can be computed by performing $\mathcal{O}(n/\varepsilon^d)$ many $(1+\gamma)$-approximate distance queries, where γ is a positive constant smaller than ε.*

Their idea is to compute a well-separated pair decomposition of size $s = 4(1+\varepsilon)/\varepsilon$, and then for each well-separated pair $\{A_i, B_i\}$ select an arbitrary pair $a_i \in A_i$ and $b_i \in B_i$. They prove that these are the only pairs for which the stretch factor needs to be computed.

We will use their idea to speed up step 7 of the algorithm from $\mathcal{O}(n^2)$ to $\mathcal{O}(n/\varepsilon^d)$. There will be two changes in the ExpandGraph algorithm. First, between steps 2 and 3, the following four lines are inserted:

```
- $\{(C_j, D_j)\}_{j=1}^{\ell}$ ←WSPD of $V$ with $s' = 4(1+\varepsilon)/\varepsilon$.
- for $j$ ←1 to $\ell$
    Select a point $c_j \in C_j$ and a point $d_j \in D_j$.
- $\mathcal{S} = \{(c_1, d_1), \ldots, (c_\ell, d_\ell)\}$
```

Then, in step 7 of ExpandGraph we will instead of computing the exact stretch factor of G_i make a call to ApproximateStretchFactor, or ASF for short, with parameters G_i, (a_i, b_i), M, and $\mathcal{S}$. Note that the number of point pairs in $\mathcal{S}$ is bounded by $\mathcal{O}(n/\varepsilon^d)$.

```
Algorithm ASF($G_i, e, M, \mathcal{S}$)
Input: Euclidean graph $G(V, E)$, edge $e = (a, b) \in E$,
    distance matrix $M$ and a set of point pairs $\mathcal{S}$.
Output: A real value $\mathcal{D}_i$.
1.  $\mathcal{D}_i$ ←1
2.  for each point pair $(c_j, d_j)$ in $\mathcal{S}$
3.          dist ←min$\{M[c_j, d_j], M[c_j, a] + |ab| + M[b, d_j], M[c_j, b] + |ba| + M[a, d_j]\}$
4.          $\mathcal{D}_i$ ←max$\{\mathcal{D}_i$, dist$/|c_j d_j|\}$
5.  return $\mathcal{D}_i$.
```

We denote the modified algorithm ExpandGraph2.

Theorem 8 *Given a Euclidean graph $G = (V, E)$ and a real constant $\epsilon > 0$ one can in $\mathcal{O}(nm + n^2(\log n + 1/\epsilon^{3d}))$ time, using $\mathcal{O}(n^2)$ space, compute a t'-spanner $G' = (V, E \cup \{e\})$ such $t' \leq (2+\epsilon) \cdot t_{\mathcal{P}}$.*

5 Open problems and Acknowledgements

Several problems remain open.

1. Is there an exact algorithm with running time $o(n^4)$ using linear space?
2. Can we achieve a $(1+\varepsilon)$-approximation within the same time bound as in Theorem 8?
3. A natural extension is to allow more than one edge to be added. Can we generalize our results to this case?

The authors would like to thank René van Oostrum for fruitful discussions during the early stages of this work.

References

[1] K. Alzoubi, X.-Y. Li, Y. Wang, P.-J. Wan, and O. Frieder. Geometric spanners for wireless ad hoc networks. *IEEE Transactions on Parallel and Distributed Systems*, 14(4):408–421, 2003.

[2] P. B. Callahan. *Dealing with higher dimensions: the well-separated pair decomposition and its applications.* Ph.D. thesis, Department of Computer Science, Johns Hopkins University, Baltimore, Maryland, 1995.

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

[4] S. Langerman, P. Morin, and M. A. Soss. Computing the maximum detour and spanning ratio of planar paths trees, and cycles. In *Proc. STACS*, pages 250–261, 2002.

[5] G. Narasimhan and M. Smid. Approximating the stretch factor of Euclidean graphs. *SIAM Journal of Computing*, 30(3):978–989, 2000.

[6] G. Navarro and R. Paredes. Practical construction of metric t-spanners. In *Proc. 5th Workshop on Algorithm Engineering and Experiments*, pages 69–81. SIAM Press, 2003.

[7] M. Smid. Closest point problems in computational geometry. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 877–935. Elsevier Science Publishers, Amsterdam, 2000.

An Exclusion Region for Minimum Dilation Triangulations

Christian Knauer* Wolfgang Mulzer*

Abstract

Given a planar graph G, the *graph theoretic dilation* of G is defined as the maximum ratio of the shortest-path distance and the Euclidean distance between any two vertices of G. Given a planar point set S, a triangulation of S that achieves minimum graph theoretic dilation is called a *minimum dilation triangulation* of S. In this paper, we show that a simple exclusion region for an edge e of the minimum dilation triangulation is given by the disk of radius $\alpha|e|$ centered at the midpoint of e, where α is any constant $< 3\cos(\pi/6)/(4\pi) \approx 0.2067$.

1 Introduction

In this paper, we are going to consider minimum dilation triangulations. The problem is as follows: Given a set S of points in the Euclidean plane, find a triangulation T of S such that the maximum dilation between any pair of these points in T is minimal, where the *dilation* between a pair of points (u, v) in S is defined as the ratio between the shortest path distance of u and v in T and the Euclidean distance $|uv|$ (see Section 2 for formal definitions of these terms). In Figure 1, we can see an example of a planar point set and two triangulations, one of which achieves a very low maximum dilation, while the other triangulation has a very high maximum dilation.

The maximum dilation between any pair of points in S with respect to a triangulation T of S is called the *graph theoretic dilation* of T, and the minimum graph theoretic dilation that any triangulation of S can achieve is called the graph theoretic dilation of S.

When considering optimal triangulations, it is instructive to look at local properties of the edges of these triangulations, since local properties improve our understanding of the structure of optimal triangulations and sometimes lead to efficient algorithms to compute them. One important class of local properties that has been studied for minimum weight triangulations and greedy triangulations is constituted by *exclusion regions*. Exclusion regions give us a necessary condition for the inclusion of an edge into an optimal triangulation: If u and v are two points in a given planar point set S, then the edge $e := \overline{uv}$ can only be contained in an optimal triangulation of

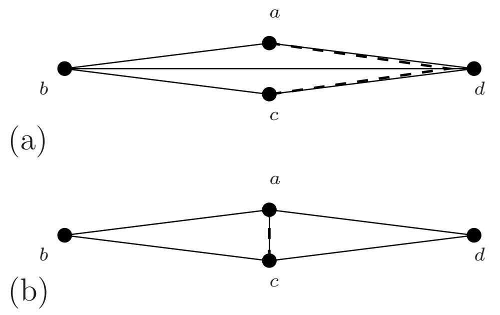


Figure 1: Two triangulations of point set $\{a, b, c, d\}$. In triangulation (a), the dilation between points a and c is very high, whereas triangulation (b) achieves a very low dilation. The bold dashed lines represent a shortest path between a and c in the respective triangulation.

S if no other points of S lie in certain parts of the exclusion region of S. For example, Das and Joseph [3] proved that e can only be included in the minimum weight triangulation of a point set S, if at least one of the two equilateral triangles with base e and base angle $\frac{\pi}{8}$ is empty (see Figure 2). This result was improved by Drysdale *et al.* [5], who proved that the base angle can be increased to $\pi/4.6$ and that also the disk of diameter $|e|/\sqrt{2}$ centered at the midpoint of e is an exclusion region for the minimum weight triangulation. A similar result with slightly different parameters also holds for the greedy triangulation [6]. In this paper, we are going to show that an analogous result applies to the minimum dilation triangulation. More specifically, we show that an edge e can only be included in the minimum dilation triangulation of S, if at least one of the two half circles with radius $\alpha|e|$ whose center is the center of e is empty (see Figure 2). Here α denotes any constant such that $0 < \alpha < 3\cos(\pi/6)/(4\pi) \approx 0.2067$.

Previous Work. Up to now, very little research has been done on minimum dilation triangulations, but there has been some work on estimating the dilation of certain types of triangulations that had already been studied in other contexts. Chew [2] shows that the rectilinear Delaunay triangulation has dilation at most $\sqrt{10}$. A similar result for the Euclidean Delaunay triangulation is given by Dobkin *et al.* [4]. They show that the dilation of the Euclidean De-

*Institut für Informatik, Freie Universität Berlin, Germany, {knauer, mulzer}@inf.fu-berlin.de

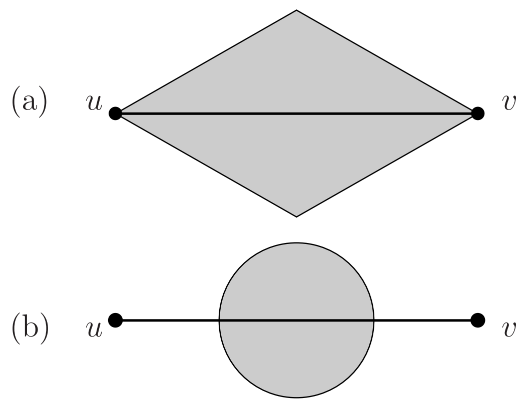


Figure 2: (a) shows the standard exclusion region for the minimum weight triangulation. (b) shows our exclusion region for the minimum dilation triangulation.

launay triangulation can be bounded from above by $\left(1+\sqrt{5}\right)\pi/2 \approx 5.08$. This bound was further improved to $2\pi/(3\cos(\pi/6)) \approx 2.42$ by Keil and Gutwin [8], and we are going to use this bound as an essential ingredient in our proof.

Das and Joseph [3] generalize these results by identifying two properties of planar graphs such that if A is an algorithm that computes a planar graph from a given set of points and if all the graphs constructed by A meet these properties, then the dilation of all the graphs constructed by A is bounded by a constant.

A more comprehensive survey of results on the graph theoretic dilation of planar and general graphs can be found in Eppstein's survey [7].

Surprisingly, very few results are known about the triangulations which actually achieve the optimum graph theoretic dilation. In his master's thesis, Mulzer [9] investigates the structure of minimum dilation triangulations for the regular n-gon, but beyond that not much is known.

2 Preliminaries

Let S be a finite set of points in the Euclidean plane, and let T be a triangulation of S. For any two points $u, v \in S$, the ratio between the shortest path distance $\pi_T(u,v)$ and the Euclidean distance $|uv|$ is called the *(relative) dilation* between u and v with respect to T, which we shall denote by $\delta_T(u,v)$. Formally, the dilation is defined as follows:

$$\delta_T(u,v) \overset{\text{def}}{=} \begin{cases} 1, & \text{if } u = v, \\ \frac{\pi_T(u,v)}{|uv|}, & \text{if } u \neq v. \end{cases}$$

The convention to define $\delta_T(v,v) = 1$ for any $v \in S$ is very natural, since from the definition it is immediate that $\delta_T(u,v) \geq 1$ for every $u, v \in S$, as clearly we have $\pi_T(u,v) \geq |uv|$ for all $u, v \in S$.

Intuitively, the dilation is a measure for the quality of the connection between u and v in T. If the dilation is large, this means that we have to travel a long way along the edges in T in order to reach v from u even though the direct route would be much shorter.

In order to get a measure for the quality of the connection between any two vertices of T, it is natural to take the maximum over all the dilations between pairs of vertices in T. This quantity is called the *graph theoretic dilation* of T. We will denote it by $\delta(T)$. The formal definition is this:

$$\delta(T) \overset{\text{def}}{=} \max_{u,v \in S} \delta_T(u,v).$$

If T has the property that its graph theoretic dilation is minimal among all triangulations of S, we call T a *minimum dilation triangulation* of S.

3 An Exclusion Region

Let $0 < \alpha < 3\cos(\pi/6)/(4\pi)$ be a constant, S a planar point set, $u, v \in S$ two points in the plane, and let D be the disk of radius $\alpha|uv|$ centered at the midpoint of line segment $e = \overline{uv}$. We are going to show that D is an exclusion region for e.

The basic idea is very simple: Even though we do not know much about the actual minimum dilation triangulation of a planar point set S, we know that the graph theoretic dilation of the Delaunay triangulation of S is bounded by the constant $\gamma = 2\pi/(3\cos(\pi/6))$ [8]. Furthermore, it is obvious that if we have an edge e and two points that are quite close to the center of e and that lie on opposite sides of e, then the dilation between these two points is very large, because the line segment e constitutes an obstacle that any path between these two points needs to circumvent (see Figure 1(a) for an example). Thus, all we need to check is that the dilation between any pair of points in the disk that lie on opposing sides of e is larger than γ, and then we know that if such a pair of points exists, then e cannot be contained in the minimum dilation triangulation of S, since the Delaunay triangulation would give us a better graph theoretic dilation than any triangulation containing e.

Thus, we assume that there exist two points $a, b \in S$ in D on opposite sides of e (see Figure 3). We need to show that $\delta_T(a,b) > \gamma$ for any triangulation T of S that contains line segment e. For this we need to know the shortest path distance between a and b in T, $\pi_T(a,b)$. Since the only thing we know about T is that T contains e, the best thing we can do is to lowerbound $\pi_T(a,b)$ by $\min(|au|+|ub|, |av|+|vb|)$.

The first thing we observe is that we can assume that the two points lie on the boundary of D, since the dilation between the intersection points of the line through a and b with the boundary of D is smaller than the dilation between a and b.

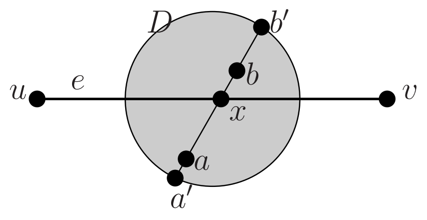


Figure 3: The situation described in Observation 1. The dilation between a' and b' is smaller than the dilation between a and b. x is the intersection point of $\overline{ab}$ and e.

Observation 1 *Let a be a point in D to the right of line segment $e = uv$, and let x be a point on e and in D. For $d > 0$, let $b(d)$ be the point to the left of line segment e on the half line ax such that $|xb(d)| = d$. Then the dilation $\delta(d)$ between a and $b(d)$ decreases as d increases.*

Proof. Due to the triangle inequality, the shortest path between a and b cannot include e, and hence $\delta(d)$ is given by

$$\delta(d) = \frac{\min(|ua| + |ub(d)|, |va| + |vb(d)|)}{|ax| + d}.$$

First, we are going to check that $\ell(d) := (|ua| + |ub(d)|)/(|ax| + d)$ is monotonically decreasing. By the law of cosines, the numerator can be written as $\mathrm{num}(d) = |ua| + \sqrt{|ux|^2 + d^2 - 2|ux|\cos\delta}$, where δ denotes the angle between $\overline{ux}$ and $\overline{xb(d)}$. An easy calculation shows $\mathrm{num}'(d) \leq 1$. The derivative of the denominator is 1. Therefore, by the mean value theorem, it follows that $\ell(d)$ is monotonically decreasing (note that the numerator is never smaller than the denominator), and the observation follows, since by a similar argument we can check that also $d \mapsto (|va| + |vb(d)|)/(|ax| + d)$ decreases monotonically, and hence $\delta(d)$ decreases. □

Now we are left with the task of bounding the dilation between two points on the boundary of D. First of all, it is clear that dilation $(2\alpha)^{-1}$ can be achieved when a and b are infinitesimally close to the two intersection points of D and e, respectively. We are going to show that this is already an optimal configuration. For our calculations we need a propitious parameterization. We proceed as follows: Let z be the center of D. By symmetry, we may assume that $\overline{ab}$ lies to the right of z. We describe the line segment $\overline{ab}$ by looking at the angle $\beta = \angle bza$ and the angle $x = \angle bzv - \beta/2$. The angle x describes the rotation of $\overline{ab}$ with respect to the position in which $\overline{ab}$ is perpendicular to e (see Figure 4). By our assumptions, we have $\beta \in (0, \pi]$ and $x \in (-\beta/2, \beta/2)$. Our parameterization is chosen in such a way that the following equations can be

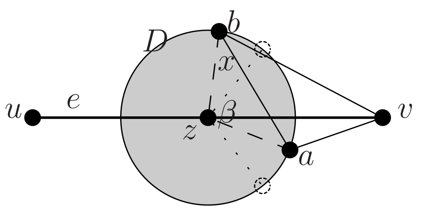


Figure 4: Our parameterization. The angle $\angle azb$ is called β. The offset x denotes the rotation of $\overline{ab}$ with respect to the vertical position (dashed lines).

written in a symmetric manner, which simplifies some of the calculations.

The angle $\angle bza$ is at most π, and hence the shortest path between a and b passes v. Thus, the dilation between a and b is given by

$$\delta(x, \beta) := \frac{f(x) + f(-x)}{2\alpha \sin(\beta/2)},$$

where

$$f(x, \beta) = \sqrt{0.25 + \alpha^2 - \alpha\cos(\beta/2 + x)}.$$

Here, $f(x)$ and $f(-x)$ denote the length of line segment $|vb|$ and $|va|$, respectively.

First, we fix $\beta \in (0, \pi]$ and optimize $x \mapsto \delta(x, \beta)$. An elementary yet tedious calculation yields the following observation:

Observation 2 *Let $\beta \in (0, \pi]$ be fixed. If we have $\cos(\beta/2) \leq 2\alpha$, the function $x \mapsto \delta(x, \beta)$ is minimal for $\cos(x) = (2\alpha)^{-1}\cos(\beta/2)$. Otherwise, $x \mapsto \delta(x, \beta)$ is minimal for $x = 0$.*

Now there are two cases to consider. If $\cos(\beta/2) \geq 2\alpha$, we need to look at $\delta(0, \beta) = f(0)/(\alpha \sin(\beta/2))$. Again, it turns out that this function is minimal if $\cos(\beta/2) = 2\alpha$, for this value of β we get that the dilation between a and b is exactly $(2\alpha)^{-1}$. What happens if $\cos(\beta/2) < 2\alpha$? In this case, we need to consider the value of $\delta(x, \beta)$, where x has the property that $\cos x = (2\alpha)^{-1}\cos(\beta/2)$. By using this property and by some trigonometric manipulations, we find that $\delta(x, \beta) = (2\alpha)^{-1}$. It follows that the dilation between a and b exhibits quite a remarkable behavior. If a and b are diametrically opposed, the minimum configuration with minimum dilation occurs when a and b are infinitesimally close to the two intersection points between D and e. As the chord $\overline{ab}$ gets shorter, the angle between e and $\overline{ab}$ in the optimal configuration becomes larger, until e and $\overline{ab}$ are perpendicular. As soon as this configuration is reached, the dilation between a and b increases as $\overline{ab}$ gets shorter.

Consequently, the minimum dilation between any points a and b in the two halves of D is $(2\alpha)^{-1}$, and by our choice of α and the upper bound on the graph theoretic dilation of the Delaunay triangulation [8], we can conclude with the following theorem:

Theorem 1 *Let $0 < \alpha < 3\cos(\pi/6)/(4\pi)$ be a constant, and let a and b be two points in the plane. Then the circle of radius $\alpha|ab|$ centered at the midpoint of $\overline{ab}$ is an exclusion region for the minimum dilation triangulation.*

Note that this exclusion region can be enlarged a little bit on the upper and lower boundary. For example, the dilation between the north- and south-pole of D is strictly less than γ. However, this would give us some curve of order 4 that is more difficult to handle than a simple circle.

4 Conclusion

We have made some progress in the field of minimum dilation triangulations and have shown that the concept of exclusion regions also makes sense for the minimum dilation triangulation. Usually, exclusion regions are applied as an initial filter of algorithms that compute minimum weight triangulations or greedy triangulations [1, 6]. It is easy to see that if the point set S is drawn independently and uniformly from a convex set C, then only an expected number of $O(n)$ edges pass the exclusion region test, so a large amount of edges can be discarded. In the algorithms for the other optimal triangulations, the remaining edges are processed using the greedy property (for the greedy triangulation) or some other local properties that give sufficient conditions for the inclusion of an edge (for the minimum weight triangulation). For the minimum dilation triangulation, however, it is not yet clear what to do with the remaining edges, since no other useful local properties are known that could be used in further processing steps. Finding such local properties remains an open problem.

It may also be interesting to look for configurations of points that show how tight our exclusion region is and whether it can be enlarged. At least it is clear that the exclusion region cannot come arbitrarily close to the endpoints of the edge, since otherwise the dilation between two points in the exclusion region can be arbitrarily close to 1.

References

[1] R. Beirouti and J. Snoeyink. Implementations of the LMT heuristic for minimum weight triangulations. *Proceedings of the Fourteenth Annual ACM Symposium on Computational Geometry*, 1998, pp. 96–105.

[2] L. P. Chew. There are planar graphs almost as good as the complete graph. *Journal of Computer and System Sciences*, vol. 39, 1989, pp. 205–219.

[3] G. Das and D. Joseph. Which triangulations approximate the complete graph? *Proceedings of the International Symposium on Optimal Algorithms*. Springer LNCS 401, 1989, pp. 168–192.

[4] D. P. Dobkin, S. J. Friedman, and K. J. Supowit. Delaunay graphs are almost as good as complete graphs. *Discrete and Computational Geometry*, vol. 5, 1990, pp. 399–407.

[5] R. L. Drysdale, S. A. McElfresh, and J. Snoeyink. On exclusion regions for optimal triangulations. *Disc. Appl. Math.*, vol. 109, 2001, pp. 49–65.

[6] R. L. Drysdale, G. Rote, and A. Aichholzer. A simple linear time greedy triangulation algorithm for uniformly distributed points. *IIG-Report-Series 408*, Technische Universität Graz, 1995.

[7] D. Eppstein. Spanning trees and spanners, in J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, elsevier, 1999, pp. 425–461.

[8] J. M. Keil and C. A. Gutwin. The Delaunay triangulation closely approximates the complete Euclidean graph. *Proceedings of the 1st Workshop on Algorithms and Data Structures.* Springer LNCS 382, 1989, pp. 47–56.

[9] W. Mulzer. *Minimum Dilation Triangulations for the regular n-gon.* Diploma Thesis. Freie Universität Berlin. 2004.

Improved Lower Bound on the Geometric Dilation of Point Sets*

Adrian Dumitrescu[†] Ansgar Grüne[‡] Günter Rote[§]

Abstract

Let G be an embedded planar graph whose edges are curves. The *detour* between two points p and q (on edges or vertices) of G is the length of a shortest path connecting p and q in G divided by their Euclidean distance $|pq|$. The maximum detour over all pairs of points is called the *geometric dilation* $\delta(G)$. Ebbers-Baumann, Grüne and Klein have shown that every finite point set is contained in a planar graph whose geometric dilation is at most 1.678, and some point sets require graphs with dilation $\delta \geq \pi/2 \approx 1.57$. They conjectured that the lower bound is not tight. We use new ideas, a disk packing result and arguments from convex geometry, to prove this conjecture. The lower bound is improved to $(1 + 10^{-11})\pi/2$.

1 Introduction

Consider a planar graph G embedded in $\mathbb{R}^2$, whose edges are curves[1] that do not intersect. Such graphs arise naturally in the study of transportation networks, like waterways, railroads or streets. For two points, p and q (on edges or vertices) of G, the *detour* between p and q in G is defined as

$$\delta_G(p, q) = \frac{d_G(p, q)}{|pq|}$$

where $d_G(p, q)$ is the shortest path length in G between p and q and $|pq|$ denotes the Euclidean distance, see Figure 1 for an example.

Good transportation networks should have small detour values. In a railroad system, access is only possible at stations, the vertices of the graph. Hence, to measure its quality we can take the maximum detour over all pairs of vertices. This results in the well-known concept of *graph-theoretic dilation* studied extensively in the literature on spanners, see [7] for a survey.

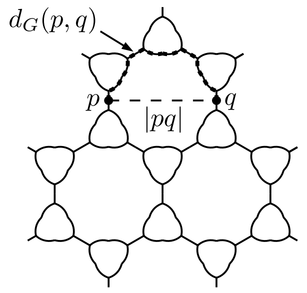


Figure 1: A grid G of small dilation $\delta(G) = \delta_G(p, q) = d_G(p, q)/|pq| < 1.678$ introduced in [5]

However, if we consider a system of urban streets, houses are usually spread everywhere along the streets. Hence, we have to take into account not only the vertices of the graph but all the points on its edges. The resulting supremum value is the *geometric dilation*

$$\delta(G) := \sup_{p,q \in G} \delta_G(p, q) = \sup_{p,q \in G} \frac{d_G(p, q)}{|pq|}$$

on which we concentrate in this article. Several papers [6, 13, 2] have shown how to efficiently compute the geometric dilation of polygonal curves. Besides this the geometric dilation was studied in differential geometry and knot theory under the notion of distortion, see e.g. [9, 12].

Ebbers-Baumann et al. [5] recently considered the problem of constructing a graph of lowest possible geometric dilation containing a given finite point set on its edges. Even for three given points this is a difficult task. Therefore they started by providing an upper and a lower bound on the dilation necessary to embed any finite point set, i.e. on the value

$$\Delta := \sup_{P \subset \mathbb{R}^2,\ P \text{ finite}} \ \inf_{G \supset P,\ G \text{ finite}} \delta(G)\ .$$

They showed that a slightly perturbed version of the grid in Figure 1 can be used to embed any finite point set. Thereby they proved $\Delta < 1.678$.

They also derived that $\Delta \geq \pi/2$, by showing that a graph G has to contain a cycle to embed a certain point set P_5 with low dilation, and by using that the dilation of every closed curve[2] C is bounded by $\delta(C) \geq \pi/2$.

*There is a full version [3] of this paper available.

[†]Computer Science, University of Wisconsin–Milwaukee, 3200 N. Cramer Street, Milwaukee, WI 53211, USA; ad@cs.uwm.edu

[‡]Universität Bonn, Institut für Informatik I, D-53117 Bonn, Germany; gruene@cs.uni-bonn.de; partially supported by a DAAD PhD-grant

[§]Freie Universität Berlin, Institut für Informatik, Takustraße 9, D-14195 Berlin, Germany; rote@inf.fu-berlin.de

[1]For simplicity we assume here that the curves are piecewise continuously differentiable, but think that the proofs can be extended to a broader class of curves.

[2]In this paper we use the notions “cycle” and “closed curve” synonymously.

They conjectured that this lower bound is not tight. It is known that circles are the only cycles of dilation $\pi/2$, see [4, Corollary 23], [1, Corollary 3.3], [12], [9]. And intuition suggests that one cannot embed complicated point sets with small dilation if every face of the graph has to be a circular disk. This idea would have to be formalized and still does not rule out that every point set could be embedded with dilation arbitrarily close to $\pi/2$. New ideas are needed to prove $\Delta > \pi/2$.

In Section 2 we show that cycles with dilation *close* to $\pi/2$ are *close* to circles, in some well-defined sense (Lemma 4). The lemma can be seen as an instance of a *stability result* for the geometric inequality $\delta(C) \geq \pi/2$, see [8] for a survey. Such results complement geometric inequalities (like the isoperimetric inequality between the area and the perimeter of a planar region) with statements of the following kind: When the inequality is fulfilled "almost" as an equation, the object under investigation is "close" to the object or class of objects for which the inequality is tight. An important idea in the proof of this stability result is a decomposition of any closed curve C into the two cycles C^* and M.

In Section 3 we use Lemma 4 to relate the dilation problem to a certain problem of packing and covering the plane by disks. By this we prove our main result $\Delta \geq (1 + 10^{-11})\pi/2$.

2 Result for Closed Curves

We want to prove that a simple closed curve C of low dilation is close to being a circle. We assume that C is given by an arc-length parameterization $c(t)$, $0 \leq t \leq |C|$, where $|C|$ denotes the length of C. Two points $p = c(t)$ and $\hat{p} = c(t \pm \frac{|C|}{2})$ on C that divide the length of C in two equal parts form a *halving pair* of C. The segment which connects them is a *halving chord*, and its length is the *halving distance*. We write $h = h(C)$ and $H = H(C)$ for the *minimum* and *maximum halving distance* of C.

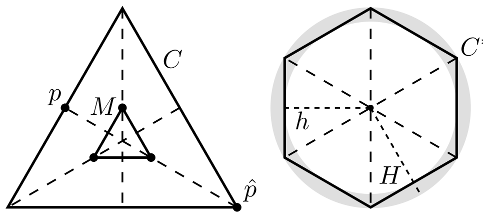


Figure 2: An equilateral triangle C, a halving pair $(p, \hat{p})$ and the derived curves C^* and M

To show that C is close to a circle, we consider a decomposition into two curves, see Figure 2 for an illustration. The *midpoint cycle* M is the cycle formed by the midpoints of the halving chords of C, and is given by the parameterization

$$m(t) := \frac{1}{2}\left(c(t) + c(t + \tfrac{|C|}{2})\right).$$

The curve C^* defined by

$$c^*(t) := \frac{1}{2}\left(c(t) - c(t + \tfrac{|C|}{2})\right)$$

is the result of the *halving pair transformation* defined in [4]. We get it by moving the midpoint of every halving chord to the origin. By definition, $c^*(t) = -c^*(t + \frac{|C|}{2})$, hence C^* is centrally symmetric. On the other hand, we have $m(t) = m(t + \frac{|C|}{2})$, and thus, M is traversed twice when C and C^* are traversed once. We define $|M|$ as the length of the curve M corresponding to one traversal.

The curve C^* has the same set of halving distances as C; thus, $h(C^*) = h(C) = h$ and $H(C^*) = H(C) = H$.

We have decomposed C into two components, from which it can be reconstructed:

$$c(t) = m(t) + c^*(t), \quad c\left(t + \tfrac{|C|}{2}\right) = m(t) - c^*(t) \quad (1)$$

To show that C is close to a circle, we first show that H/h is close to 1, i.e. C^* is close to a circle. Then, we prove that the length of the midpoint cycle is small. Combining both statements will deliver the desired result.

We use the following lemma to find an upper bound on the ratio H/h. Ebbers-Baumann et al. [4] have proved it for convex cycles using arguments from convex geometry similar to the ones in [10] but it can easily be extended to the non-convex case.

Lemma 1 *The geometric dilation $\delta(C)$ of any closed curve C satisfies*

$$\delta(C) \geq \arcsin\left(\frac{h}{H}\right) + \sqrt{\left(\frac{H}{h}\right)^2 - 1}.$$

Note that the function $g(x) = \arcsin 1/x + \sqrt{x^2 - 1}$ appearing on the right-hand side starts from $g(1) = \pi/2$ and is increasing on $[1, \infty)$. Approximating it by its Taylor expansion, we can show that $H/h \leq 1 + O(\varepsilon^{\frac{2}{3}})$ if $\delta(C) \leq (1 + \varepsilon)\frac{\pi}{2}$.

We still need an upper bound on the length $|M|$ of the midpoint cycle. We use the following lemma, which we think is of independent interest.

Lemma 2

$$4|M|^2 + |C^*|^2 \leq |C|^2.$$

Proof. Using the linearity of the scalar product and $|\dot{c}(t)| = 1$, we obtain from (1)

$$\begin{aligned}\langle \dot{m}(t), \dot{c}^*(t)\rangle &= \frac{1}{4}\left\langle \dot{c}(t) + \dot{c}(t + \tfrac{|C|}{2}), \dot{c}(t) - \dot{c}(t + \tfrac{|C|}{2})\right\rangle \\ &= \frac{1}{4}\left(|\dot{c}(t)|^2 - |\dot{c}(t + \tfrac{|C|}{2})|^2\right) = \frac{1}{4}(1 - 1) = 0\,.\end{aligned}$$

This means that the derivative vectors $\dot{c}^*(t)$ and $\dot{m}(t)$ are always orthogonal, thus $|\dot{m}(t)|^2 + |\dot{c}^*(t)|^2 = |\dot{c}(t)|^2 = 1$. This implies

$$\begin{aligned} |C| &= \int_0^{|C|} \sqrt{|\dot{m}(t)|^2 + |\dot{c}^*(t)|^2}\, \mathrm{d}t \\ &\geq \sqrt{\left(\int_0^{|C|} |\dot{m}(t)|\, \mathrm{d}t\right)^2 + \left(\int_0^{|C|} |\dot{c}^*(t)|\, \mathrm{d}t\right)^2} \\ &= \sqrt{4|M|^2 + |C^*|^2} \end{aligned}$$

The above inequality can be seen by a geometric argument. The left integral is the length of the curve $\gamma(s) := \left(\int_0^s |\dot{m}(t)|\, \mathrm{d}t, \int_0^s |\dot{c}^*(t)|\, \mathrm{d}t\right)$, while the right expression equals the distance of its end-points $\gamma(0) = (0,0)$ and $\gamma(|C|)$. □

Lemma 3 *If* $\delta(C) \leq (1+\varepsilon)\frac{\pi}{2}$, *then* $|M| \leq \frac{\pi h}{2}\sqrt{2\varepsilon + \varepsilon^2}$.

Proof. Because the dilation of C is at least the detour of a halving pair attaining minimum distance h, we get $(1+\varepsilon)\pi/2 \geq \delta(C) \geq |C|/2h$, implying

$$|C| \leq (1+\varepsilon)\pi h. \qquad (2)$$

If $|c^*(t)| < h/2$ held for any t, then, due to the central symmetry of C^*, the points $c^*(t)$ and $-c^*(t)$ would form a halving pair of distance $< h$, a contradiction. Hence, C^* encircles but does not enter the open disk $B_{h/2}(0)$ of radius $h/2$ centered at the origin 0. It follows

$$|C^*| \geq \pi h. \qquad (3)$$

By plugging everything together, we get

$$\begin{aligned} |M| &\overset{\text{Lemma 2}}{\leq} \frac{1}{2}\sqrt{|C|^2 - |C^*|^2} \\ &\overset{(2),(3)}{\leq} \frac{1}{2}\pi h\sqrt{(1+\varepsilon)^2 - 1} = \frac{\pi h}{2}\sqrt{2\varepsilon + \varepsilon^2}\ , \end{aligned}$$

which concludes the proof of Lemma 3. □

It should be intuitively clear (remember Figure 2) that the upper bound on H/h from Lemma 1 and the upper bound on $|M|$ of Lemma 3 imply that the curve C is contained in a thin ring if its dilation is close to $\frac{\pi}{2}$. This is the idea behind the omitted proof of the following lemma. We say that a cycle C is *enclosed in an* $(1+\varepsilon)$*-ring* if there is a radius $r > 0$ and a center $c \in \mathbb{R}^2$ such that the open region R bounded by C satisfies $B_r(c) \subseteq R \subseteq B_{(1+\varepsilon)r}(c)$.

Lemma 4 *Let* $C \subset \mathbb{R}^2$ *be any simple closed curve with dilation* $\delta(C) \leq (1+\varepsilon)\pi/2$ *for* $\varepsilon \leq 0.0001$. *Then* C *can be enclosed in a* $(1+3\sqrt{\varepsilon})$*-ring.*

By a special cycle C we can also show that this result cannot be improved apart from the coefficient of $\sqrt{\varepsilon}$. The lemma can be extended to a larger, more practical range of ε, by increasing the coefficient of $\sqrt{\varepsilon}$.

3 New Lower Bound

We will combine Lemma 4 with a disk packing result. A (finite or infinite) set $\mathcal{C}$ of disks in the plane with disjoint interiors is called a *packing.*

Theorem 5 (Kuperberg, Kuperberg, Matoušek and Valtr [11]) *Let* $\mathcal{C}$ *be a packing in the plane with circular disks of radius at most* 1. *Consider the set of disks* $\mathcal{C}'$ *in which each disk* $C \in \mathcal{C}$ *is enlarged by a factor of* 1.00001 *from its center. Then* $\mathcal{C}'$ *covers no square with side length* 4.

From Lemma 4 and Theorem 5 we deduce our main result:

Theorem 6 *The minimum geometric dilation* Δ *necessary to embed any finite set of points in the plane satisfies* $\Delta \geq (1 + 10^{-11})\pi/2$.

Proof. (Sketch) Consider the set $P := \{\, (x,y) \mid x,y \in \{-9,-8,\ldots,9\}\,\}$ of grid points with integer coordinates in the square $Q_1 := [-9,9]^2 \subset \mathbb{R}^2$, see Figure 3. We use a proof by contradiction and assume that there exists a planar connected graph G that contains P (as vertices or on its edges) and satisfies $\delta(G) \leq (1+10^{-11})\pi/2 < 2$. In the full paper we

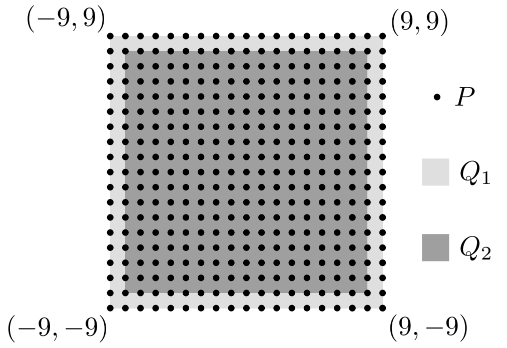


Figure 3: The point set $P := \{-9,-8,\ldots,9\}^2$ and the squares $Q_1 := [-9,9]^2$ and $Q_2 := [-8,8]^2$

show that if G attains such a low dilation, G contains a collection $\mathcal{M}$ of cycles with disjoint interiors which cover the smaller square $Q_2 := [-8,8]^2$. The length of each cycle $C \in \mathcal{M}$ is bounded by 8π implying that every disk encircled by C has a radius $r \leq 4$. Additionally, the dilation of every $C \in \mathcal{M}$ is at most $\delta(G) \leq (1+10^{-11})\pi/2$. Hence, Lemma 4 shows that every C has to be contained in an 1.00001-ring. It follows that the inner disks of these rings are disjoint and their 1.00001-enlargements cover Q_2 in contradiction to Theorem 5 (situation scaled by 4).

We would like to use the cycles bounding the faces of G for $\mathcal{M}$. Indeed, $\delta(G) < 2$ implies that they cover Q_2 (analogous to Figure 4b). However, their dilation could be bigger than the dilation $\delta(G)$ of the

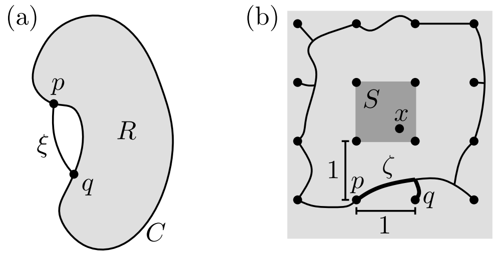


Figure 4: (a) The path ξ is a shortcut for some points of C. (b) Every $x \in Q_2$ is encircled by a cycle of length $\leq 12 \cdot \delta(G)$.

graph, see Figure 4a. G can offer shortcuts in the exterior of C, i.e., the shortest path between $p, q \in C$ does not necessarily use C.

Therefore, we have to find a different class of disjoint cycles covering Q_2 which do not allow shortcuts. The idea is to consider for every point x in Q_2 the shortest cycle of G such that x is contained in the open region bounded by the cycle. The regions of these cycles cannot intersect partly, we have $R_1 \cap R_2 = \emptyset$ or $R_1 \subseteq R_2$ or $R_2 \subseteq R_1$. If we define $\mathcal{M}$ to contain only the cycles maximal with respect to inclusion of their regions, it provides all the properties we need. Due to space limitations we can not prove all of them here.

However, one argument is displayed in Figure 4b. Every $x \in Q_2$ is contained in a square S of the integer grid. A shortest path ζ of G connecting neighbor points p, q of P next to S cannot enter S because $|\zeta| \leq \delta(G)|pq| < 2$. Hence, the concatenation of 12 such shortest paths contains a cycle of length $\leq 12\delta(G) \leq 12(1 + 10^{-11})\pi/2 \leq 8\pi$ encircling x. This shows that the regions of $\mathcal{M}$ cover Q_2 and that the length of every $C \in \mathcal{M}$ is bounded by $|C| < 8\pi$. $\square$

4 Conclusion

Our result looks like a very minor improvement over the easier bound $\Delta \geq \pi/2$, but it settles the question whether $\Delta > \pi/2$ and has required the introduction of new techniques. Our approximations are not very far from optimal, and we believe that new ideas are required to improve the lower bound to, say, $\pi/2 + 0.01$. An improvement of the constant 1.00001 in the disk packing result of [11] (Theorem 5) would of course immediately imply a better bound for the dilation.

We do not know whether the link between disk packing and dilation that we have established works in the opposite direction as well: Can one construct a graph of small dilation from a "good" circle packing (whose enlargement by a "small" factor covers a large area)? If this were true (in some meaningful sense which would have to be made precise) it would mean that a substantial improvement of the lower bound on dilation cannot be obtained without proving, at the same time, a strengthening of Theorem 5 with a larger constant than 1.00001.

References

[1] A. Abrams, J. Cantarella, J. Fu, M. Ghomi, and R. Howard. Circles minimize most knot energies. *Topology*, 42(2):381–394, 2002.

[2] P. K. Agarwal, R. Klein, C. Knauer, and M. Sharir. Computing the detour of polygonal curves. Technical report, Freie Universität Berlin, Fachbereich Mathematik und Informatik, 2002.

[3] A. Dumitrescu, A. Grüne, and G. Rote. On the geometric dilation of curves and point sets. http://arxiv.org/abs/math.MG/0407135, preprint, July 2004.

[4] A. Ebbers-Baumann, A. Grüne, and R. Klein. Geometric dilation of closed planar curves: New lower bounds. *to appear in special issue of CGTA dedicated to Euro-CG '04*, 2004.

[5] A. Ebbers-Baumann, A. Grüne, and R. Klein. On the geometric dilation of finite point sets. *to appear in Algorithmica*, 2005.

[6] A. Ebbers-Baumann, R. Klein, E. Langetepe, and A. Lingas. A fast algorithm for approximating the detour of a polygonal chain. *Computational Geometry: Theory and Applications*, 27(2):123–134, 2004.

[7] D. Eppstein. Spanning trees and spanners. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 425–461. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000.

[8] H. Groemer. Stability of geometric inequalities. In P. M. Gruber and J. M. Wills, editors, *Handbook of Convex Geometry*, volume A, pages 125–150. North-Holland, Amsterdam, Netherlands, 1993.

[9] M. Gromov, J. Lafontaine, and P. Pansu. Structures métriques pour les variétés riemanniennes. volume 1 of *Textes math.* CEDIC / Fernand Nathan, Paris, 1981.

[10] T. Kubota. Einige Ungleichheitsbeziehungen über Eilinien und Eiflächen. *Sci. Rep. Tŏhoku Univ.*, 12:45–65, 1923.

[11] K. Kuperberg, W. Kuperberg, J. Matousek, and P. Valtr. Almost-tiling the plane by ellipses. *Discrete & Computational Geometry*, 22(3):367–375, 1999.

[12] R. B. Kusner and J. M. Sullivan. On distortion and thickness of knots. In S. G. Whittington, D. W. Sumners, and T. Lodgejournal, editors, *Topology and Geometry in Polymer Science*, volume 103 of *IMA Volumes in Math. and its Applications*, pages 67–78. Springer, 1998.

[13] S. Langerman, P. Morin, and M. A. Soss. Computing the maximum detour and spanning ratio of planar paths, trees and cycles. *STACS 2002*, pages 250–261, 2002.

On Computing Fréchet Distance of Two Paths on a Convex Polyhedron [*]

Anil Maheshwari Jiehua Yi

Abstract

We present a polynomial time algorithm for computing Fréchet distance between two simple paths on the surface of a convex polyhedron.

1 Introduction

Distance measures used to match geometric patterns include: Hausdorff-distance, Fréchet-distance, uniform distance, etc. Alt and Godau [1, 2] proposed that Fréchet distance is one of the most fundamental measures to compute the similarity between two polygonal curves. Fréchet distance is often referred to as the dog-leash distance [1]. The unique property is its sensitivity to the order along the two curves. Fréchet distance is the minimum leash distance that can keep the person and the dog walking on their own tracks from the beginning to the end (without retracting). Some issues on similarity related to Fréchet distance have been considered, for example, find a curve that is similar to a given curve [3, 4], and the application of Fréchet distance on protein backbone matching.

This paper focuses on the following problem: Given a convex polyhedron P consisting of n triangular faces, and two simple paths Z and Z' on the surface of P, how can we use Fréchet distance to measure the similarity of Z and Z'? For the sake of simplicity, assume that each segment in these paths is an edge of P, and Z consists of z segments and Z' consists of z' segments. Referring to the dog-leash distance, the person is walking on the path Z and the dog is walking on the path Z'. The leash defines a geodesic path on the surface of P. Here, Fréchet distance is measured using Euclidean shortest path distance on the surface of P.

In this paper, we present a polynomial time algorithm to compute Fréchet distance between the paths Z and Z' on the surface of a convex polyhedron P. To accomplish this we make use of two data structures: (i) a data structure of the visibility diagram that encodes shortest path information for any pair of points on a pair of edges (e_s, e_t), where $e_s \in Z$ and $e_t \in Z'$ and (ii) the data structure of the free space diagram proposed in [1] for paths in plane. The novelty is in adapting the free space diagram with the aid of visibility diagrams for the problem discussed in this paper. In the next section we discuss the visibility diagram data structure and in Section 3 we present an algorithm to compute Frèchet distance for paths on a convex polyhedron.

[*]Research supported by NSERC. School of Computer Science, Carleton University, Ottawa, Canada K1S 5B6, {anil,jyi}@scs.carleton.ca

2 Visibility Diagram

The visibility diagram of a pair of edges, say e_s and e_t, lying on the surface of P is a data structure that concisely represents geodesic distances between any pair of points p and q, where $p \in e_s$ and $q \in e_t$. (Due to the lack of space we cannot discuss the literature regarding the computation of geodesic paths on a convex polyhedron. For detailed discussion on this we refer the reader to [5, 6, 10].) We make use of the algorithm of [6], and that in turn makes calls to the algorithm of [5].

Algorithm 1 *Visibility-Diagram*

```
(1) Construct the edge sequence tree $T$ of edge $e_s$ using the algorithm of [5].
(2) Identify those edge sequences in tree $T$ which start at $e_s$ and end at the edge $e_t$. Let the set of these edge sequences be $\mathcal{E}$.
(3) Unfold each of the edge sequence in $\mathcal{E}$ and construct the visibility polygon for each unfolding.
(4) Compute the overlay of the visibility polygons to obtain the visibility diagram. Label each area in the overlay with the corresponding edge sequence.
(5) Output the final visibility diagram.
```

Details of this algorithm are provided in [9]. This algorithm uses the concept of *edge sequence*. According to the shortest path properties in [7, 8], a shortest path $\Pi(p \in e_s, q \in e_t)$ on P is identified uniquely by its endpoints and the sequence of edges $\{e_s, e_1, e_2, ..., e_k, e_t\}$ that it crosses. This sequence of edges is called an *edge sequence* of P $(\xi(\Pi(p, q)))$. The faces $\{f_1, f_2, ..., f_{k+1}\}$ that the shortest path traversed can be unfolded to a plane, by rotating the face f_1 into the coordinate system of f_2 around the common edge e_1 of f_1 and f_2, and then rotate f_1 and f_2 to the coordinate system of f_3, and so on. Following these steps, all of the faces can be located in the coordinate system of f_{k+1}, and this forms a planar graph. Geodesic paths in the unfolding map to straight line segments. Refer to Figure 1(b).

Define the domain $z = e_s \times e_t$ as a unit square and it is an affine mapping of the edges e_s and e_t on

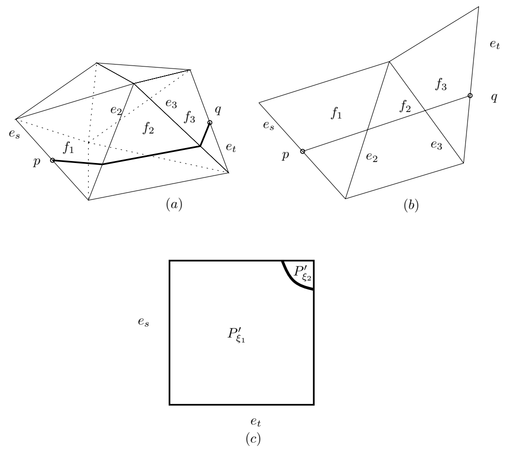


Figure 1: (a): A sequence of edges and faces that a shortest path from $p \in e_s$ to $q \in e_t$ passes through; (b): The planar unfolding relative to the edge sequence; (c): The visibility diagram of e_s and e_t. P'_{ξ_1} corresponds to the outer boundary of the unfolded edge sequence in (b), which is a polygon. P'_{ξ_2} corresponds to another unfolded edge sequence.

$[0, 1]$, see Figure 1(c). The visibility diagram is defined as the partition of the domain z, each partition corresponds to an unfolded edge sequence starting at e_s and ending at e_t. (Mount [8] has shown that the number of such edge sequences is $O(n^2)$.) In a partition the pair of points are visible to one another in their corresponding unfolded edge sequence. During the construction of the visibility diagram, if the pair of points in a partition can see each other in more than one unfolded edge sequence, then this partition is further subdivided, until each partition corresponds to one unfolded edge sequence. It can be shown that the boundary between each pair of partition in the visibility diagram is a hyperbolic curve, and each partition is a polygon. Observe that geodesic path for any pair of points in e_s and e_t can be computed from their corresponding unfolded edge sequence in the visibility diagram. Thus, the visibility diagram for a pair of edges can be computed by simultaneously overlaying $O(n^2)$ visibility polygons corresponding to each of the unfolded edge sequence; it can be computed in $O(n^3 \log n)$ time.

3 Algorithm to compute Fréchet Distance

First we briefly outline Fréchet Diagram for two polygonal curves Z and Z' in plane as described in [1]. They used affine mapping to represent a continuous and piecewise linear curve. If the curve Z is a line segment and similarly the curve Z' is a line segment then the set

$$F_\epsilon = \{(s,t) \in [0,1]^2 \mid d(Z(s), Z'(t)) \leq \epsilon\} \quad (1)$$

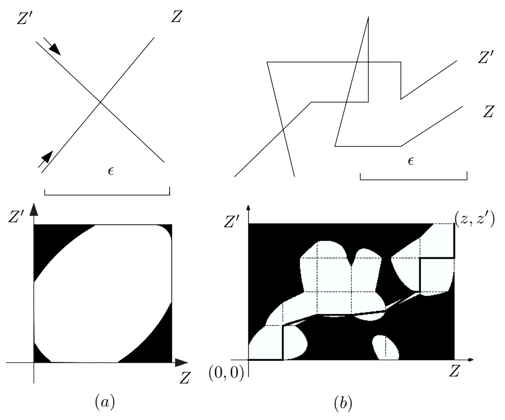


Figure 2: (a): Two segments Z and Z' and their free space diagram for a given ϵ. (b): Two polygonal curves Z and Z' and their free space diagram for a given ϵ. A monotone path from $(0,0)$ to (z, z') in the free space diagram.

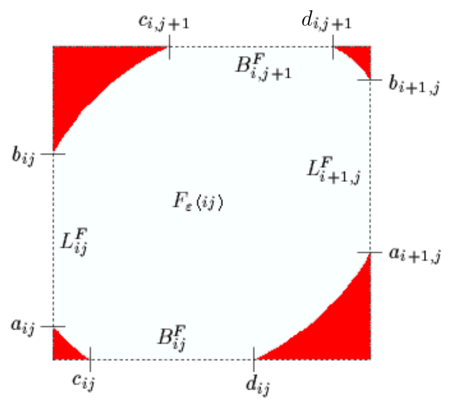


Figure 3: The boundary values of a unit cell in the free space diagram that must be calculated

describes all of the pairs of points in the affine mapping of Z and Z', whose Euclidean distance is at most ϵ. Figure 2(a) shows line segments Z and Z', and a distance $\epsilon > 0$; F_ϵ is the white area within the unit square, which is an ellipse [1], subsequently called as the *free space diagram*. Figure 2(b) shows polygonal curves Z and Z' with z and z' segments, respectively, and their free space diagram F_ϵ. This is obtained by combining the free space diagrams for each pair of segments of Z and Z'.

Lemma 1 *[1] For polygonal curves Z and Z' in plane, their Fréchet distance, $\delta_F(Z, Z') \leq \epsilon$, only if there exists a curve within the corresponding free space diagram F_ϵ from $(0,0)$ to (z, z') that is monotone in both coordinates.*

In order to ensure that there is a passage for the path between neighboring cells in the diagram, Figure 3 illustrates certain boundary values that needs to be calculated. These values correspond to the in-

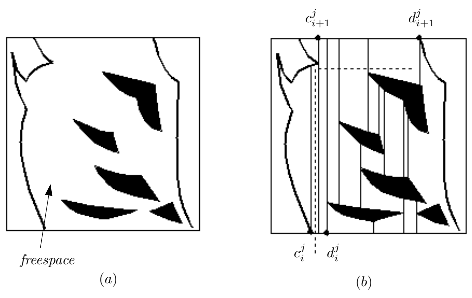


Figure 4: The trapezoidation of a free space after union. (a) : The free space (white areas). (b) : After trapezoidation and executing BFS (the dashed line is one of the paths), the upper boundary value is $c^j_{i+1}d^j_{i+1}$.

tersections of the ellipse with the boundary of the cell. Above lemma provides a mechanism to check whether Fréchet distance is at most ϵ. In [1] it is shown that the exact value of the distance is determined by an ϵ corresponding to one of the following cases: (i) $(0,0) \in F_\epsilon$ and $(z,z') \in F_\epsilon$, (ii) $L^F_{i,j}$ or $B^F_{i,j}$ becoming nonempty for some pair (i,j), or (iii) $a_{i,j} = b_{k,j}$ or $c_{i,j} = d_{i,k}$ for some i,j,k. Therefore, to determine the exact distance, one needs to apply Lemma 1 for only the set of critical values of ϵ as determined by the above cases. It turns out that the total number of critical values is $O(z^2z' + zz'^2)$, and hence Fréchet distance between two paths Z and Z' in the plane can be computed in $O(zz'\log(zz'))$ time.

In the rest of this paper we sketch how we can adapt the free space diagram with the aid of the visibility diagrams for the case of convex polyhedron. Let $Z : [0..z]$ and $Z' : [0..z']$ be the two paths on the boundary of convex polyhedron consisting of z and z' segments, respectively. Moreover, $Z(i-1)Z(i)$ and $Z'(j-1)Z'(j)$, for $1 \leq i \leq z, 1 \leq j \leq z'$, represents those segments. For a fixed ϵ, we construct a free space cell for a pair of edges $Z(i-1)Z(i)$ and $Z'(j-1)Z'(j)$. The outer boundary of the free space cell is a unit square and it is exactly the same square as the boundary of the visibility diagram of those two edges. But the main difficulty arises due to the fact that the polygons in the visibility diagram belong to different unfolded edge sequences, and each unfolded edge sequence has its own coordinate system in the plane. Because of this, the free spaces of $Z(i-1)Z(i)$ and $Z'(j-1)Z'(j)$ in the unfolded edge sequences are different from one another. Thus, we compute the intersection of the free space cell with the visibility polygon, where both the cell and the polygon correspond to the same unfolded edge sequence. For each pair of points in the intersection area, their shortest path distance is at most ϵ. The free space diagram of the unit cell of $Z(i-1)Z(i)$ and $Z'(j-1)Z'(j)$ is obtained by taking the union of the free spaces for all the

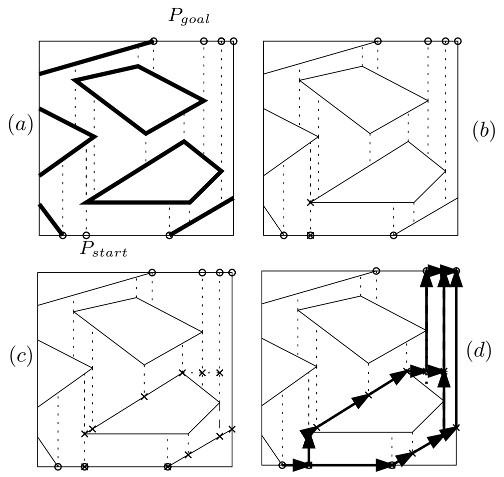


Figure 5: (a)Placing boundary nodes in the trapezoidal map. $(b)(c)$: Illustration of placing interior nodes in the trapezoidal map. (d): Adding arcs between the nodes to form a directed graph. For simplifying the figures, boundaries of the polygons are drawn as straight line segments.

corresponding edge sequences. Refer to Figure 4(a). The free space diagram for Z and Z' for a fixed ϵ is obtained by applying the above computation steps on each pair of segments.

Once we have the free space diagram for a specific value of ϵ, we need to test whether we have a monotone path from $(0,0)$ to (z,z'). Observe that if there is a monotone path then each part of the path in the corresponding cells must be monotone. The algorithm starts from the first pair of edges including $(0,0)$, ends at the last pair of edges including (z,z'), and computes the boundary values for each unit cell in the free space diagram subject to the monotone path restrictions. If (z,z') can be reached, then this particular choice of ϵ results in a valid monotone path in the corresponding free space diagram. The boundary values for each cell are obtained by performing breadth first search on a modified dual map of the trapezoidal decomposition of each cell, refer to Figure 5. The computation of the boundary values for a unit cell proceeds in three steps. First, we place candidate nodes in the trapezoidal map from where a monotone path could pass through. The nodes placed on the boundary of the unit cell are candidates for the boundary values, we use P_{start} and P_{goal} to mark them (marked as “ $\circ$ ” in Figure 5(a)), respectively. The nodes placed in the interior of the free space cell (marked as “ $\times$ ” in Figure 5$(b)(c)$) can be reached monotonically from the neighboring nodes placed earlier. Second, connect the neighboring nodes with the directed arcs, the arcs must also be in the free space area. Thus, a directed graph is formed in the free space area of the unit cell. Third, by applying the breadth first search in this directed graph, a monotone path (if it exists) from a starting point P_{start} to a point P_{goal} is identified (see Figure 4(b)). There is a technical detail involved here, and this arises due to the fact that the edges of the polygons in a unit cell can be hyperbolic curves or elliptical arcs, refer to Figure 4(a). Therefore, trapezoidation needs to take this in account, as well as care must be taken while placing the nodes to ensure that the directed arcs reside in the free space. We omit the details of this part here.

Suppose that we have already computed the visibility diagram for a pair of edges in $O(n^3 \log n)$ time. For a fixed ϵ, we can show that the union of the free spaces in the unit cell for the pair of edges can be computed in $O(n^2 \log n)$ time, since there are $O(n^2)$ polygons in the visibility diagram; Computing the boundary value of a unit cell takes $O(n^2 \log n)$ time, which includes the trapezoidal map and the directed acyclic graph construction, and performing breadth first search. Therefore, in $O(zz'n^3 \log n)$ time, we can determine whether Fréchet distance between the paths Z and Z' is at most ϵ. This is summarized in the following lemma, and it can be viewed as the analog of Lemma 1 for the case of convex polyhedron.

Lemma 2 *For simple paths Z and Z' on the surface of convex polyhedron, their Fréchet distance, $\delta_F(Z, Z') \leq \epsilon$, only if there exists a curve within the corresponding free space diagram F_ϵ from $(0, 0)$ to (z, z') that is monotone in both coordinates. Moreover this can be determined in $O(zz'n^3 \log n)$ time where z and z' are the number of the segments in Z and Z', and n is the number of faces on P.*

Next we need to determine what are the critical values of ϵ, and then we can search among them to determine Fréchet distance between the paths. The three kinds of critical values of ϵ, as in the planar case, are also suitable for the convex polyhedron case. But due to the complex nature of cells in this case, we need to extend the third case; the main idea is captured in the following observation. Assume that the minimum value of ϵ is not determined by the first two cases. Then we claim that the minimum value of ϵ that ensures that there is a monotone path in the free space diagram will consists of a segment that is parallel to one of the coordinate axes.

Recall that in the two dimensional case, the third case corresponds to either $a_{i,j} = b_{k,j}$ or $c_{i,j} = d_{i,k}$ for some i, j, k. In the light of the above observation, for the convex polyhedron case this needs to be extended to the case when $a_{i,j}$ and $b_{k,j}$ are located inside the cells, if we use the same labels for the points inside the cells as the labels for the boundary values; see Figure 3. Imagine that the black areas, corresponding to non feasible regions, where $a_{i,j}$ and $b_{i+1,j}$ are located, are moved inside the unit cell. These critical values can be computed from the free space diagrams, and will require solving a degree four equation in ϵ since the boundaries of the corresponding polygons (elliptic or hyperbolic) are defined by degree two equations.

We claim that there is an upper bound of $O((z^2z' + zz'^2)n^4)$ on the number of critical values of ϵ, since each cell has $O(n^2)$ polygons and every two of them need to be tested in the computation of the third kind of critical values. In addition to this there are potentially $O(zz'n^2)$ critical values for ϵ as determined in the first two cases. Sorting the critical values first, then using the binary search, for each value of ϵ, we test whether we can obtain a monotone path in the free space diagram. The smallest ϵ, that results in a monotone path located in the free space diagram, is Fréchet distance between the paths Z and Z'. We summarize the result in the following theorem.

Theorem 3 *Fréchet-distance between two simple paths Z and Z' on the surface of a convex polyhedron, consisting of n triangular faces, can be computed in $O((z^2z' + zz'^2)n^4 \log(zz'n))$ time, where Z consists of z segments and Z' consists of z' segments.*

References

[1] H. Alt, M. Godau. Computing the Fréchet Distance Between Two Polygonal Curves. In *IJCGA*, 5, 1995, pp 75-91.

[2] M. Godau. A Natural Metric for Curves–Computing the Fréchet Distance for Polygonal Chains and Approximation Algorithms. In *STACS, LNCS*, 480, 1991, pp 127-136.

[3] H. Alt,A. Efrat, G. Rote, C. Wenk. Matching Planar Maps. In *J. of Algo.*, 49, 2003, pp 262-283

[4] H. Alt, C. Knauer, C. Wenk. Matching Polygonal Curves with respect to the Fréchet Distance. In *STACS*, 2001, pp 63-74.

[5] J. Chen, Y. Han. Shortest Paths on a Polyhedron. In *IJCGA*, 6, 1996, 127-144.

[6] Y. H.Hwang, R. C.Chang, H. Y.Tu. Finding all Shortest Path Edge Sequences on a Convex Polyhedron. In *WADS, LNCS*, 382, 1989 August, pp 252-266.

[7] M. Sharir, A. Schorr. On Shortest Paths in Polyhedral Spaces. In *SIAM J. of Comp.*, 15, 1986, pp 193-215.

[8] D. M.Mount. The Number of Shortest Path on the Surface of a Polyhedron. In *SIAM J. of Comp.* 19, 1990 August, pp 593-611.

[9] J. Yi. Computing Fréchet Distance of Two Paths on A Convex Polyhedra. In *M.C.S Thesis, Carleton University*, 2004 August.

[10] L. Aleksandrov, A. Maheshwari, J.-R.Sack. Determining approximate shortest paths on weighted polyhedral surfaces. In *JACM*, 50, 2005, pp 25-53.

Semi-Computability of the Fréchet Distance Between Surfaces

Helmut Alt Maike Buchin *

Abstract

The Fréchet distance is a distance measure for parameterized curves or surfaces. Using a discrete approximation, we show that for triangulated surfaces it is *upper semi-computable*, i.e., there is a non-halting Turing machine which produces a monotone decreasing sequence of rationals converging to the result. It follows that the decision problem, whether the Fréchet distance of two given surfaces lies below some specified value, is recursively enumerable.

1 Introduction

The *Fréchet distance* was first introduced by Fréchet for curves [Fré06] and later for surfaces [Fré24]. The idea of the Fréchet distance is to take into account the "flow" of the curve or surface given by its parameterization. In some cases, the Fréchet distance is a more suitable distance measure than the commonly used Hausdorff distance (see [AG95]).

Formally the Fréchet distance is defined as follows.

Definition 1 *Let f, g be parameterizations of k–dimensional surfaces, i.e., continuous functions*

$$f, g : [0,1]^k \to \mathbb{R}^d, \quad k \leq d.$$

Then their Fréchet distance is

$$\delta_F(f,g) := \inf_{\sigma:[0,1]^k \to [0,1]^k} \max_{t \in [0,1]^k} ||f(t) - g(\sigma(t))||.$$

where the reparameterization *σ ranges over all orientation preserving homeomorphisms.*

The norm $||.||$ underlying the definition in this paper can be the L_1-, L_2-, or L_∞-norm as long as it can be computed or approximated by rational arithmetic.

For dimension $k = 1$ of the parameter space, in particular for polygonal curves, δ_F is known to be computable in polynomial time [AG95]. For two–dimensional surfaces, however, the computation of the Fréchet distance surprisingly seems to be much harder. In fact, Godau showed [God98] that computing the Fréchet distance between triangulated surfaces even in two–dimensional space is NP-hard. It remained open, how hard the problem really is, not even its computability could be shown.

In this paper, we present a partial result concerning the computability. More specifically, we will show that the Fréchet distance between triangulated surfaces is *upper semi-computable*, i.e., there is a non-halting Turing machine which produces a monotone decreasing sequence of rationals converging to the result. It follows that the decision problem whether the Fréchet distance of two given surfaces lies below some specified value is *recursively enumerable*.

The computationally hard part of computing the Fréchet distance for dimensions $k > 1$ seems to be, that according to the definition, the infimum over all homeomorphisms of the parameter space has to be taken. For dimension $k = 1$ the orientation-preserving homeomorphisms on $[0,1]$ are the continuous, onto, monotone increasing functions on $[0,1]$. For higher dimensions the homeomorphisms can be much "wilder".

We tackle this problem by approximating the homeomorphisms by discrete maps which are easier to handle. We do this by first approximating arbitrary homeomorphisms by piecewise linear homeomorphisms which is a known result from topology. The piecewise linear homeomorphisms are then approximated by *mesh homeomorphisms*, i.e., homeomorphisms that are compatible with certain subdivisions of the original triangulations of the parameter spaces. Finally, for mesh homeomorphisms on fine subdivisions the distance between the surfaces can be approximated by the distances at only a finite number of points.

It remains open, whether the Fréchet distance between triangulated surfaces is a computable function in the strong sense.

2 Model of computation, main results

We assume that the input to our algorithm are two triangulated surfaces in space $\mathbb{R}^d, d \geq 2$, which are represented as *piecewise linear* parameterizations $f, g : [0,1]^2 \to \mathbb{R}^d$. For simplicity, we will denote the surfaces themselves by f and g, as well.

Piecewise linear means that the parameter spaces of f and g are triangulated and on each triangle f and g are linear maps in the sense that for a triangle $\Delta = \langle u, v, w \rangle$ we have $f(\lambda_1 u + \lambda_2 v + \lambda_3 w) = \lambda_1 f(u) + \lambda_2 f(v) + \lambda_3 f(w)$ for all $\lambda_1, \lambda_2, \lambda_3$ with $\lambda_1 + \lambda_2 + \lambda_3 = 1$ and g has an analogous property.

*Freie Universität Berlin, Institut für Informatik, {alt,mbuchin}@inf.fu-berlin.de. This research was partially supported by the Deutsche Forschungsgemeinschaft within the European graduate program 'Combinatorics, Geometry, and Computation' (No. GRK 588/2).

We denote the triangulated parameter spaces of f and g by K and L. The vertices of the individual triangles have rational coordinates, and the coefficients describing the linear maps are rational, as well. Thus, a problem instance has a canonical finite representation which can be given as input to a Turing machine.

We will show that the Fréchet distance between triangulated surfaces is computable in a weak sense according to the following definition which has been considered in the complexity-of-real-functions community (see, e.g., [WZ00]).

Definition 2 *A function $\varphi : \mathbb{N} \to \mathbb{R}$ is called* upper (lower) semi-computable *if there is a Turing machine which on input x outputs an infinite, monotone decreasing (increasing) sequence of rational numbers converging to $\varphi(x)$.*

Now we can formulate our main result:

Theorem 1 *The Fréchet distance between two triangulated surfaces in space $\mathbb{R}^d$, $d \geq 2$, is upper semi-computable.*

Theorem 1 immediately implies the following corollary, where $\langle f, g, a\rangle$ denotes some standard encoding of a triple consisting of two triangulated surfaces f and g, and some rational $a > 0$.

Corollary 2 *The set $\{\langle f, g, a\rangle \mid \delta_F(f, g) < a\}$, i.e., the decision problem for the Fréchet distance between triangulated surfaces, is recursively enumerable.*

In fact, consider the Turing machine producing a monotone decreasing sequence converging to $\delta_F(f, g)$ which exists by Theorem 1. Stop this Turing machine as soon as it produces a value less than a. This algorithm will eventually halt for all triples $\langle f, g, a\rangle$ in the language and else will run forever.

The computability of δ_F in the strong sense of computability theory of real functions (see, e.g., [Wei00]) remains open, since the sequence produced by the algorithm in the proof of Theorem 1 is not shown to *effectively* converge to $\delta_F(f, g)$, i.e., we cannot give any estimate on the rate of convergence.

Our proof can be modified to show a weaker form of Theorem 1 for more *general surfaces*. More precisely, if we just assume that the parameterizations f and g are *computable real functions*, it is still correct that there is an algorithm producing on input f, g (represented, say, by the Turing machines computing f and g) an infinite sequence of rational numbers converging to $\delta_F(f, g)$. However, this sequence is not necessarily monotone decreasing, and the corollary cannot be deduced anymore.

3 Approximating the homeomorphisms

In this section, we approximate homeomorphisms arbitrarily closely by mesh homeomorphisms.

Let us first recall some standard definitions and notations from topology. For a simplicial complex K, a triangulation in our case, let K^m denote its m^{th} *barycentric subdivision*, where in one subdivision step the barycenters of the previous simplices are taken as vertices. *Mesh(K)* denotes the maximal diameter of simplices in K, again triangles in our case. The *underlying space of* K, denoted by $|K|$, is the set of all points lying in simplices of K. In our case $|K|$ is always the unit square $[0, 1]^2$.

We now define mesh homeomorphisms.

Definition 3 *Given two triangulations K and L, a piecewise linear homeomorphism $h : |K^m| \to |L^n|$ is called a* mesh homeomorphism *if it maps the edges of K^m to edge chains of L^n, i.e., polygonal chains made up of edges of L^n.*

For approximating homeomorphisms arbitrarily closely by mesh homeomorphisms, we need only a weak form of closeness which is defined as follows.

Definition 4 *Given two homeomorphisms $h, h' : |K| \to |L|$ on triangulations K and L, let*

$$d_K(h, h') := \max_{\Delta \in K} \delta_H(h(\Delta), h'(\Delta))$$

where $\Delta \in K$ ranges over all triangles in K and δ_H denotes the Hausdorff distance.

Now we can approximate homeomorphisms by mesh homeomorphisms.

Lemma 3 *Let K and L be triangulations, $\sigma : |K| \to |L|$ a homeomorphism, $m \in \mathbb{N}$, and $\varepsilon > 0$. Then there exist $n \in \mathbb{N}$ and a mesh homeomorphism $h : |K^m| \to |L^n|$ such that $d_{K^m}(\sigma, h) < \varepsilon$.*

Proof. We omit the details of this proof in this extended abstract but sketch the main idea.

By a theorem from topology (see, e.g., chapter 6 in [Moi77]), a homeomorphism can be approximated arbitrarily closely by a piecewise linear homeomorphism. We use this as a first step, because piecewise linear homeomorphisms are easier to handle than arbitrary homeomorphisms. For a piecewise linear homeomorphism, we see that it can be approximated arbitrarily closely (in the sense of Definition 4) by a mesh homeomorphism. Together this proves the lemma.

The idea of approximating piecewise linear homeomorphisms by mesh homeomorphisms, is to subdivide sufficiently using, e.g., barycentric subdivision. Because of growing degrees of vertices and growing fineness of the triangulations, we can find mesh homeomorphisms arbitrarily close to a piecewise linear homeomorphism. A simple example is shown in Figure 1.

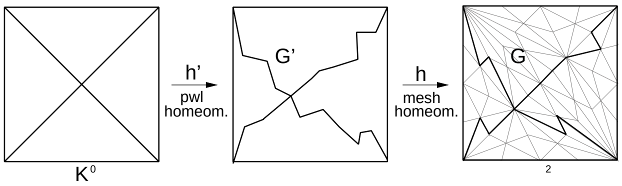


Figure 1: Approximating a piecewise linear homeomorphism by a mesh homeomorphism.

□

4 Discrete Fréchet distance

In this section we define a discrete Fréchet distance for surfaces and show that it is equal in value to the Fréchet distance.

We define the discrete Fréchet distance of two surfaces by taking the infimum over all mesh homeomorphisms and for each mesh homeomorphism taking the maximum over distances at vertices.

More formally, we define

Definition 5 *Let f, g be parametrized, triangulated two–dimensional surfaces in $\mathbb{R}^d$, $d \geq 2$, with underlying triangulations K, L respectively, of the parameter space, i.e.,*

$$f : |K| \to \mathbb{R}^d, \quad g : |L| \to \mathbb{R}^d$$

are piecewise linear maps. Then their discrete Fréchet distance *is defined as*

$$\delta_{dF}(f,g) := \inf_{\substack{m,n \\ h:|K^m|\to|L^n|}} \max_{\Delta\in K^m_T} \max_{\substack{v\in V_\Delta \\ w\in M^n_{h(\Delta)}}} ||f(v)-g(w)||$$

where h ranges over all orientation preserving mesh homeomorphisms, K^m_T is the set of triangles in K^m, V_Δ are the vertices of Δ, and $M^n_{h(\Delta)}$ is the set of vertices of L^n that lie in $h(\Delta)$.

First we show that this definition yields a discrete Fréchet distance not smaller than the Fréchet distance.

Lemma 4 $\delta_F \leq \delta_{dF}$

Proof. Any mesh homeomorphism is, in particular, a homeomorphism. Therefore, it suffices to show that for a mesh homeomorphism $h : |K^m| \to |L^n|$ we can bound the pointwise maximum by the maximum taken at vertices, i.e.,

$$\max_{t\in[0,1]^2} ||f(t)-g(h(t))|| \leq \max_{\Delta\in K^m_T} \max_{\substack{v\in V_\Delta \\ w\in M^n_{h(\Delta)}}} ||f(v)-g(w)||. \tag{1}$$

To see this, let $t \in [0,1]^2$ be arbitrary. Then t lies in a triangle Δ of K^m and $h(t)$ lies in a triangle Δ' of $h(\Delta) \subset L^n$. Since f and g are piecewise linear and K^m and L^n are refinements of the underlying triangulations of the parameter spaces, $f(\Delta)$ and $g(\Delta')$ are triangles, as well. Since the maximum distance between points of two triangles is attained between two corners, we have that $||f(t) - g(h(t))|| \leq ||f(v) - g(w)||$ for some $v \in \Delta, w \in \Delta'$. Taking the maximum on both sides yields equation (1). □

Now we show that also the discrete Fréchet distance is not larger than the Fréchet distance.

Lemma 5 *For all $\varepsilon > 0$, $\quad \delta_{dF} \leq \delta_F + \varepsilon$.*

Proof. The idea is that for any homeomorphism there is a mesh homeomorphism arbitrarily close and for the mesh homeomorphism the distance computation at vertices comes arbitrarily close to the distance computation on all parameter values by sufficient subdivision of the domain complex.

Let σ be a homeomorphism close to realizing δ_F, i.e., $\max_{t\in[0,1]^2} ||f(t) - g(\sigma(t))|| \leq \delta_F + \varepsilon_1$ for a small $\varepsilon_1 > 0$.

By Lemma 1, for any $\varepsilon_2 > 0$ and any $m \in \mathbb{N}$ there is a mesh homeomorphism $h : |K^m| \to |L^n|$ such that $d_{K^m}(\sigma, h) \leq \varepsilon_2$.

Let Δ be some triangle in $|K^m|$ and v one of its vertices. Since $d_{K^m}(\sigma, h) \leq \varepsilon_2$, for any $w \in h(\Delta) \subset L^n$ there is an $x \in \sigma(\Delta)$ with $||w - x|| < \varepsilon_2$. Using $t = \sigma^{-1}(x)$ and the Lipschitz-continuity of g we get $||g(w) - g(\sigma(t))|| < c_g \cdot \varepsilon_2$ for some $t \in \Delta$ where c_g denotes the Lipschitz constant of g.

Since t and v lie in the same triangle $\Delta \in K^m$, we have $||v - t|| \leq \text{mesh}(K^m)$ and $||f(v) - f(t)|| \leq c_f \cdot \text{mesh}(K^m)$ with c_f the Lipschitz constant of f.

Putting everything together and using the triangle inequality repeatedly we get

$$\begin{aligned}
\delta_{dF} &\leq \max_{\Delta\in K^m_T} \max_{\substack{v\in V_\Delta \\ w\in M^n_{h(\Delta)}}} ||f(v) - g(w)|| \\
&\leq \max_{\Delta\in K^m_T} \max_{\substack{v\in V_\Delta \\ x\in\sigma(\Delta)}} ||f(v) - g(x)|| + c_g \cdot \varepsilon_2 \\
&\leq \max_{\Delta\in K^m_T} \max_{t\in\Delta} ||f(t) - g(\sigma(t))|| + c_g \cdot \varepsilon_2 \\
&\quad + c_f \cdot \text{mesh}(K^m) \\
&\leq \delta_F + \varepsilon_1 + c_g \cdot \varepsilon_2 + c_f \cdot \text{mesh}(K^m).
\end{aligned}$$

Since $\varepsilon_1, \varepsilon_2$, and $\text{mesh}(K^m)$ can be made arbitrarily small, this concludes the proof. □

Lemmas 2 and 3 yield the following corollary.

Corollary 6 $\delta_F = \delta_{dF}$

5 Semi-Computability of the Fréchet distance

We can now give an algorithm showing the upper semi-computability of the Fréchet distance between triangulated surfaces as claimed in Theorem 1. This algorithm will, on input f, g, run forever and produce a monotone decreasing sequence of rational numbers converging to $\delta_F(f,g)$.

Algorithm CompFrec(f,g)

Input: Triangulated surfaces f, g, including triangulations K, L of the parameter spaces, in a finite description as explained in Section 2.

```
1 $D := \infty$;
2 for all $(m,n) \in \mathbb{N} \times \mathbb{N}$ do
  2.1 generate the barycentric subdivisions $K^m$ of $K$ and $L^n$ of $L$, let $E = \{e_1, ..., e_k\}$ be the set of edges in $K^m$;
  2.2 for all $k$-tuples $(\pi_1, ..., \pi_k)$ of simple polygonal chains in $L^n$ do
    2.2.1 assign the polygonal chain $\pi_i$ to the edge $e_i$ for $i = 1, ..., k$ and check whether this assignment results in an orientation preserving homeomorphic image of $K^m$, i.e., whether
      2.2.1.1 the edges on the boundary of $|K|$ are mapped onto the boundary of $|L|$ preserving the orientation; and
      2.2.1.2 if a set of edges in $K^m$ share an endpoint, the corresponding chains do, as well; and
      2.2.1.3 other than that, there are no intersection points between two chains;
  2.3 If the test in step 2.2.1 is passed, the chains form a subdivision of $|L|$ such that each triangle $\Delta$ of $K^m$ has a corresponding area $H_\Delta \subset |L|$.
    2.3.1 for each triangle $\Delta$ of $K^m$ do
      2.3.1.1 for all vertices $v$ of $\Delta$ and all vertices $w$ of $L^n$ lying in $H_\Delta$ do compute $||f(v) - g(w)||$;
    2.3.2 $M :=$ the maximum of all the values found in step 2.3.1.1;
    2.3.3 $D := \min(D, M)$; output $D$;
```

In essence, algorithm *CompFrec* approximates the discrete Fréchet distance which is, by Section 4 the same as the Fréchet distance. Line 2 can be realized by some standard enumeration method for pairs of integers.

Observe, that the number of k-tuples of polygonal chains of L^n checked in step 2.2 is finite. In fact, it is bounded by $(l!)^k$ where l is the number of edges in L^n, which itself is exponential in n, whereas k is exponential in m. But efficiency is not the issue here.

In step 2.3.1.1 we assume that the norm $||.||$ underlying the Fréchet distance can be evaluated by rational operations. This is correct for, e.g., the L_1- or L_∞-norm but not directly for L_2. In that case, one should rather operate with the square of the distance in line 2.3.1.1 and output some suitable rational approximation of $\sqrt{D}$ (which is possible) in line 2.3.3.

Note that checking that the boundary of $|K|$ is mapped orientation preserving onto the boundary of $|L|$ in step 2.2.1.1, entails that the mesh homeomorphism is orientation preserving also on the interior.

For each pair $(m,n) \in \mathbb{N} \times \mathbb{N}$ all mesh homeomorphisms $h : K^m \to L^n$ are evaluated, i.e.,

$$\delta_{h,m,n} = \max_{\Delta \in K_T^m} \max_{\substack{v \in V_\Delta \\ w \in M^n_{h(\Delta)}}} ||f(v) - g(w)||$$

(see Definition 5) is computed[1].

To see that the algorithm produces values arbitrarily close to $\delta_{dF}(f,g)$, observe that any neighborhood of $\delta_{dF}(f,g)$ must, by Definition 5, contain some value of the form $\delta_{h,m,n}$. The algorithm will eventually encounter that pair (m,n) and the subdivision corresponding to h and output $\delta_{h,m,n}$.

By line 2.3.3 the output sequence is monotone decreasing. Since for all triples (h,m,n) by Definition 5, $\delta_{h,m,n} \geq \delta_{dF}(f,g)$, line 2.3.3 is justified.

Since by Corollary 6 $\delta_F = \delta_{dF}$, algorithm *CompFrec* arbitrarily closely approximates $\delta_F(f,g)$ which proves Theorem 1.

References

[AG95] H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *Internat. J. Comput. Geom. Appl.*, 5:75–91, 1995.

[Fré06] M. Fréchet. Sur quelques points du calcul fonctionnel. *Rendic. Circ. Mat. Palermo*, 22:1–74, 1906.

[Fré24] M. Fréchet. Sur la distance de deux surfaces. *Ann. Soc. Polonaise Math.*, 3:4–19, 1924.

[God98] M. Godau. *On the complexity of measuring the similarity between geometric objects in higher dimensions.* PhD thesis, Freie Universität Berlin, 1998.

[Moi77] E. E. Moise. *Geometric Topology in Dimensions 2 and 3*, volume 47 of *Graduate Texts in Mathematics.* Springer-Verlag, New York, Heidelberg, Berlin, 1977.

[Wei00] K. Weihrauch. *Computable Analysis.* Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin, Heidelberg, 2000.

[WZ00] K. Weihrauch and X. Zheng. Computability on continuous, lower semi-continuous, and upper semi-continuous real functions. *Theoretical Computer Science*, 234:109–133, 2000.

[1]A more detailed analysis shows that, in fact, it suffices to consider only the pairs $(m, 2m), m \in \mathbb{N}$

Matching Surfaces with Characteristic Points

Darko Dimitrov, Christian Knauer, Klaus Kriegel*

Abstract

We study approximation algorithms for a matching problem that is motivated by medical applications. Given a small set of points $P \subset \mathbb{R}^3$ and a surface S, the optimal matching of P with S is represented by a rigid transformation which maps P as 'close as possible' to S. Previous solutions either require polynomial runtime of high degree [2] or they make use of heuristic techniques which could be trapped in some local minimum. We propose a modification of the problem setting by introducing subsets of characteristic points $P_c \subseteq P$ and $S_c \subseteq S$, and assuming that points from P_c must be matched with points from S_c. We will show that especially in the case $|P_c| \geq 2$ this restriction results in new fast and reliable algorithms for the matching problem.

1 Introduction

Today an increasing number of surgeries is supported by medical navigation systems. The basic task of such a system is to transform real world data (positions in the operating field) into a 3-dimensional model (CT or MR) and to display the transformed position in the model. Real world data are gaged by optical, electromagnetic or mechanical tracking systems. A common technique for computing the transformation is based on markers which are fixed on bones. The markers have to be fixed already during the model acquisition. Their positions in the model are computed using appropriate image processing methods. Later, at the beginning of the surgery, at least three markers must be gaged with the tracking system. Since the total number of markers is small, one could compute the correct matching transformation even by brute force techniques. A more advanced approach making use of geometric hashing techniques is presented in [3].
There is strong need to develop algorithmic methods for computing a transformation without using markers. The main reason for that is an anatomical one: in many cases (e.g. spinal surgery) it would be very hard or even impossible to fix markers before the surgery. One solution is to gage a few points on the surface of a bone and to compute the corresponding points in the model. This point registration is a hard algorithmic problem, which cannot be solved by the following standard approaches:
1) The gaged points could be anywhere on the model surface and hence, a combinatorial search does not work.
2) The number of gaged points is too small to apply surface reconstruction and surface matching algorithms.
Moreover, the registration is part of the surgery and thus real time algorithms are required. In contrast to that, it is possible to spend more time for preprocessing the model. Here, we try to retrieve some ideas of the landmark approach to that new setting. The role of markers could be played by so-called characteristic points. Such points can be determined by the surgeon, based only on their anatomic properties, e.g., the root of the nose or of the thorn of a vertebra. If a set of characteristic points is known in the model and the surgeon can track at least three of them, the old landmark registration algorithms can be applied. Our main goal is to solve the registration problem if only two characteristic points can be tracked. To compute the transformation in that case, one must track some more (non-characteristic) points on the surface.

In this paper we present our approach for solving this problem and sketch some first results. In the next section we introduce some notations and give a formal definition of the problem. In section 3 we present the basic algorithm and show how to use this method for the approximation of the optimal matching.

2 Problem description

We consider two point sets P and S in $\mathbb{R}^3$. Usually we assume that S is the (infinite) set of points on a triangulated surface. The corresponding triangulation will be denoted by $\mathcal{S}$. However, this assumption is not crucial. If S is a finite, dense sample of points on a surface, the algorithms presented in the next sections, can be applied with small changes.

Our main goal is to register P into a model S. The quality of the registration will be evaluated by the *directed Hausdorff distance*. The distance between a point a and a compact point set B in d-dimensional space $\mathbb{R}^d$ is defined as

$$\mathrm{dist}(a, B) = \min_{b \in B} ||a - b||$$

*Institut für Informatik, Freie Universität Berlin, darko/knauer/kriegel@inf.fu-berlin.de

where $|| \cdot ||$ is the Euclidean norm in $\mathbb{R}^d$. For two compact sets A, B in the one-sided Hausdorff distance from A to B is defined as

$$\overrightarrow{H}(A,B) = \max_{a\in A} \operatorname{dist}(a,B) = \max_{a\in A}\min_{b\in B} ||a-b||.$$

The size of a problem instance (P, S) depends on two parameters: k, the number of points in P, and n, the number of triangles in $\mathcal{S}$. We remark that in our applications $k << n$ can be assumed. Moreover, we assume that two subsets of characteristic points $S_c \subseteq S$ and $P_c \subseteq P$ are given. In order to prepare a precise analysis of our algorithms we introduce additional parameters $k_c = |P_c|$ and $n_c = |S_c|$. Both parameters should be seen as some reasonable constants. The special role of characteristic points is expressed by the additional requirement, that each $p \in P_c$ must be mapped onto (or close to) a characteristic point $q \in S_c$.

To proceed to algorithmic solutions we have to classify several types of matchings.

Definition 1 *Given two parameters $\mu, \eta \geq 0$ a rigid transformation $t : \mathbb{R}^3 \to \mathbb{R}^3$ is called (μ, η)-matching if the following two conditions hold:*

1. $\mu(t) := \overrightarrow{H}(t(P \setminus P_c), S) \leq \mu$, *and*

2. $\eta(t) := \overrightarrow{H}(t(P_c), S_c) \leq \eta$.

If ϵ is an upper bound for $\mu(t)$ and $\eta(t)$ we denote t as an *ϵ-matching*. In line with the notations above, we have $\epsilon(t) = \max(\mu(t), \eta(t))$. The minimal $\epsilon(t)$ is denoted by ϵ_{opt}, and a corresponding matching is an *optimal matching*. For a given $\lambda > 1$, a matching t is a *λ-approximate matching*, if $\epsilon(t) \leq \lambda\epsilon_{opt}$.

Furthermore, we introduce the notion of *semioptimal matchings*. To this end we fix a set $\overline{S} = \{\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_{k_c}\}$ of predefined matching positions for the characteristic points $P_c = \{p_1, p_2, \ldots, p_{k_c}\}$. Now we restrict our attention to matchings t with $t(p_i) = \bar{s}_i$ for $i = 1, \ldots, k_c$. Let us denote this set of matchings by $\mathcal{M}_{\overline{S}}$. We assume that P_c and $\overline{S}$ are congruent, because otherwise $\mathcal{M}_{\overline{S}} = \emptyset$.

A matching $t \in \mathcal{M}_{\overline{S}}$ is a $(\mu(t), \eta_0)$-matching, where $\eta_0 = \overrightarrow{H}(\overline{S}, S_c)$ is a common value for all matchings in $\mathcal{M}_{\overline{S}}$. A matching $t \in \mathcal{M}_{\overline{S}}$ is called *semioptimal matching* (with respect to $\overline{S}$) if $\mu(t)$ is minimal.

A trivial case with $|\mathcal{M}_{\overline{S}}| \leq 6$ occurs, if P_c contains three or more non-collinear points. If additionally the side lengths of the triangle spanned by the three points are pairwise different, there is only one matching in $\mathcal{M}_{\overline{S}}$. Thus, we will focus our attention to matchings with two characteristic points. In a first step we design an algorithm to compute semioptimal matchings for a given set $\overline{S}$. Then, based on the semioptimal solution, we show how to compute a λ-approximate matching for any $\lambda > 1$.

3 The 2 point case

3.1 Semioptimal matching

Now, let us assume that the matching positions $\overline{s_1}, \overline{s_2}$ for the two characteristic points p_1, p_2 are already given (see figure 1). First, we present an algorithm which reports (for any μ) all transformations t with the given matching positions for p_1 and p_2 and with $\mu(t) \leq \mu$.

Basic Algorithm (outline)

1. Fix a rigid transformation $t_0 : \mathbb{R}^3 \to \mathbb{R}^3$ such that $t_0(p_1) = \overline{s_1}$, $t_0(p_2) = \overline{s_2}$. For all $p_i \in P\backslash\{p_1, p_2\}$ let $C_i = C(p_i)$ be the circle with the following properties (see figure 1):

   i) the center of C_i is on the line defined by p_1 and p_2,

   ii) C_i is lying in a hyperplane perpendicular to $\overline{p_1, p_2}$, and

   iii) p_i is on C_i.

2. Consider the transformed circle $t_0(C_i)$ and let the point $p'_i(\alpha)$ rotate along this circle starting from $t_0(p_i)$, i.e., $p'_i(0) = t_0(p_i)$. Compute sets of intervals $I_i = \{\alpha \mid \; dist(p'_i(\alpha), S) \leq \mu\}$, for $i = 3, \ldots, k$.

3. Compute $I = \cap_{i=3}^{k} I_i$. For each $\alpha \in I$, $r_\alpha(s, s') \circ t_0$ is a rigid transformation mapping P onto S, where $r_\alpha(s, s')$ is the rotation around axis $\overline{s, s'}$ with angle α.

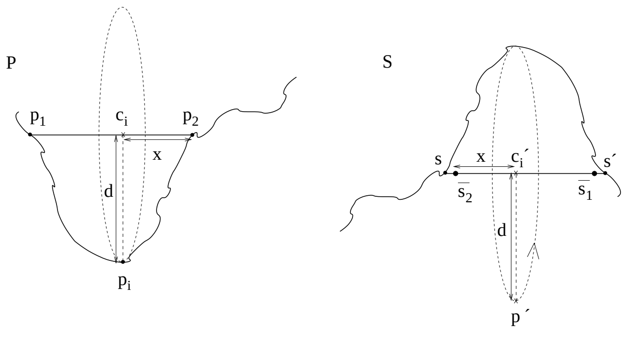


Figure 1: Corresponding points and the rotation of the point p'_i

A straightforward analysis shows that the algorithm runs in $O(kn \log n)$ time.
We remark that this time bound can be improved by a refined analysis under some assumption about the surface representation. The main idea is a subdivision of the bounding box of the surface S into $\sqrt{n} \times \sqrt{n} \times \sqrt{n}$ subboxes. The assumptions on the surface representation imply that each subbox intersects only a constant number of surface triangles. Since each cycle

intersects at most $O(\sqrt{n})$ subboxes, one can compute an interval set I_i considering only $O(\sqrt{n})$ triangles (if μ is not larger than the minimal subbox side length). We will leave the details of this analysis to a full paper.

The algorithm above can be used as a decision algorithm answering the question whether for a given μ there is some matching t with $t(p_1) = s, t(p_2) = s'$ and $\mu(t) \leq \mu$ for all other points of P. Thus, using binary search one can approximate a semioptimal matching. However, it is also possible to compute the precise value μ of a semioptimal matching by the following modification of the basic algorithm. Instead of computing the interval sets $I_i = \{\alpha \mid \operatorname{dist}(p_i'(\alpha), S) \leq \mu\}$ we compute the functions $f_i(\alpha) := \operatorname{dist}(p_i'(\alpha), S)$. This function is the lower envelope of the distance functions of a rotating point to the surface triangles. Thus, the description complexity of f_i is $O(n \log n)$, see [5]. Then, instead of computing $I = \cap_{i=3}^{k} I_i$, we compute the upper envelope f of all functions f_i. The minimum of f is the μ-value of a semioptimal matching.

3.2 The approximation problem

There are two groups of standard approaches to this Problem. The first group consists of simulated annealing [6] and ICP variants [1], [4]. These methods have proved to be useful in many practical situations, but, they have the disadvantage that they could be trapped in a local minimum, and thus it is hard to prove something about the approximation ratio. The second group consists of discretization patterns, inducing the repeated computation of the semioptimal solution for a dense discrete set of transformations. We will exploit this approach to compute a λ-approximation of the optimal matching, where $\lambda > 1$ is an arbitrary constant.
A common key problem of many approximation problems focusses on the fact the the value of an optimal solution is unknown. Here we are able to derive upper and lower bounds for ϵ_{opt} from the results of some semioptimal matchings. Since each pair (s, s') of characteristic points on S could constitute the approximated destination of (p_1, p_2) we apply this procedure for each such pair. More precisely, we compute the semioptimal matching $t_{s,s'}$ mapping (p_1, p_2) onto the point pair $(\overline{s_1}, \overline{s_2})$, where $(\overline{s_1}, \overline{s_2})$ forms the same line and has the same center as (s, s'), but $||\overline{s_1} - \overline{s_2}|| = ||p_1 - p_2||$. We denote the best value obtained in this way by $\delta = \min_{s,s' \in S_c}\{\epsilon(t_{s,s'})\}$.
Furthermore, we introduce the radius r_P and the relative radius R_P of the point set P with respect to the center of the characteristic points as follows:

$$r_P = \max_{p \in P} ||\frac{p_1 + p_2}{2} - p||, \quad R_P = \frac{r_P}{\frac{||p_1 - p_2||}{2}} = \frac{2\, r_P}{||p_1 - p_2||}.$$

Proposition 1 $\delta \geq \epsilon_{opt} \geq \frac{\delta}{R_P + 2}$.

Given an approximation factor $\lambda > 1$ we try to improve the best value δ obtained so far by small changes of the predefined matching positions $\overline{s_1}, \overline{s_2}$. The bounds above can be used to design a grid based set of pertubed matching positions which is dense enough to include a λ-approximation of an optimal matching.

Let us set two grids around points s and s' in the following manner. First, a 2-dimensional squared grid, centered at the point s', normal to segment (s, s'), with size $2\sqrt{3}\delta$ and with size of the subsquares $\frac{(\lambda-1)\delta}{8(R_P+1)R_P}$. Second, a 3 dimensional grid centered at the point s, parallel to the 2-dimensional grid, with size 2δ and size of subcubes $\frac{\sqrt{3}(\lambda-1)\delta}{6(R_P+1)}$. Using these grids for fixing a dense set of matching positions $(\overline{s_1}, \overline{s_2})$, at least one of the semioptimal matchings $t_{\overline{s_1},\overline{s_2}}$ is a λ-approximation of the optimal matching.
The number of the grid combinations, defining possible matching positions $(\overline{s_1}, \overline{s_2})$, is $(\frac{16\sqrt{3}R_P(R_P+1)}{\lambda-1} + 1)^2 (\frac{\sqrt{3}(R_P+1)}{\lambda-1} + 1)^3$. This implies the following estimation of the total run time:

Lemma 2 *The run time complexity of the λ-approximation presented above is $O(n_c^2\, k\, n \log n\, \frac{R_P^5}{(\lambda-1)^5})$.*

We remark that the factor n in this formula can be improved in the same way as discussed in the analysis of the semioptimal matching. Moreover, in the applications the ratio R_P can be regarded as a small constant.

References

[1] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Trans. on Pattern Recognition and Machine Intelligence*, 14(2):239–256, February 1992.

[2] M. Hagedoorn. Pattern matching using similarity measures. *PhD thesis, Utrecht University*, 2000.

[3] F. Hoffmann, K. Kriegel, S. Schönherr, and C. Wenk. A simple and robust algorithm for landmark registration in computer assisted neurosurgery. *Technical report B 99-21, Freie Universität Berlin*, 1999.

[4] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *Third International Conference on 3D Digital Imaging and Modeling*, pages 145–152, 2001.

[5] M. Sharir and P. Agarwal. Davenport-Schinzel sequences and their geometric applications. *Cambridge University Press*, 1995.

[6] P. J. M. van Laarhoven and E. H. L. Aarts. Simulated Annealing: Theory and Applications. *D.Reidel Publishing Company, Kluwer*, 1987.

Approximation Algorithms for the Earth Mover's Distance Under Transformations Using Reference Points

Oliver Klein* Remco C. Veltkamp†

1 Introduction

Reference points have been introduced in [2] and [1] to construct approximation algorithms for matching compact subsets of $\mathbb{R}^d$ under a given class of transformations. Also a general discussion of reference point methods for matching according to the Hausdorff-distance has been given in [1]. Another distance measure used for shape matching is the Earth Mover's Distance (EMD) for weighted point sets ([7]). Here we will extend the definition of reference points to weighted point sets and get fast constant factor approximation algorithms for matching weighted point sets under translations, rigid motions and similarity operations with respect to the Earth Mover's Distance. A first iterative algorithm to solve this problem has been given by Cohen ([3]). Thus we want to find algorithms where $EMD^{apx}(A,B) \leq \varepsilon EMD^{opt}(A,B)$. Under this assumption ε is called the loss factor of the approximation algorithm. Unless stated otherwise, the results given in this paper are independent of the distance measure on the ground set, therefore the results are widely applicable. Additionally, all theorems hold in arbitrary dimension d. For a full version of this extended abstract see [5].

2 Basic Definitions

In this chapter we provide all definitions required.

Definition 1 (Weighted Point Set) *([4]) Let $A = \{a_1, a_2, ..., a_n\}$ be a weighted point set such that $a_i = (p_i, \alpha_i)$ for $i = 1, ..., n$, where p_i is a point in $\mathbb{R}^d$ and $\alpha_i \in \mathbb{R}^+ \cup \{0\}$ its corresponding weight. Let $W^A = \sum_{i=1}^n \alpha_i$ be the total weight of A. Let $\mathbb{W}^d$ be the set of all weighted point sets in $\mathbb{R}^d$ and $\mathbb{W}^{d,G}$ be the set of all weighted point sets in $\mathbb{R}^d$ with total weight $G \in \mathbb{R}^+$.*

In the following we will use transformations on weighted point sets. By this we mean to transform the point set and leave the weights unchanged.

*Department of Computer Science, FU Berlin, oklein@inf.fu-berlin.de. This research was supported by the Deutsche Forschungsgemeinschaft within the European graduate program 'Combinatorics, Geometry and Computation' (No. GRK 588/2).

†Department of Computer Science, Universiteit Utrecht, remco.veltkamp@cs.uu.nl

A point related to each weighted point set is the center of mass. This point will play an important role in our approximation algorithms. Note that this point can be computed in linear time and therefore does not affect the runtime of the presented algorithms.

Definition 2 (Center of Mass) *The center of mass of a weighted point set $A = \{(p_i, \alpha_i)_{i=1,...,n}\} \in \mathbb{W}^d$ is defined as*

$$C(A) = \frac{1}{W^A} \sum_{i=1}^n \alpha_i p_i.$$

Definition 3 (Reference Point) *([1]) Let $\mathcal{K}$ be a subset of $\mathbb{W}^d$ and $\delta : \mathcal{K} \to \mathbb{R}$ be a distance measure on $\mathcal{K}$. A mapping $r : \mathcal{K} \to \mathbb{R}^d$ is called a δ-reference point for $\mathcal{K}$ with respect to a set of transformations $\mathcal{T}$ on $\mathcal{K}$, if the following two conditions hold:*

1. *Equivariance with respect to $\mathcal{T}$: For all $A \in \mathcal{K}$ and $T \in \mathcal{T}$ we have*
$$r(T(A)) = T(r(A)).$$
2. *Lipschitz-continuity: There is a constant $c \geq 0$, such that for all $A, B \in \mathcal{K}$,*
$$||r(A) - r(B)|| \leq c \cdot \delta(A, B).$$

We call c the quality of the reference point r.

Later, when we want to construct an approximation algorithm for similarities, we will have to rescale at least one of the weighted point sets. Unfortunately, rescaling in a way that the diameters of the underlying point sets are equal, does not work. Please note again that we will keep the weights of the points unchanged.

The key for a working algorithm is to rescale in a way that the normalized first moments with respect to their reference points coincide. Here we give the definition of the normalized first moment of a weighted point set with respect to an arbitrary point $p \in \mathbb{R}^d$. Note that this point can be computed in linear time.

Definition 4 (Normalized First Moment) *The normalized first moment of a weighted point set $A = \{(p_i, \alpha_i)_{i=1,...,n}\} \in \mathbb{W}^d$ with respect to a point $p \in \mathbb{R}^d$ is defined as*

$$m_p(A) = \frac{1}{W^A} \sum_{i=1}^n \alpha_i ||p_i - p||.$$

Next we will introduce the Earth Mover's Distance (EMD,[7]), a commonly known distance measure on weighted point sets.

Definition 5 (Earth Mover's Distance) *Let $A = \{(p_i, \alpha_i)_{i=1,...,n}\}$, $B = \{(q_i, \beta_i)_{i=1,...,m}\} \in \mathbb{W}^d$ be weighted point sets with total weights W^A, W^B. Let $D : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a distance measure on the ground set $\mathbb{R}^d$. The Earth Mover's Distance between A and B is defined as*

$$EMD(A, B) = \frac{\min_{F \in \mathcal{F}} \sum_{i=1}^{n} \sum_{j=1}^{m} f_{ij} D(p_i, q_j)}{\min\{W^A, W^B\}}$$

where $F = \{f_{ij}\}$ is a feasible flow, i.e.

1. $f_{ij} \geq 0, i = 1, ..., n, j = 1, ..., m$
2. $\sum_{j=1}^{m} f_{ij} \leq \alpha_i, i = 1, ..., n$
3. $\sum_{i=1}^{n} f_{ij} \leq \beta_j, j = 1, ..., m$
4. $\sum_{i=1}^{n} \sum_{j=1}^{m} f_{ij} = \min\{W^A, W^B\}$

For a detailed discussion of the EMD see [7], [3] and [4]. In the following, the distance measure D used in the definition of the EMD should be the same as the one used in the defintion of the reference point. If D is the Euclidean Distance, we will also use $EEMD$ as a notation for the Earth Mover's Distance.

3 Approximation Algorithms Using Reference Points

In this chapter we present approximation algorithms using reference points. Since this would be useless if there was no reference point, we provide the following theorem:

Theorem 1 *The center of mass is an EMD-reference point for weighted point sets with equal total weights with respect to affine transformations. Its quality is* 1.

The proof of the Lipschitz-continuity was already given in [7] as a proof for a lower bound for the EMD. The equivariance of the center of mass is commonly known.

The following three sections are organized as follows: In each section we consider a class of transformations, construct an approximation algorithm for matching under these transformations for general reference points and finally use the center of mass to get a concrete algorithm.

For the whole chapter let $A = \{(p_i, \alpha_i)_{i=1,...,n}\}$, $B = \{(q_i, \beta_i)_{i=1,...,m}\} \in \mathbb{W}^{d,G}$ be two weighted point sets in dimension d with positive equal total weight $G \in \mathbb{R}^+$. Please note that the following results do not hold for weighted point sets with unequal total weights. For simplicity let m be $O(n)$. Further let $r : \mathbb{W}^{d,G} \to \mathbb{R}^d$ be an EMD-reference point for weighted point sets with respect to the considered class of transformations with quality c. Let $T^{ref}(n)$ be the time to compute the reference point, $T^{EMD}(n)$ and $T^{EEMD}(n)$ be the time to compute the EMD and $EEMD$ between two weighted point sets and $T^{rot}(n)$ be the time needed to find the optimal rotation around a fixed point. An upper bound on $T^{EMD}(n)$ and $T^{EEMD}(n)$ is $O(n^4 \log n)$ using a strongly polynomial minimum cost flow algorithm by Orlin ([6]). A in practice faster algorithm can be developed by solving the linear programming problem using the simplex method.

3.1 Translations

Consider the following algorithm to get an approximation on the problem of finding a translation minimizing the EMD under translations:

```
Algorithm TranslationApx:
  1. Compute $r(A)$ and $r(B)$ and move $B$ by $r(A) - r(B)$. Let $B'$ be the image of $B$.
  2. Output $B'$ as an approximately optimal solution and the approximate distance $EMD(A, B')$.
```

Extending the proof in [1] to weighted point sets we can prove the following:

Theorem 2 *Algorithm TranslationApx finds an approximately optimal matching for translations with loss factor $c + 1$ in time $O(T^{ref}(n) + T^{EMD}(n))$.*

Corollary 3 *Algorithm TranslationApx using the center of mass as EMD-reference point induces an approximation algorithm with approximation ratio* 2. *Its runtime is $O(T^{EMD}(n))$.*

3.2 Rigid Motions

The following algorithm gives a first approximation on the EMD under rigid motions, i.e. combinations of translations and rotations:

```
Algorithm RigidMotionApx:
  1. Compute $r(A)$ and $r(B)$ and move $B$ by $r(A) - r(B)$. Let $B'$ be the image of $B$.
  2. Find an optimal matching of $A$ and $B'$ under rotations of $B'$ around $r(A) = r(B')$. Let $B''$ be the image of $B'$ under this rotation.
  3. Output $B''$ as an approximately optimal solution together with the approximate distance $EMD(A, B'')$.
```

Theorem 4 *Algorithm RigidMotionApx finds an approximately optimal matching for rigid motions with loss factor $c+1$ in $O(T^{ref}(n) + T^{EMD}(n) + T^{rot}(n))$.*

Since the position of the reference point as rotation center is fixed, several degrees of freedom have been eliminated and the problem is easier than the one finding the optimal rigid motion itself. Unfortunately, even for this problem no efficient algorithm is known so far. Therefore it would be nice to have at least an approximation algorithm for this problem. We will show one based on the following lemma. Please note that this lemma is only proven if we take the Euclidean distance on the ground set.

Lemma 5 *Let $p \in \mathbb{R}^d$ be some point. Let $Rot(p)$ be the set of all rotations around p. Then there is a rotation $R' \in Rot(p)$ such that*

$$EEMD(A, R'(B)) \leq 2 \cdot \min_{R \in Rot(p)} EEMD(A, R(B)),$$

where R' aligns p and at least one point of each set A and B.

Therefore, given a fixed point $p \in \mathbb{R}^d$ we get a 2-approximation on the problem of finding an optimal rotation of B around p by the following algorithm:

```
Algorithm RotationApx
  1. Compute the minimum $EEMD$ over all possible
     alignments of $p$ and at least one point of each set
     $A$ and $B$.
```

The runtime of this algorithm is $O(n^2 T^{EEMD}(n))$. Using this algorithm combined with reference points we now get an easy to implement and fast approximation algorithm for rigid motions. Unfortunately, the increased efficiency must be paid by the increased approximation ratio $(2c+2)$.

```
Algorithm RigidMotionApxUsingRotationApx
  1. Compute $r(A)$ and $r(B)$ and move $B$ by $r(A) - r(B)$.
     Let $B'$ be the image of $B$.
  2. Find a best matching of $A$ and $B'$ under rotations
     of $B'$ around $r(A) = r(B')$ where $r(A)$ and at
     least one point in $A$ and $B'$ are aligned. Let $B''$
     be the image of $B'$ under this rotation.
  3. Output $B''$ as an approximately optimal solution
     together with the approximate distance $EEMD(A, B'')$.
```

Theorem 6 *RigidMotionApxUsingRotationApx finds an approximately optimal matching for rigid motions with loss factor $2c + 2$ in time $O(T^{ref}(n) + n^2 T^{EEMD}(n))$. This holds for the Euclidean distance on the ground set.*

In the next two corollaries we apply the center of mass to the last two theorems:

Corollary 7 *RigidMotionApx using the center of mass as EMD-reference point induces an approximation algorithm with approximation ratio 2 in time $O(T^{rot}(n) + T^{EMD}(n))$.*

Corollary 8 *RigidMotionApxUsingRotationApx using the center of mass as EMD-reference point induces an approximation algorithm with approximation ratio 4. Its runtime is $O(n^2 T^{EEMD}(n))$. This holds for the Euclidean distance on the ground set.*

3.3 Similarities

In this section we present approximation algorithms for matching two given weighted point sets under similarity transformations, i.e. combinations of translations, rotations and scalings. More precisely, we want to compute $\min_S EMD(A, S(B))$, where the minimum is taken over all similarity operations S. Note that exchanging A and B makes a difference.

```
Algorithm SimilarityApx:
  1. Compute $r(A)$ and $r(B)$ and move $B$ by $r(A) - r(B)$.
     Let $B'$ be the image of $B$.
  2. Determine the normalized first moments
     $m_{r(A)}(A)$ and $m_{r(B')}(B')$ and scale $B'$ by
     $\frac{m_{r(A)}(A)}{m_{r(B')}(B')}$ around the center $r(A) = r(B')$.
     Let $B''$ be the image of $B'$ under this scaling.
  3. Find an optimal matching of $A$ and $B''$ under
     rotations of $B''$ around $r(A) = r(B'')$. Let $B'''$
     be the image of $B''$ under this rotation.
  4. Output $B'''$ as an approximately optimal solution
     together with the approximate distance $EMD(A, B''')$.
```

To show the correctness of this algorithm we use the following two lemmata:

Lemma 9 *Let $A \in \mathbb{W}^d$ be a weighted point set with normalized first moment $m_p(A)$ with respect to a point $p \in \mathbb{R}^d$. Let τ_1, τ_2 be scalings with center p and ratios γ_1 and γ_2, respectively. Then*

$$EMD(\tau_1(A), \tau_2(A)) \leq |(\gamma_1 - \gamma_2) m_p(A)|.$$

The next lemma gives a new lower bound for the EMD of two weighted point sets:

Lemma 10 *Let $A, B \in \mathbb{W}^{d,G}$ and $r : \mathbb{W}^{d,G} \to \mathbb{R}^d$ a reference point with quality c. Then*

$$|m_{r(A)}(A) - m_{r(B)}(B)| \leq (1 + c) EMD(A, B).$$

Using these lemmata we can prove the following:

Theorem 11 *SimilarityApx finds an approximately optimal matching for similarities with loss factor $2c+2$ in time $O(T^{ref}(n) + T^{EMD}(n) + T^{rot}(n))$.*

As for *RigidMotionApx*, *SimilarityApx* depends on finding the optimal rotation, which is impractical. Again, we make this algorithm practical and efficient by using *RotationApx* and again we have to pay by a worse approximation ratio:

Algorithm *SimilarityApxUsingRotationApx*

1. Compute $r(A)$ and $r(B)$ and move B by $r(A) - r(B)$. Let B' be the image of B.
2. Determine the normalized first moments $m_{r(A)}(A)$ and $m_{r(B')}(B')$ and scale B' by $\frac{m_{r(A)}(A)}{m_{r(B')}(B')}$ around the center $r(A) = r(B')$. Let B'' be the image of B' under this scaling.
3. Find a best matching of A and B'' under rotations of B'' around $r(A) = r(B'')$ where $r(A)$ and at least one point in each set A and B'' are aligned. Let B''' be the image of B'' under this rotation.
4. Output B''' as an approximately optimal solution and the approximate distance $EEMD(A, B''')$.

Theorem 12 *Algorithm SimilarityApxUsingRotationApx finds an approximately optimal matching for similarities with loss factor* $4c + 4$ *in time* $O(T^{ref}(n) + n^2 T^{EEMD}(n))$. *This holds for the Euclidean distance on the ground set.*

Corollary 13 *Algorithm SimilarityApx using the center of mass as EMD-reference point induces an approximation algorithm with approximation ratio* 4. *Its runtime is* $O(T^{EMD}(n) + T^{rot}(n))$.

Corollary 14 *Algorithm SimilarityApxUsingRotationApx using the center of mass as EMD-reference point induces an approximation algorithm with loss factor* 8. *Its runtime is* $O(n^2 T^{EEMD}(n))$. *This holds for the Euclidean distance on the ground set.*

3.4 Lower Bound for Algorithm TranslationApx

In Section 3.1 we presented the center of mass as an EMD-reference point with quality 1, and thus inducing an approximation algorithm for translations with ratio 2. We now show that this bound is tight:

Theorem 15 *There are sets where the upper bound for algorithm TranslationApx is assumed.*

Proof. Let $A := \{((0,0),1),((1,0),K)\}$ and $B := \{((0,0),1),((0,1),K)\}$, where $K \in \mathbb{R}^+$ is some constant. Let $EMD^C(A,B)$ be the Earth Mover's Distance, where the center of masses coincide and $EMD^{opt}(A,B)$ be the optimal distance under translation. Then $\frac{EMD^C(A,B)}{EMD^{OPT}(A,B)} \to 2$ as $K \to \infty$. This can be seen easily by using an upper bound for $EMD^{opt}(A,B)$ by matching the two thick points. □

4 Conclusion

In this paper we introduced EMD-reference points for weighted point sets and constructed efficient approximation algorithms for matching under various classes of transformations. Additionally, we presented the center of mass as an EMD-reference point for weighted point sets with equal total weights. Unfortunately, the center of mass is no EMD-reference point if you consider the set of all weighted point sets, including those with different total weights. Even worse, we show in [5] that there is no EMD-reference point for all weighted point sets. A variation of the EMD is the Proportional Transportation Distance (PTD), see [4]. In [5] we also show, that the center of mass is a PTD-reference point even for weighted point sets with different total weight and all theorems and corollaries mentioned in this paper carry over. But the PTD has a couple of disadvantages against the EMD, for example it is not suitable for partial matching applications.

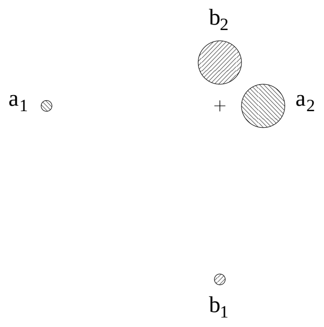


Figure 1: Matching according to center of mass

References

[1] H. Alt, O. Aichholzer, G. Rote. Matching Shapes with a Reference Point. In *Proc. 10th Annual Symposium on Computational Geometry*, pages 85–92, 1994.

[2] H. Alt, B. Behrends, J. Blömer. Approximate Matching of Polygonal Shapes. In *Proc. 7th Ann. Symp. on Comp. Geometry*, pages 186–193, 1991.

[3] S. Cohen. Finding Color and Shape Patterns in Images. PhD thesis, Stanford University, Department of Compute Science, 1999.

[4] P. Giannopoulos, R. Veltkamp. A pseudo-metric for weighted point sets. In *Proc. 7th European Conf. Comp. Vision*, LNCS 2352, pages 715–731, 2002.

[5] O. Klein, R. C. Veltkamp. Approximation Algorithms for the Earth Mover's Distance Under Transformations Using Reference Points. Technical Report UU-CS-2005-003, http://ftp.cs.uu.nl/pub/RUU/CS/techreps/CS-2005/2005-003.pdf, 2005.

[6] J. B. Orlin. A Faster Strongly Polynomial Minimum Cost Flow Algorithm. In *Operations Research*, vol.41,no.2, pages 338–350, 1993.

[7] Y. Rubner, C. Tomasi, L. J. Guibas. The Earth Mover's Distance as a Metric for Image Retrieval. In *Int. J. of Comp. Vision 40(2)*, pages 99–121, 2000.

Matching Point Sets with respect to the Earth Mover's Distance

Sergio Cabello* Panos Giannopoulos† Christian Knauer† Günter Rote†

Abstract

Let A and B be two sets of m resp. n weighted points in the plane, with $m \leq n$. We present $(1+\epsilon)$ and $(2+\epsilon)$-approximation algorithms for the minimum Euclidean Earth Mover's Distance between A and B under translations and rigid motions respectively. In the general case where the sets have unequal total weights the algorithms run in $O((n^3 m/\epsilon^4)\log^2(n/\epsilon))$ time for translations and $O((n^4 m^2/\epsilon^4)\log^2(n/\epsilon))$ time for rigid motions. When the sets have equal total weights, the respective running times decrease to $O((n^2/\epsilon^4)\log^2(n/\epsilon))$ and $O((n^3 m/\epsilon^4)\log^2(n/\epsilon))$. We also show how to compute a $(1+\epsilon)$ and $(2+\epsilon)$-approximation of the minimum cost Euclidean bipartite matching under translations and rigid motions in $O((n^{3/2}/\epsilon^{7/2})\log^5 n)$ and $O((n/\epsilon)^{7/2}\log^5 n)$ time respectively.

1 Introduction

Let $A = \{a_1, \ldots, a_m\}$ and $B = \{b_1, \ldots, b_n\}$ be two planar weighted point sets with $m \leq n$. A weighted point $a_i \in A$ is defined as $a_i = \{(x_{a_i}, y_{a_i}), w_i\}, i = 1,..,m,$ where $(x_{a_i}, y_{a_i}) \in \mathbb{R}^2$ and $w_i \in \mathbb{R}^+ \cup \{0\}$ is its weight. A weighted point $b_j \in B$ is defined similarly as $b_j = \{(x_{b_j}, y_{b_j}), u_j\}, j = 1,..,n$. Let $W = \sum_{i=1}^m w_i$ and $U = \sum_{j=1}^n u_j$ be the total weight, or simply weight, of A and B respectively.

The Earth Mover's Distance (EMD) is a similarity measure for weighted point sets with applications in colour-based image retrieval [7], shape matching [7, 2, 1] and music score matching [9]. In a typical scenario, a pattern is reduced to a set of feature weighted points; the larger the weight, the more important the point for the whole pattern. Informally, a weighted point a_i can be seen as an amount (supply) of earth or mass, equal to w_i units, positioned at (x_{a_i}, y_{a_i}); alternatively it can be taken as an empty hole (demand) of w_i units of earth capacity. We assign arbitrarily the role of the supplier to A and that of the receiver/demander to B, setting, in this way, a direction of earth transportation. The Earth Mover's Distance (EMD) of A to B measures the minimum amount of work needed to fill the holes with earth. A formal definition of the EMD will be given shortly.

In order to measure the similarity of two sets A and B independently of transformations, one wants to find a transformed version of, say, A that attains the minimum possible distance to B. In this paper we are interested in transformations that change only the position of the points, not their weights; in particular, we focus on translations and rigid motions – sometimes referred to as isometries. We consider B to be fixed, while A can be translated and/or rotated relative to B. We assume some initial positions for both sets, denoted simply by A and B. Let $\mathcal{I}$ be the set of all possible rigid motions in the plane. We denote by R_θ a rotation about the origin by some angle $\theta \in [0, 2\pi)$ and by $T_{\vec{t}}$ a translation by some $\vec{t} \in \mathbb{R}^2$. Any rigid motion $I \in \mathcal{I}$ can be uniquely defined as a translation followed by a rotation, that is, $I = I_{\vec{t},\theta} = R_\theta \circ T_{\vec{t}}$, for some $\theta \in [0, 2\pi)$ and $\vec{t} \in \mathbb{R}^2$. In general, transformed versions of A are denoted by $A(\vec{t}, \theta) = \{a_1(\vec{t}, \theta), \ldots, a_m(\vec{t}, \theta)\}$ for some $I_{\vec{t},\theta} \in \mathcal{I}$. For simplicity, translated only versions of A are denoted by $A(\vec{t}) = \{a_1(\vec{t}), \ldots, a_m(\vec{t})\}$.

The EMD between $A(\vec{t}, \theta)$ and B, is a function $\mathrm{EMD} : \mathcal{I} \to \mathbb{R}^+ \cup \{0\}$ defined as

$$\mathrm{EMD}(\vec{t}, \theta) = \min_{F \in \mathcal{F}(A,B)} \frac{\sum_{i=1}^m \sum_{j=1}^n f_{ij} d_{ij}(\vec{t}, \theta)}{\min\{W, U\}},$$

where $d_{ij}(\vec{t}, \theta)$ is the distance of $a_i(\vec{t}, \theta)$ to b_j, and $F = \{f_{ij}\} \in \mathcal{F}(A, B)$ with $\mathcal{F}(A, B)$ being the set of all feasible flows between A and B defined by the constraints: (i)$f_{ij} \geq 0, i = 1, ..., m, j = 1, ..., n,$ (ii)$\sum_{j=1}^n f_{ij} \leq w_i, i = 1, ..., m,$ (iii)$\sum_{i=1}^m f_{ij} \leq u_j, j = 1, ..., n,$ and (iv)$\sum_{i=1}^m \sum_{j=1}^n f_{ij} = \min\{W, U\}$. In case that $\vec{t}$ or θ or both are constant, we simply write $\mathrm{EMD}(\theta)$, $\mathrm{EMD}(\vec{t})$ and EMD respectively. The EMD is a metric when d_{ij} is a metric and $W = U$ [7]. When $W \neq U$ the EMD inherently performs partial matching since a portion of the weight of the 'heavier' set remains unmatched. We deal with the Euclidean EMD where d_{ij} is given by the L_2-norm. For simplicity, and without loss of generality, we assume that $\min\{W, U\} = 1$. We study the following problem:

Given two weighted point sets A, B compute a rigid motion $I_{\vec{t}_{opt},\theta_{opt}}$ that minimizes $\mathrm{EMD}(\vec{t}, \theta)$.

This problem was first studied by Cohen [7] who

*IMFM, Department of Mathematics, Jadranska 19, SI-1000 Ljubljana, Slovenia, sergio.cabello@imfm.uni-lj.si; partially supported by the European Community Sixth Framework Programme under a Marie Curie Intra-European Fellowship.

†Institut für Informatik, Freie Universität Berlin, Takusstraße 9, D-14195 Berlin, Germany, {panos, knauer, rote}@inf.fu-berlin.de

presented a Flow–Transformation iteration which alternates between finding the optimum flow for a given transformation, and the optimum transformation for a given flow. They showed that this iterative procedure converges, but not neccessarily to the global optimum. Computing the EMD for a given transformation is actually the transportation problem, a special minimum cost network flow problem for the solution of which there is a variety of polynomial time algorithms [3, 4]. However, as we discuss later on, the task of finding the optimal transformation for a given flow is not trivial. Cohen also gave simple algorithms that compute the optimum translation for the special case where $W = U$ and d_{ij} is the squared Euclidean distance. This case is quite restrictive since, in general, the sets need not have the same weight, and the use of squared Euclidean distance is statistically less robust than Euclidean distance [5]. Currently, no algorithm that computes the optimal translation and/or rotation is known for the Euclidean EMD.

Observe that the objective function is not linear in $\vec{t}$ and θ but it is still linear in F. Thus, the minimum EMD occurs at some vertex of the convex polytope $\mathcal{F}(A, B)$. This suggests the following straightforward algorithm: for every vertex $F = \{f_{ij}\}$ of $\mathcal{F}(A, B)$ compute the optimal rigid motion, i.e., the one that minimizes $\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij} d_{ij}(\vec{t}, \theta)$. For translations, the latter problem reduces to the $Fermat - Weber$ problem where one wants to find a point that minimizes the sum of weighted distances to a set of given points. No exact solution to this problem is known even in the real RAM model of computation [5]. However, Bose et al. [5] gave a $O(n \log n)$-time $(1 + \epsilon)$-approximation algorithm for any fixed dimension. Using their algorithm for every vertex of $\mathcal{F}(A, B)$ gives only a $(1 + \epsilon)$-approximation of the minimum EMD under translations in exponential time.

In this paper, we give simple polynomial-time algorithms that achieve a $(1+\epsilon)$ and $(2+\epsilon)$-approximation for translations and rigid motions respectively.

2 Lower bounds and approximations of the EMD

First, we give two simple lower bounds on the EMD that are vital for the approximation algorithms given in the next sections. In these algorithms we need to compute the EMD for a given transformation. Computing the EMD exactly is expensive, and unecessary since we opt for approximations of the minimum EMD under transformations. We show how to get a $(1+\epsilon)$-approximation of the EMD in almost quadratic time.

The following lower bound comes directly from the definition of the EMD.

Observation 1 *Given two weighted point sets A and B,* $\text{EMD} \geq \min_{i,j} d_{ij}$.

The next lower bound is due to Cohen [7]. The *center of mass* $C(A)$ of a planar weighted point set $A = \{(x_{a_i}, y_{a_i}), w_i\}, i = 1, \ldots, m$ is defined as $C(A) = \left(\sum_{i=1}^{m} w_i(x_{a_i}, y_{a_i})\right) / \sum_{i=1}^{m} w_i$.

Theorem 1 *[7, Theorem 6] Let A and B be two weighted point sets with equal weights. Then* $\text{EMD} \geq |C(A) - C(B)|$.

Currently, the fastest strongly polynomial-time algorithm for the minimum cost flow problem on a graph $G(V, E)$ is due to Orlin [3], and runs in $O((|E| \log |V|)(|E| + |V| \log |V|))$ time. Using the algorithm of Callahan and Kosaraju [6], we can construct, in $O(n \log n + (n/\epsilon^2) \log 1/\epsilon)$ time, a linear size $(1 + \epsilon)$-spanner G_s, i.e., a graph $G_s(V, E')$ with $|E'| = O(n/\epsilon)$ such that the shortest path between any two points in G_s is at most $(1 + \epsilon)$ times the Euclidean distance of the points. Running the algorithm of Orlin on G_s produces an approximate value EMD_s such that $\text{EMD} \leq \text{EMD}_s \leq (1 + \epsilon)\text{EMD}$ in $O((n^2/\epsilon^2) \log^2(n/\epsilon))$ time; we refer to this procedure as $\textsc{ApxEMD}(A, B, \epsilon)$.

Lemma 2 *For any given $\epsilon > 0$, a value EMD_s with $\text{EMD} \leq \text{EMD}_s \leq (1 + \epsilon)\text{EMD}$ can be computed in $O((n^2/\epsilon^2) \log^2(n/\epsilon))$ time.*

Next, consider the case where $|A| = |B| = n$ and $w_i = u_j = 1, i = 1, ..., n, j = 1, ..., n$. The integer solutions property of the minimum cost flow problem and the fact that $0 \leq f_{ij} \leq 1$ imply that there is a minimum cost flow on G that results in a (perfect) matching between A and B. Hence, we can restrict ourselves to finding a minimum cost matching–usually called the *assignment* problem. Varadarajan and Agarwal [8] presented an algorithm that finds a matching with cost at most $(1+\epsilon)$ times that of an optimal matching in $O((n/\epsilon)^{3/2} \log^5 n)$ time; we refer to this algorithm as $\textsc{ApxMATCH}(A, B, \epsilon)$.

3 Approximation algorithms for translations

We denote by $\vec{t}_{i \to j}$ the translation which matches a_i and b_j; we call such a translation a *point-to-point* translation. Observation 1 implies that the point-to-point translation that is closest to $\vec{t}_{opt}$ gives a 2-approximation of $\text{EMD}(\vec{t}_{opt})$. Hence, we have the following:

Lemma 3 *Given two weighted point sets A and B,* $\text{EMD}(\vec{t}_{opt}) \leq \min_{i,j} \text{EMD}(\vec{t}_{i \to j}) \leq 2\text{EMD}(\vec{t}_{opt})$.

According to Observation 1, the point-to-point translation which is closest to $\vec{t}_{opt}$ can be at most $\text{EMD}(\vec{t}_{opt})$ away from $\vec{t}_{opt}$. This bound is crucial for the $(1 + \epsilon)$-approximation algorithm given in Figure 1. Using a uniform square grid of suitable size we compute the EMD for a limited number of grid translations within a small neighborhood – of size $\text{EMD}(\vec{t}_{opt})$ – of every point-to-point translation. Note that we do not know $\text{EMD}(\vec{t}_{opt})$ but we can compute $\min_{i,j}\text{EMD}(\vec{t}_{i\to j})$ which, according to Lemma 3, approximates $\text{EMD}(\vec{t}_{opt})$ well-enough. In order to save time, rather than computing EMD exactly, we will approximate it using the procedure ApxEMD.

```
Translation($A, B, \epsilon$):
  1. Let $\alpha = \min_{i,j}\text{ApxEMD}(A(\vec{t}_{i\to j}), B, 1)$ and let $G$
     be a uniform square grid of spacing $c\epsilon\alpha$, where
     $c = 1/\sqrt{72}$.
  2. For each pair of points $a_i \in A$ and $b_j \in B$ do:
     (a) Place a disk $D$ of radius $\alpha$ around $\vec{t}_{i\to j}$.
     (b) For every grid point $\vec{t}_g \in D \cap G$ compute a
         value $\widetilde{\text{EMD}}(\vec{t}_g) = \text{ApxEMD}(A(\vec{t}_g), B, \epsilon/3)$.
  3. Report the grid point $\vec{t}_{apx}$ that minimizes
     $\widetilde{\text{EMD}}(\vec{t}_g)$.
```

Figure 1: Algorithm Translation(A, B, ϵ).

Theorem 4 *For any given $\epsilon > 0$, a translation $\vec{t}_{apx}$ such that $\text{EMD}(\vec{t}_{apx}) \le (1+\epsilon)\text{EMD}(\vec{t}_{opt})$ can be computed in $O((n^3m/\epsilon^4)\log^2(n/\epsilon))$ time.*

Next, consider the case of equal weight sets. Let $\vec{t}_{C(A)\to C(B)}$ be the translation that matches the centers of mass $C(A)$ and $C(B)$. Theorem 1 suggests the following trivial 2-approximation algorithm: compute $\text{EMD}(\vec{t}_{C(A)\to C(B)})$. According to Theorem 1, $\vec{t}_{opt}$ is at most $\text{EMD}(\vec{t}_{opt})$ far away from $\vec{t}_{C(A)\to C(B)}$. Hence, we need to search for $\vec{t}_{opt}$ only within a small neighborhood of $\vec{t}_{C(A)\to C(B)}$. We modify algorithm Translation(A, B, ϵ) as follows: First we compute $C(A)$ and $C(B)$. Then, we run $\text{ApxEMD}(A(\vec{t}_{C(A)\to C(B)}), B, 1)$ and set α to the value returned. Next, we run $\text{ApxEMD}(A(\vec{t}_g), B, \epsilon/3)$ for all the grid points $\vec{t}_g$ which are at most α away from $\vec{t}_{C(A)\to C(B)}$. The minimum over all these approximations gives the desired approximation bound. Hence, we have managed to save an nm term from the time bound of Theorem 4.

Theorem 5 *If A and B have equal total weights then, for any given $\epsilon > 0$, a translation $\vec{t}_{apx}$ such that $\text{EMD}(\vec{t}_{apx}) \le (1+\epsilon)\text{EMD}(\vec{t}_{opt})$ can be computed in $O((n^2/\epsilon^4)\log^2(n/\epsilon))$ time.*

For the assigment problem under translations, we can use the above algorithm for equal weight sets , running ApxMATCH instead of ApxEMD. This reduces the running time further.

Theorem 6 *For any given $\epsilon > 0$, a $(1+\epsilon)$-approximation of the minimum cost assignment under translations can be computed in $O((n^{3/2}/\epsilon^{7/2})\log^5 n)$ time.*

Note that, the latter algorithm can be also applied to equal weight sets with bounded integer point weights by replacing each point by as many points as its weight.

4 Approximation algorithms for rigid motions

We first give a $(2+\epsilon)$-approximation algorithm for rotations. Then, we combine this $(2+\epsilon)$-approximation algorithm with the $(1+\epsilon)$-approximation algorithms for translations to get $(2+\epsilon)$-approximation algorithms for rigid motions.

Rotations. Let $a_i\widehat{o}b_j$ be the angle between the segments $\overline{oa_i}$ and $\overline{ob_j}$ such that $0 \le a_i\widehat{o}b_j \le \pi$. Also, let $\theta_{i\to j}$ be the rotation of a_i by $a_i\widehat{o}b_j$ that aligns the origin o and points a_i and b_j such that both a_i and b_j are on the same side of o. Note that this is the rotation that minimizes $d_{ij}(\theta)$; we call such a rotation an *alignment rotation.* We have the following simple lemma.

Lemma 7 *Let a_i and b_j be two points in the plane with $a_i\widehat{o}b_j = \phi$. If a_i is rotated by an angle $\theta \le \phi$, then $d_{ij}(\theta) < 2d_{ij}$.*

Similarly to Lemma 3, and using Lemma 7, we can prove that the alignment rotation that is closest to θ_{opt} gives a 2-approximation of $\text{EMD}(\theta_{opt})$. Hence, we have the following:

Lemma 8 *Given two weighted point sets A and B, $\text{EMD}(\theta_{opt}) \le \min_{i,j}\text{EMD}(\theta_{i\to j}) \le 2\text{EMD}(\theta_{opt})$.*

By approximating $\text{EMD}(\theta_{i\to j})$ with ApxEMD or ApxMATCH we can get a $(2+\epsilon)$-approximation of $\text{EMD}(\theta_{opt})$. We call this algorithm Rotation(A, B, ϵ); from the context it will be always clear whether ApxEMD or ApxMATCH is used. Apart from the cost value, Rotation returns the corresponing rotation $\theta_{i\to j}$ as well.

Lemma 9 *For any given $\epsilon > 0$, a rotation θ_{apx} such that $\text{EMD}(\theta_{apx}) \le (2+\epsilon)\text{EMD}(\theta_{opt})$ can be computed in $O((n^3m/\epsilon^2)\log^2(n/\epsilon))$ time. For the minimum cost assignment problem under rotations the same approximation can be computed in $O((n^{7/2}/\epsilon^{3/2})\log^5 n)$ time.*

Rigid Motions. We can combine the algorithms implied by Lemma 3 and Lemma 9 to get a $(4+\epsilon)$-approximation of the minimum EMD under rigid motions in the following way: for each point-to-point translation $\vec{t}_{i\to j}$, compute a $(2+\epsilon)$-approximation of the optimum EMD between $A(\vec{t}_{i\to j})$ and B under rotations about b_j. The minimum over all these approximations gives a $2(2+\epsilon)$-approximation of $\mathrm{EMD}(\vec{t}_{opt}, \theta_{opt})$.

Lemma 10 *For any given $\epsilon > 0$, a $(4+\epsilon)$-approximation of the minimum EMD under rigid motions can be computed in $O((n^4m^2/\epsilon^2)\log^2(n/\epsilon))$ time.*

The $(2+\epsilon)$-approximation algorithm for rigid motions is based on similar ideas. Accoring to Observation 1, there exist two points a_i, b_j whose distance at $I_{\vec{t}_{opt},\theta_{opt}}$ is at most $\mathrm{EMD}(\vec{t}_{opt}, \theta_{opt})$. We place a grid of suitable size around each $\vec{t}_{i\to j}$. For each grid point $\vec{t}_g$ that is at most $\mathrm{EMD}(\vec{t}_{opt}, \theta_{opt})$ away from $\vec{t}_{i\to j}$ we compute a $(2+\epsilon)$-approximation of the optimum EMD betwenn $A(\vec{t}_g)$ and B under rotations about b_j. The minimum over all these approximations is within a factor of $(2+\epsilon)$ of $\mathrm{EMD}(\vec{t}_{opt}, \theta_{opt})$. Since we do not know $\mathrm{EMD}(\vec{t}_{opt}, \theta_{opt})$, we first compute a $(4+\epsilon)$-approximation of it as shown above. Algorithm RigidMotion(A, B, ϵ) is shown in Figure 2.

```
RigidMotion$(A, B, \epsilon)$:
  1. For each pair of points $a_i \in A$ and $b_j \in B$ do:
     (a) Set the center of rotation, i.e. the origin, to be $b_j$ by translating $B$ appropriately.
     (b) Run Rotation$(A(\vec{t}_{i\to j}), B, 1)$ and let $\alpha_{ij}$ the cost value returned.
     Let $\alpha = \min_{ij} \alpha_{ij}$.
  2. Let $G$ be a uniform grid of spacing $c\alpha\epsilon$, where $c$ is a suitable constant. For each pair of points $a_i \in A$ and $b_j \in B$ do:
     (a) Set the center of rotation, i.e. the origin, to be $b_j$ by translating $B$ appropriately.
     (b) Place a disk $D$ of radius $\alpha$ around $\vec{t}_{i\to j}$.
     (c) For every grid point $\vec{t}_g \in D \cap G$ run Rotation$(A(\vec{t}_g), B, \epsilon/3)$ Let $\widetilde{\mathrm{EMD}}(\vec{t}_g)$ and $\theta^g_{apx}$ be the cost value and angle returned respectively.
  3. Report the grid point $\vec{t}_{apx}$ that minimizes $\widetilde{\mathrm{EMD}}(\vec{t}_g)$, and the corresponding angle $\theta_{apx}$.
```

Figure 2: Algorithm RigidMotion(A, B, ϵ).

Theorem 11 *For any given $\epsilon > 0$, a rigid motion $I_{\vec{t}_{apx},\theta_{apx}}$ such that $\mathrm{EMD}(\vec{t}_{apx}, \theta_{apx}) \leq (2+\epsilon)\mathrm{EMD}(\vec{t}_{opt}, \theta_{opt})$ can be computed in $O((n^4m^2/\epsilon^4)\log^2(n/\epsilon))$ time.*

As in the case of translations, for equal weight sets we need to search for the optimal translation only around $\vec{t}_{C(A)\to C(B)}$. We set the center of rotation to be $C(B)$. Computing the 6-approximation of $\mathrm{EMD}(\vec{t}_{opt}, \theta_{opt})$ can be done simply by running Rotation$(A(\vec{t}_{C(A)\to C(B)}), B, 1)$. Similarly, we need to run Rotation$(A(\vec{t}_g), B, \epsilon/3)$ only for grid points $\vec{t}_g$ that are close to $\vec{t}_{C(A)\to C(B)}$.

Theorem 12 *If A and B have equal total weights, then, for any given $\epsilon > 0$, a rigid motion $I_{\vec{t}_{apx},\theta_{apx}}$ such that $\mathrm{EMD}(\vec{t}_{apx}, \theta_{apx}) \leq (2+\epsilon)\mathrm{EMD}(\vec{t}_{opt}, \theta_{opt})$ can be computed in $O((n^3m/\epsilon^4)\log^2(n/\epsilon))$ time. For the minimum cost assignment problem under rigid motions the same approximation can be computed in $O((n/\epsilon)^{7/2}\log^5 n)$ time.*

References

[1] K. Grauman and T. Darell. Fast contour matching using approximate Earth Mover's Distance. In *Proc. of the IEEE Conf. Comp. Vision and Pattern Recognition*, pages I: 220-227, 2004.

[2] P. Giannopoulos and R. C. Veltkamp. A pseudometric for weighted point sets. In *Proc. of the 7th European Conf. Comp. Vision*, vol. 2352 of *LNCS*, pages 715–731, 2002.

[3] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms and Applications.* Prentice-Hall, 1993.

[4] D.S. Atkinson and P.M. Vaidya. Using geometry to solve the transportation problem in the plane. *Algorithmica*, 13:442–461, 1995.

[5] P. Bose, A. Maheshwari, and P. Morin. Fast approximations for sums of distances clutering and the Fermat–Weber problem. *Comput. Geom. Theory and Appl.*, 24:135–146, 2003.

[6] P. B. Callahan and S. R. Kosaraju. Faster algorithms for some geometric graph problems in higher dimensions. In *Proc. of the 4h Ann. ACM-SIAM Sympos. Discr. Algorithms*, pages 291–300, 1993.

[7] S. Cohen. *Finding Color and Shape Patterns in Images.* PhD thesis, Stanford University, Department of Computer Science, 1999.

[8] K. R. Varadarajan and P. K. Agarwal. Approximation algorithms for bipartite and non-bipartite matching in the plane. In *Proc. of the 10th Ann. ACM-SIAM Sympos. Discr. Algorithms*, pages 805–814, 1999.

[9] R. Typke, P. Giannopoulos, R. C. Veltkamp, F. Wiering, and R. van Oostrum. Using transportation distances for measuring melodic similarity. In *Proc. of the 4th Int. Conf. Music Inf. Retrieval* , pages 107–114, 2003.

Abstract Order Type Extension and New Results on the Rectilinear Crossing Number - Extended Abstract

Oswin Aichholzer[*] Hannes Krasser[†]

Abstract

We provide a complete data base of all realizable order types of 11 points in general position in the plane. Moreover, we develop a novel and efficient method for complete extension to (abstract) order types of size 12 and more. With our approach we have been able to determine the exact rectilinear crossing number for up to $n = 17$, and slightly improved the asymptotic upper bound. We briefly discuss further applications of this approach.

1 Introduction

A finite point set in the plane belongs to the most common ingredients for computational and combinatorial geometry problems. For quite many, especially combinatorial problems, the exact metric properties are not relevant, but the combinatorial properties of the underlying point set play the main role. More precisely, the crossing properties of the line segments spanned by the point set already determine the problem. Triangulations, crossing numbers, convexity problems are among other famous examples. Order types provide a means to encode the combinatorial properties of finite point sets. The order type of a point set $S = \{p_1, .., p_n\}$ is a mapping that assigns to each ordered triple (p_i, p_j, p_k) an orientation. Throughout this work we assume that S is in general position, that is, the orientation of each point triple is either clockwise or counter-clockwise. Two point sets S_1, S_2 are of the same order type if and only if there is a bijection between S_1 and S_2 such that either all (or none) corresponding triples are of equal orientation.

To achieve results for point sets of fixed size for the problems mentioned above, it is sufficient to check one instance of each order type instead of looking at all (infinitely many) point sets. A data base containing all order types of size up to 10 already exists [2] and has been applied fruitfully to many problems in computational and combinatorial geometry [6].

Order types have played a crucial role in gathering knowledge about crossing numbers. The crossing number of a graph G is the least number of edge crossings attained by a drawing of G in the plane. We consider the problem of finding the rectilinear (edges are required to be straight line segments) crossing number $\overline{cr}(K_n)$ of the complete graph K_n on n vertices [12]. Determining $\overline{cr}(K_n)$ is commonly agreed to be a difficult task, see [3] for references and details. So far the exact values of $\overline{cr}(K_n)$ have been known for $n \leq 12$ [2, 3]. In Section 4 we extend this range to $n \leq 17$. Moreover, we also present an improvement on the asymptotic upper bound of $\overline{cr}(K_n)$. Our results are available on-line [1]. We close with a brief discussion of further applications of our approach.

2 Order type data base for n=11

A complete data base of order types for sets with up to 10 points has already been established [2]. We present an extension to this data base for point sets of size 11. Our approach is strongly related to [2] and uses improved techniques to cover the following three steps, cf. [2] for the necessary concepts and definitions.

1. Generating a complete candidate list of abstract order types
2. Grouping abstract order types into projective classes and deciding realizability
3. Realizing all realizable order types by point sets with "nice" coordinate representation

For the first step, we acquired 2 343 203 071 inequivalent abstract order types. We only stored one representative of each projective class explicitly at this time. This evaluates to 41 848 591 abstract projective order types of size $n = 11$, see Table 1.

The second step - deciding realizability - is the hardest part of the construction. The trouble is, that this decision problem is known to be NP-hard [13] and no practical algorithms are known, not even for small sets, say of size 10 or 11. We tried to find realizations and started by applying refined versions of the heuristic methods from [2] for each projective order type class. These worked for most of the abstract order types in question. For classifying non-realizable order types, we used a well-known practical algorithm for a non-realizability proof developed by Bokowski and Richter [9]. To our benefit, the heuristics for finding realizations and proving non-realizability were suffi-

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

[†]Institute for Theoretical Computer Science, Graz University of Technology, Graz, Austria, hkrasser@igi.tugraz.at

cient to completely settle the case for $n = 11$, see Table 1.

The main goal of the third step is to store the data base in an application friendly way. To this end, we provide two representations of the data base. An explicit version of the data base contains one point set for each planar order type, all in 16-bit integer representation.

projective abstract o.t.	41 848 591
− thereof non-realizable	155 214
= projective order types	41 693 377
abstract order types	2 343 203 071
− thereof non-realizable	8 690 164
= order types	2 334 512 907

Table 1: Number of order types of cardinality $n = 11$.

Supporting the reliability in the construction of our data base, all algorithms to generate the complete data base of abstract order types are of purely combinatorial nature. The applied methods for deciding realizability are heuristics, but the acquired results can be checked in a deterministic way.

The vast storage and the lack of applicability are the two main reasons - apart from calculation time - that we do not have a complete data base of order types with 12 or more points.

3 Complete abstract point extension

For several problems and conjectures the complete order type data base of sets of up to 11 points has been sufficient to give a final answer, cf. [3]. However, many problems tend to be harder and cannot be settled just by checking all cases for size up to 11. Still it looks highly plausible to gain significantly more insight with a few additional points, say 12 or 13 points. To evade these obstacles we make use of well-known theoretical results. For many problems on point sets there exist inductive restrictions, so-called subset properties.

Definition 1 (**Subset property**) *Let S_n be an order type consisting of n elements, $n \geq 4$, and consider some property that is valid for S_n. Then this property is called a subset property if and only if there exists some $S_{n-1} \subset S_n$ of $n-1$ elements such that a similar property holds for S_{n-1}.*

Our general idea is to exploit subset properties for order type based problems to obtain results beyond point sets of size 11. First, we are applying the order type data base to completely determine the problem for point sets of small size, that is, up to $n = 11$. This gives a set of result order types of cardinality 11, all realized by point sets. Next, we enumerate all order types of size 12 that contain one of the 11-point result order types as a subset. Applying the subset property, we are able to filter these 12-point order types. Only order types that fulfill the subset property are kept. Then we repeat this procedure, theoretically extending the set of result order types to arbitrary n.

For this technique, we require an algorithm that calculates for a given order type of cardinality n all $(n+1)$-point order types that contain the input order type as a sub-order type. We call this step complete point extension. It is well known that an extension technique relying only on the geometric realizations of the data base cannot guarantee completeness of the extension, see Figure 1. For a specific n-point realization of an order type we cannot derive all required $n+1$ order types just by adding a new point to this realization. To achieve completeness of the extension, we use an abstract extension method, that is, applying a combinatorial extension technique. We provide a one-element extension to an abstract order type by adding a pseudoline to the dual pseudoline arrangement in all combinatorially possible ways.

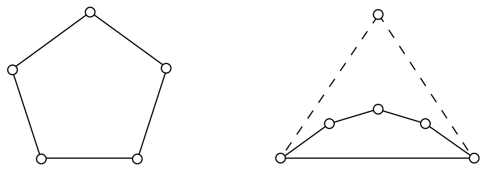


Figure 1: Two realizations of the order type of five points in convex position. Only the right point set can be extended in a way such that the resulting point set has three points on its convex hull.

For specific applications with a subset property, we define an order type extension graph. In this graph each order type is represented by a node. For each order type of size $n + 1$ (son), there is exactly one connection by an edge to a predecessor sub-order type of size n (father). By this definition we have that each order type corresponds to a unique predecessor order type by removal of a single point. On the other hand, an extension process that only extends corresponding to the edges of an order type extension graph (from father to son) enumerates each extended order type exactly once.

In general, the algorithm of complete abstract point extension extends one input order type point by point, then continuing on the remaining set of order types. After extension with one abstract point, we check if the created order type of size $n + 1$ (son) has the initial order type of size n as its predecessor order type (father) in the order type extension graph. Only if this is the case we keep it as a candidate for the output. The very general approach with the order type extension graph guarantees to avoid duplicates in the construction process, thus it can be used for recursive enumeration techniques, known as reverse search, cf. Avis and Fukuda [7]. An additional benefit of this technique is that it can be applied iteratively, i.e., extending from n points to $n+1$, $n+2$, and so on, without storing intermediate results. In fact, only the order types corresponding to a single path of the order type extension graph have to be kept in memory, that is, the edges describing the father-son relationship between order types of size n, $n+1$, $n+2$, and so on. This allows calculations which otherwise would not be possible because of enormous storage requirements for intermediate steps. In addition, applications based on the order type extension graph are easily executed in parallel. Thus, highly time intensive problems may be settled through distributed computing approaches.

4 New Rectilinear Crossing Numbers

4.1 Subset Property for $\overline{cr}(K_n)$

The next two well-known lemmas (see e.g. Guy [11] for references) provide the necessary relations to obtain a subset property for $\overline{cr}(K_n)$.

Lemma 1 $\overline{cr}(K_n) \geq \lceil \frac{n}{n-4}\, \overline{cr}(K_{n-1}) \rceil$

Corollary 2 (**Crossing number subset property**) *For any drawing of K_n with c crossings there exists at least one sub-drawing K_{n-1} with at most $\lfloor \frac{n-4}{n}\, c \rfloor$ crossings.*

Lemma 3 *Let $n \in \mathbb{N}$ be odd. Consider a straight-line drawing of K_n with c crossings. Then: $c \equiv \binom{n}{4} (mod\, 2)$.*

A drawing of K_{13} with 229 (or fewer) crossings contains at least one sub-drawing K_{12} with $\lfloor \frac{9}{13}\, 229 \rfloor = 158$ (or fewer) crossings. Recursive application shows that there exists a sub-drawing of size 11 with $\lfloor \frac{8}{12}\, 158 \rfloor = 105$ crossings. By the parity property we can further reduce the number of crossings for the 11-point subset to at most 104. Thus to achieve a data base of all order types of size 13 with 229 (or fewer) crossings, one can start with a complete data base of order types of size 11 defining drawings of K_n with at most 104 crossings, i.e., either 102 or 104 crossings.

4.2 Results on $\overline{cr}(K_n)$ for $n \geq 12$

Using the crossing number subset property, we were able to calculate the rectilinear crossing numbers for $n = 12, ..., 17$, see Table 2.

n	12	13	14	15	16	17
$\overline{cr}(K_n)$	153	229	324	447	603	798
d_n	1	4 534	20	16 001	36	$\geq$ 37269

Table 2: $\overline{cr}(K_n)$ for $n = 12, ..., 17$.

The numbers d_n of inequivalent drawings of K_n minimizing the number of crossings are given in the last row of Table 2. To obtain these numbers we had to perform the more challenging task of deciding the realizability of the calculated abstract order types. Our heuristics - see Section 2 - found realizing point coordinates for all optimal abstract drawings for $n \leq 16$. Thus, the calculated values are exact. Note that the numbers of inequivalent optimal drawings of K_n follow a parity pattern. There are relatively few drawings of K_n with $\overline{cr}(K_n)$ crossings for even n compared to the case of odd n. This property is the main reason that allows complete abstract extension to work so well, as the problem itself cuts down on the number of interesting sets periodically.

In addition to new results on $\overline{cr}(K_n)$ for constant n, we also achieved an improvement on the asymptotic upper bound. We constructed a set of 54 points with 115999 crossings such that with the strategy of lens replacement [3] we were able to prove the next theorem. The previously best known bound was $\overline{\nu}^* < 0.38074$, whereas $\overline{\nu}^* > 0.37533$ still holds as a lower bound [8].

Theorem 4 $\overline{\nu}^* = \lim_{n\to\infty} \overline{cr}(K_n)/\binom{n}{4} < 0.38058$

5 Further Applications

5.1 Happy End Problem

Erdös and Szekeres asked in 1935 for the smallest number $g(k)$, such that each point set in the plane with at least $g(k)$ points contains a convex k-gon [10]. For $k > 5$ this problem is still unsolved, where it is known that $g(6) \leq 37$. The conjecture is that the true value for $g(6)$ is 17. To answer this conjecture our plan is to apply our abstract extension technique in order to obtain all sets without empty convex hexagons for $n \leq 17$. If we cannot find a set for $n = 17$ this will prove the conjecture to be true. The subset property for this problem is obvious: any $n-1$ point subset of a set of n points to be considered must not contain a convex hexagon.

5.2 Decomposition

Similar to the convex-decomposition problem of decomposing a point set into convex polygons one might allow the resulting faces to be either convex polygons or pseudo-triangles [4]. When investigating this problem it turned out to be important to know optimal decompositions of small sets. In this context we asked for independent (disjoint) empty convex polygons spanned by the set. Let us briefly mention two results we got from the data base, see [4] for details. First: Any set of 8 points contains either an empty convex pentagon or two independent empty convex quadrilaterals. And with a similar flavor: Any set of 11 points contains either an empty convex hexagon or an independent empty convex pentagon and an empty convex quadrilateral. The mentioned results directly lead to an upper bound of $7n/10$ for the number of convex or pseudotriangular faces used to decompose a set of n points.

5.3 Counting Triangulations

Counting the number of triangulations of a set of points in the plane is another interesting geometric problem. Exact numbers, using our data base, are known for all sets with $n \leq 11$ points. The best general asymptotic lower bound for this problem is based on these results for small sets [5]. To improve the bound it will be useful to obtain a tight lower bound for $n = 12, 13, \ldots$. As a subset property for this task we can use the fact that adding an interior point to a given set increases the number of triangulations by some constant factor.

6 Open Problems

The next steps of our investigation will be to compute $\overline{cr}(K_{18})$. The possible range for $\overline{cr}(K_{18})$ is $\{1026, 1027, 1028, 1029\}$, where our conjecture is $\overline{cr}(K_{18}) = 1029$. Using heavy distributed computing we consider this task to be realistic in the near future.

An interesting open problem is whether there always exists at least one optimal drawing of K_n which contains an optimal sub-drawing of K_{n-1}. A potential counter-example is $n = 18$, as all 17-point subsets of the only known drawing of K_{18} with 1029 crossings determine more than $\overline{cr}(K_{17}) = 798$ crossings.

7 Acknowledgments

We would like to thank Birgit Breitenlechner, Clemens Huemer and Bernhard Kornberger for implementing many algorithms used for our investigations. We are grateful to H.F. Jensen for generously sharing his data and ideas about crossing numbers with us.

References

[1] O.Aichholzer, *Rectilinear Crossing Number Page.* `http://www.ist.tugraz.at/staff/aichholzer/crossings.html`

[2] O.Aichholzer, F.Aurenhammer, H.Krasser, *Enumerating order types for small point sets with applications.* Order, 19:265-281, 2002.

[3] O.Aichholzer, F.Aurenhammer, H.Krasser, *On the crossing number of complete graphs.* 18^{th} ACM Symposium on Computational Geometry (SoCG), Barcelona, Spain, pages 19-24, 2002.

[4] O.Aichholzer, C.Huemer, S.Renkl, B.Speckmann, and C.Toth, *Partitioning Point sets into Empty Convex Polygons and Pseudo triangles.* manuscript, 2004.

[5] O.Aichholzer, F.Hurtado, and M.Noy, *A lower bound on the number of triangulations of planar point sets.* Computational Geometry: Theory and Applications, 29(2):135-145, 2004.

[6] O.Aichholzer, H.Krasser, *The point set order type data base: A collection of applications and results.* 13^{th} Canadian Conference on Computational Geometry (CCCG), Waterloo, Ontario, Canada, pages 17-21, 2001.

[7] D.Avis, K.Fukuda, *Reverse search for enumeration.* Discrete Applied Mathematics 65, pages 618-632, 1996.

[8] J.Balogh, G. Salazar, *On k-sets, convex quadrilaterals, and the rectilinear crossing number of K_n.* Manuscript, submitted.

[9] J.Bokowski, J.Richter, *On the finding of final polynomials.* European J. Combinatorics 11, pages 21-34, 1990.

[10] P.Erdös, G.Szekeres, *A combinatorial problem in geometry.* Compositio Mathematica, (2):463-470, 1935.

[11] R.K.Guy, *A combinatorial problem.* Nabla (Bulletin of the Malayan Mathematical Society), 7, 68-72, 1960.

[12] F.Harary, A.Hill, *On the number of crossings in a complete graph.* Proc. Edinburgh Math. Society (2) 13 (1962), 333-338.

[13] N.E.Mnëv, *On manifolds of combinatorial types of projective configurations and convex polyhedra.* Soviet Math. Dokl., 32, pages 335-337, 1985.

Quadrangulations and 2-Colorations

Carmen Cortés*¶ Alberto Márquez†¶ Atsuhiro Nakamoto‡ Jesús Valenzuela§¶

Abstract

Any metric quadrangulation (made by segments of straight line) of a point set in the plane determines a 2-coloration of the set, such that edges of the quadrangulation can only join points with different colors. In this work we focus in 2-colorations and study whether they admit a quadrangulation or not, and whether, given two quadrangulations of the same 2-coloration, it is possible to carry one into the other using some local operations, called *diagonal slides* and *diagonal rotation*. Although the answer is negative in general, we can show a very wide family of 2-colorations, called *onions 2-coloration*, that are quadrangulable and which graph of quadrangulations is always connected.

1 Introduction

Given a set S, either a polygon or a point set, a quadrangulation of S is a partition of the interior of S, if S is a polygon, or of the convex hull of S, if S is a point set, into quadrangles (quadrilaterals) obtained by inserting edges between pairs of points (diagonals between vertices of the polygon) such that the edges intersect each other only at their end points. Not all polygons or point sets admit quadrangulations, even when the quadrangles are not required to be convex. In the study of finite element methods and scattered data interpolation, it has recently been shown that quadrangulations of point sets may be more desirable objects than triangulations [2]. The quadrangulations of polygons have been also investigated in Computational Geometry, mostly in the context of guarding or illumination problems.

From now on we call a polygon or point set *quadrangulable* if it admits a quadrangulation without adding any additional point (Steiner point).

There are two different characterizations of quadrangulable point sets:

- if and only if there exists a triangulation of the set such that its dual graph contains a perfect matching [5].
- if and only if it has an even number of points in its convex hull [1].

A quadrangulation is constructed (with the addition of a Steiner point to obtain an even number of points in the convex hull, if necessary) in $\Theta(n \log n)$.

For a more complete vision on quadrangulations we recommend Toussaint's survey [6].

Given a quadrangulation of a point set in the plane, it determines a 2-coloration of the set, such that edges of the quadrangulation can only join points with different colors. In this work we focus on 2-colorations and study whether they admits a quadrangulation or not (Section 2), and whether, given two quadrangulations of the same 2-coloration, it is possible to carry one into the other using some local operations (Section 3). Finally, in Section 4 we present a very wide family of 2-colorations, called *onions 2-coloration*, that are quadrangulable and which graph of quadrangulations is always connected.

2 2-colorations and quadrangulations

Suppose we have a 2-colorated point set S in the plane and we want to know if it is possible to construct a quadrangulation of its convex hull. A similar condition to the one given by [1] is, in this context, evident:

Lemma 1 *A necessary condition for a 2-coloration of a point set S to admit a quadrangulation (to be* quadrangulable*) is that*

1. *the number of points of the convex hull of S is even; and*
2. *consecutive points of the hull have different color.*

But even when the conditions of Lemma 1 are fulfilled, it is easy to find non-quadrangulable 2-colorations, as the one at the left of Figure 1. In order to construct a quadrangulation, point 1 cannot be joined with c because then b can be joined with no black point but 1. So we draw an edge from 1 to b. Now b cannot be joined either with 3 or with 4, because then c or 2, respectively, would be isolated, and cannot be part of any quadrangulation. But if

*Department of Applied Mathematics I, University of Seville, ccortes@us

†Department of Applied Mathematics I, University of Seville, almar@us.es

‡Department of Mathematics, Yokohama National University, nakamoto@edhs.ynu.ac.jp

§Department of Mathematics, University of Extremadura jesusv@unex.es

¶Partially supported by MCyT project BFM2001-2474.

we join b with 2 the only way to complete a quadrilateral is to match 2 and d, that leaves 3 isolated. It is important to remark that we are talking about 2-colorations instead sets of points. Thus, while the 2-coloration at the left of Figure 1 is not quadrangulable, the underlying point set is, as we see in the right picture.

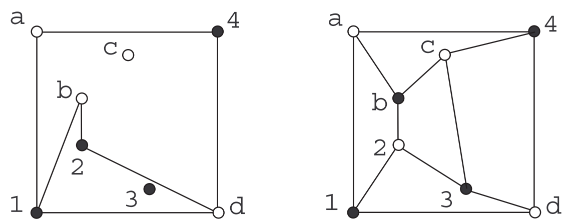


Figure 1: The set is either quadrangulable or not depending on the coloration.

Notice that in the right picture we have interchanged the colors of 2 and b, obtaining a set with two convex layers, both of them made by points with alternate colors. This is a interesting configuration since, as we will see in Section 4, it is always quadrangulable.

3 Diagonal transformation in quadrangulations

Nakamoto [4], working with topological quadrangulations on surfaces, defines two diagonal transformations; the diagonal slide and the diagonal rotation, that are shown in Figure 2. Note that, while the diagonal slide does not modify the coloration of the set, the diagonal rotation changes the color of the *center* of rotation (because, in other case, points with the same color are joined).

Since the same point set can have different colorations, it is not always possible to change any two quadrangulations one into each other using only diagonal slides. In Figure 3 two colorations of the same point set are shown; one with four black and four white points, and another with five white and three black points. Since diagonal slides preserve colorations, it is not possible to use them to transform one quadrangulation into the other. However, it is easy to see that this can be done using also diagonal rotations.

Nakamoto [4] proved that in any closed surface it is always possible to carry one topological quadrangulation of a set into any other if

1. both diagonal slides and rotations are allowed; or
2. both quadrangulations have the same number of points of each color by using only diagonal slides.

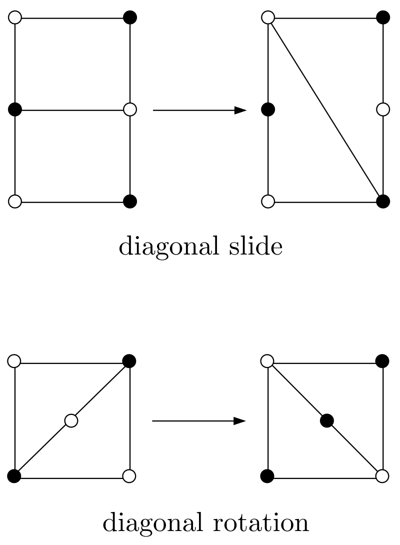


Figure 2: Diagonal transformations on quadrangulations.

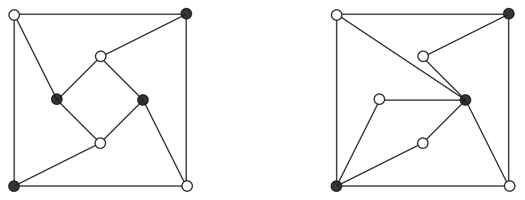


Figure 3: Quadrangulations of the same set with different colorations.

This can be seen in terms of the connectivity of the graph of quadrangulations. The *graph of quadrangulations of a point set* is the graph having all the quadrangulations of the set as nodes, and with adjacentcies corresponding to diagonal slides or diagonal rotations. Similarly, the *graph of quadrangulations of a 2-coloration* has as nodes the quadrangulations of a given 2-coloration. Since diagonal rotations change the 2-coloration, the adjacentcies are determined only by diagonal slides.

Both graph of quadrangulations are, in general, not connected. In Figure 4, it is shown a 2-coloration that admits only two quadrangulations, being not possible to perform any diagonal slide. If we also allow diagonal rotations it can be shown that it is not possible to transform one quadrangulation into the other. This gives rise to the following theorems:

Theorem 2 *There are 2-colorations with disconnected graph of quadrangulations.*

Theorem 3 *There are point sets with disconnected graph of quadrangulations.*

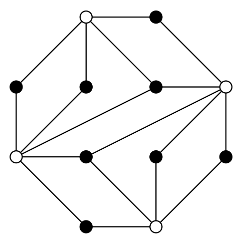


Figure 4: A set with disconnected graph of quadrangulations.

Our example has disconnected graph of quadrangulations both as a 2-coloration and as a point set. An open problem is to determine if both things always come together, or if there exit point sets with connected graph of quadrangulations that admit 2-colorations which graph of quadrangulations is not.

In spite of the graph of quadrangulations of an arbitrary 2-coloration is, in general, not connected, in the next section we present a wide family of 2-colorations having this property.

4 Onion 2-colorations

If a set of sites have an even number of vertices in its convex hull, then it is quadrangulable, and vice-versa [1]. This is rewritten for 2-colorations in Lemma 1, but only as a necessary condition, since we find non-quadrangulable 2-colorations that fulfill it, as the one we saw in Figure 1. But, what about if we extend Lemma 1 to the interior of the set? If the convex hull of the 2-coloration fulfill the lemma, we remove it and examine the convex hull of the remaining points, and so on. A 2-coloration with this property is quadrangulable and its graph of quadrangulations is connected.

We call *onion 2-coloration* to a 2-coloration of a point set such that all its convex layers have an even number of points with alternate colors. An *onion layer* of an onion 2-coloration is the set of edges that are part of a convex layers of the set. We call O_i, with $i = 0, \ldots, l$, to the onion layers of the onion 2-coloration, such that O_i is inside the polygon defined by O_j if $i > j$ (Figure 5). Note that the polygon defined by O_l does not contain any point of the onion 2-coloration and that O_0, the convex hull, is always included in every quadrangulation of the onion 2-coloration. By definition, the points on every onion layer satisfy Lemma 1, that implies the following result.

Proposition 4 *Onion 2-colorations are quadrangulable.*

The main idea of the proof is to draw a triangulation joining points between two consecutive onion layers. By deleting the edges matching points with the same color (Figure 6) we obtain a quadrangulation of the onion 2-coloration. It should be note that we are drawing quadrangulations of convex polygons with a convex hole, being the general case, decide whether a polygon with holes admits a quadrangulation, an NP-complete problem [3].

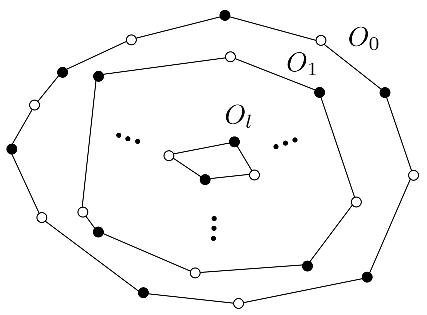


Figure 5: An onion 2-coloration and its onion layers.

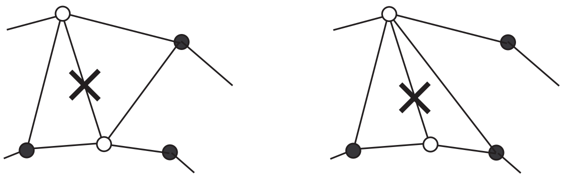


Figure 6: By deleting the diagonals between points with the same color we obtain quadrilaterals.

But onion 2-colorations are not the only quadrangulable 2-colorations, since there are quadrangulable 2-colorations with non alternate colors in some of its convex layer (Figure 7) or with a odd number of points on them.

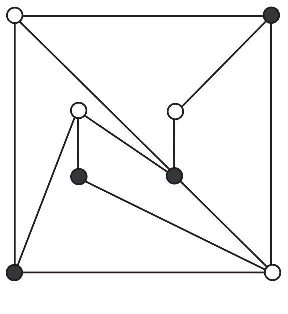


Figure 7: The colors of the inner convex layer are not alternate.

In addition to be quadrangulable, onion 2-colorations have connected graph of quadrangulations. The proof is based in the following lemmas:

Lemma 5 *Given two quadrangulations of an onion 2-coloration containing all their onion layers, we can transform one into the other by only using diagonal slides.*

Lemma 6 *Any quadrangulation of an onion 2-coloration can be carried into another containing its onion layers using diagonal slides.*

From these lemmas it can be easily proved the connectivity of the graph of quadrangulations of any onion 2-coloration.

Theorem 7 *The graph of quadrangulations of an onion 2-coloration is non-empty and connected.*

In particular, if the onion 2-coloration have only one layer, we obtain the following result:

Corollary 8 *The graph of quadrangulations of any quadrangulable 2-coloration in convex position is connected.*

5 Conclusions and open problems

Two main ideas can be extracted from this work: to be quadrangulable depends on the 2-coloration of the set, and the graph of quadrangulations of both a 2-coloration and a point set is, in general, not connected. However, there exits a wide family of 2-colorations, the onion 2-colorations, that are quadrangulable and which graph of quadrangulations is connected.

There are several questions that appear all along the present work. One is to explore new conditions for a 2-coloration to be quadrangulable, searching for new families of quadrangulable 2-colorations. Related to the graph of quadrangulations, an interesting approach is to study the relationship, if it exits, between the connectivity of the graph and the coloration of the set. And, since the example presented (Figure 4) of a set with disconnected graph of quadrangulations have rows with until four collinear points, it would be convenient to construct a new example in general position. Probably this implies to work with sets with greater cardinal and complexity. Finally, another line for future works is, since they also admit 2-colorations, to extend this study from quadrangulations to 2n-lations of point sets.

References

[1] P. Bose and G. Toussaint. Characterizing and efficiently computing quadrangulations of planar point sets. *Computer Aided Geometric Design*, vol. 14, 1997, pp. 763-785.

[2] M. L. Lai and L. L. Schumaker. Scattered data interpolation using piecewise polynomials of degree six. *SIAM Numer. Anal.*, 34(1997), pp.905–921.

[3] A. Lubiw Decomposing polygonal regions into convex quadrilaterals. *Proc. Symposium on Computational Geometry*, 1985, pp. 97–106

[4] A. Nakamoto. Diagonal transformations in quadrangulations of surfaces. *J. Graph Theory*, 21:289–299, 1996.

[5] S. Ramaswami, P. Ramos and G. Toussaint. Converting triangulations to quadrangulations *Computational Geometry: Theory and Applications*, Vol. 9, March 1998, pp. 257-276.

[6] G. Toussaint. Quadrangulations of planar sets. In *Proceedings of the 4th International Workshop on Algorithms and Data Structures*, pages 218–227. Springer-Verlag, 1995.

Discrete Curvatures and Gauss Maps for Polyhedral Surfaces

Lyuba Alboul*   Gilberto Echeverria†   Marcos Rodrigues‡

Abstract

The paper concerns the problem of correct curvatures estimates directly from polygonal meshes. We present a new algorithm that allows the construction of unambiguous Gauss maps for a large class of polyhedral surfaces, including surfaces of non-convex objects and even non-manifold surfaces. The resulting Gauss map provides shape recognition and curvature characterisation of the polyhedral surface (polygonal mesh) and can be used further for optimising the mesh or for developing subdivision schemes.

1 Introduction

In many applications a physical object is represented by discrete data, obtained by some measurement system. A polyhedral model (a triangular mesh, piecewise linear surface) is the easiest way to obtain a preliminary sketch of the given object. A solid object is represented by its boundary, i.e. by the surface that bounds the object. Triangular or polygonal meshes are commonly used in modern computer-related applications to represent surfaces in three-dimensional space. Therefore, there is a substantial need for accurate estimates of geometric attributes such as surface area, normal vectors, and curvatures directly from a mesh. A smooth surface S is uniquely characterised and quantified by the *metric tensor* and by the *Weingarten map* or the *shape operator* [Kuhn02]. The shape operator is the second-order invariant (in other words, curvature) that completely determines the shape of the surface S. In recent years significant efforts have been made to define the analogues of differential geometry concepts on meshes that imitate those of a smooth surface ([Alb96, Bor03, Dyn01, Malt02, Mey03]). Among those concepts surface curvatures are most important. Surface curvatures are basic measures to describe the local shape of a smooth surface. However, a mesh (a polyhedral surface) is not smooth, and there is still no consensus about the most appropriate way of estimating such geometric quantities as curvatures. On the other hand, methods are being developed to capture curvature information without referring to higher-order formulas of differential geometry. These methods are based on the discrete curvature concepts and are of growing interest for geometric modelling [Malt02, Alb03, Dyn01, Mey03]. Discrete curvatures can be computed directly from the polygonal mesh. The principal difference to smooth surfaces is that the curvatures in polyhedral surfaces are concentrated around the vertices and along the edges.

If we think of a polyhedral surface as an approximation of a smooth surface, then, informally speaking, curvatures of a domain of the underlying smooth surface are 'glued' together in the corresponding domain of a polyhedral surface.

Therefore, analogue measures of curvature in a piecewise linear setting should be analogues of *integral formulae* for curvature in a 'smooth' setting and should preserve integral relations for curvature, such as the Gauss-Bonnet theorem ([Br79, Banch70, Alb96]. Such analogues exist and were introduced long ago in the frames of the theory of non-regular surfaces (see an overview in [Alb96]). These analogues were discussed in detail in [Br79], where the authors also compare discrete curvatures with their smooth counterparts.

In the last five-six years the amount of papers that explore discrete curvatures in one or another context has increased significantly. Much attention is paid to the discrete Gaussian curvature, known also as the *angle deficit*. The angle deficit is used to estimate the Gauss curvature of smooth surfaces. In [Bor03] the problem of the correct estimation of the Gauss curvature is investigated in detail, and they show that approaches based on the use of normalized angular deficits are often erroneous, and can be applied correctly only if the geometry of meshes is precisely controlled. We agree with them, and in this paper we highlight why the angular deficit is neither sufficient to estimate the Gaussian curvature of the underlying smooth surface nor to capture the curvature information of a polyhedral surface. Loosely speaking, the reason is that there are more curvatures for polyhedral surfaces than for smooth ones [Br79, Alb96, Alb03]. This fact is still not fully acknowledged in geometric applications, but without addressing it, it is impossible to develop correct curvature estimates.

Besides the need to derive correct curvature estimates directly from polygonal meshes, there is also a need for visualisation of an object in order to explore complex shapes and emphasize hidden details. We pro-

*Materials and Engineering Research Institute, Sheffield Hallam University, UK, L.Alboul@shu.ac.uk

†idem, G.Echeverria@shu.ac.uk

‡idem, M.Rodrigues@shu.ac.uk

pose an approach that addresses both needs, and that empowers us to correctly and consistently describe and visualise complex 3D shapes based on curvature properties. Our method to characterise surface shape is based on constructing the Gauss map directly from the polygonal mesh, an area of research with still scarce and ambiguous results for non-convex objects [Low02]. The resulting Gauss map provides a description of the surface by determining its domains with respect to incorporated curvatures. Each domain can be split up into uniquely determined sub-domains; therefore each surface can be associated with the introduced *Gauss map signature*, abbreviated as GMS. The GMS extracts convex, concave and saddle regions in the underlying surface. These regions are often only implicitly present in a polyhedral surface, and cannot be determined by the sign of the angle deficit only. The GMS method besides shape recognition and description can be used for optimisation of the underlying model or for developing subdivision schemes. The method provides also a better insight into the geometric structure of complex triangle meshes, by describing various vertex types, some of them with a very complex GMS. A good understanding of the geometry of meshes is a step towards more robust mesh manipulation algorithms. Finally, the proposed GMS method is simple to compute, easy to view dynamically and effective in visualising complex polyhedral surfaces.

2 Polyhedral Surfaces: Discrete curvatures and Gauss map

By a polyhedral surface we understand a triangulated polyhedral surface. Designating $\mathbf{V}$ as a finite point set in three-dimensional space, $\mathbf{V} = \{V_i, i = 1, \ldots, n\}$, we denote by $\mathbf{P}(\mathbf{V})$ a polyhedral surface with the vertex set $\mathbf{V}$. The term *polyhedron* refers to a closed polyhedral surface. In such a setting a polyhedron is bounded, but might be non-simple, i.e. non-homeomorphic to a sphere, as well as being multi-connected and self-intersecting, and its interior volume is not necessarily part of the polyhedron. Therefore, a polyhedron is not necessarily a solid body. Given a polyhedron $\mathbf{P}(\mathbf{V})$, the set of its vertices is denoted by V, the edges by E, and the faces by F.

Definition 1 *The star* $\mathbf{Str}(\nu)$ *of a vertex* ν *is the union of all the faces and edges that contain the vertex, and the link* $\mathbf{Lnk}(\nu)$ *of the star (the boundary of the star) is the union of all those edges of the faces of the star* $\mathbf{Str}(\nu)$ *that are not incident to* ν.

2.1 Discrete Curvatures

In this paper we are interested only in discrete curvatures related to the integral Gaussian curvature, i.e. those that are supported on the vertices. The common expression for the integral Gaussian curvature of a domain U of a smooth surface S is $\int_U K dA$ [Kuhn02]. **Curvature ω around vertex ν** is defined as:

$$\omega = 2\pi - \theta \tag{1}$$

where $\theta = \sum \alpha_i$ is the total angle around vertex ν, and α_i are those angles of the faces in the $Str(\nu)$ that are incident to ν. Sometimes one refers to ω simply as the Gaussian curvature around the vertex, or *discrete Gaussian curvature*. Obviously, expression 1 is valid for any point $x \in P$. For a domain $U \subseteq P$ the total curvature Ω_U is determined as $\Omega_U = \sum_{\nu \in U} \omega_\nu$. For an oriented closed polyhedral surface P of genus g Ω_U is equal to $(1-g)4\pi$, so the discrete analogue of the Gauss-Bonnet theorem is preserved.

Positive (extrinsic) curvature ω^+: The following measure which we determine is an analogue of the total absolute curvature of a polyhedral domain. However, in Figure 1 we can see that in both polyhedra all curvatures ω_ν are positive and actually are equal for every corresponding vertex.

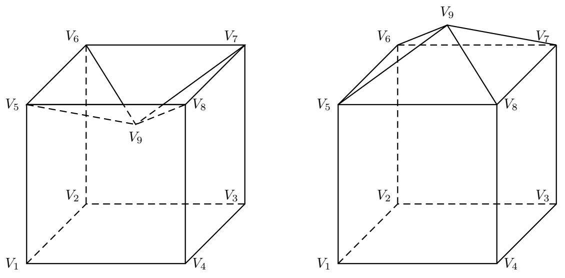


Figure 1: Two polyhedra

Therefore, we have:

$$\Omega(P_1) = \Omega(P_2) = \sum_{\nu \in P_1} |\omega_\nu| = \sum_{\nu \in P_2} |\omega_\nu| = 4\pi. \tag{2}$$

The left polyhedron is non-convex, but the above equation does not reflect this fact. For a closed non-convex smooth surface S the total absolute curvature $K_{abs} = \int_S |K| dA$ is greater than 4π; therefore, $\sum_{\nu \in P} |\omega_\nu|$ is not an appropriate analogue of K_{abs}. The problem is that the curvature ω around a vertex may consist of positive and negative 'parts' that are 'glued' together; and the task is to separate them. If vertex ν belongs to the boundary of the convex hull of its star (i.e. the convex hull of ν and all vertices in its star), then we can single out another star $\mathbf{Str}^+(\nu)$ with ν as the vertex and those edges of $\mathbf{Str}(\nu)$ that belong to the boundary of the convex hull. The edges of $\mathbf{Str}^+(\nu)$ will determine the faces of $\mathbf{Str}^+(\nu)$. We refer to $\mathbf{Str}^+(\nu)$ as the convex cone of vertex ν. Then

$$\omega^+ = 2\pi - \theta^+ \tag{3}$$

where θ^+ is the total angle around ν in $\mathbf{Str}^+(\nu)$. ω^+ is equal to zero, if the vertex and all the vertices in its

star lie in the same plane. If the convex cone around ν doesn't exist, i.e. ν lies inside the convex hull of $\mathbf{Str}(\nu)$, then ω^+ is, by definition, equal to zero.
Negative (extrinsic) curvature ω^-: We can now 'extract' the negative part of ω as follows

$$\omega^- = \omega^+ - \omega \tag{4}$$

Absolute (extrinsic) curvature ω_{abs}:

$$\omega_{abs} = \omega^+ + \omega^- \tag{5}$$

On the basis of the types of curvatures around a vertex one distinguishes three basic types of vertices for an embedded polyhedral surface ([Br79, Alb96]): convex vertices ($\omega^+ = \omega$), saddle vertices ($\omega^- = -\omega$) and mixed vertices ($\omega^+ > 0, \omega^+ \neq \omega$) (see Figure 2).

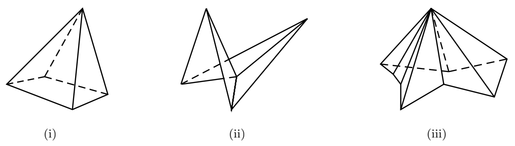


Figure 2: Examples of convex (i), saddle (ii), and mixed (iii) vertices

A mixed vertex, however, possesses always the convex cone around its star. A mixed vertex and its correspondent convex cone are shown in Figure 3.

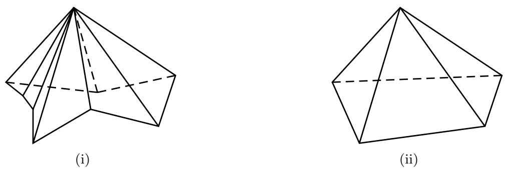


Figure 3: Mixed vertex (i) and its convex cone (ii)

Total absolute extrinsic curvature Ω_{abs}: is defined as the sum of absolute extrinsic curvatures of all the vertices of a polyhedral surface P:

$$\Omega_{abs} = \sum_i \omega_{abs}(\nu_i) = \sum \left[\omega^+(\nu_i) + \omega^-(\nu_i)\right] \tag{6}$$

Ω_{abs} takes different values on the polyhedra that are depicted in Figure 1. It is equal to 4π on the right polyhedron, as it represents a convex body, and is greater than 4π on the left polyhedron.

2.2 Gauss map

Separation of the positive and negative parts of the curvature for a mixed vertex can also be carried out using the Gauss map. For a domain U of smooth surface S the Gauss map $N(U)$ may be thought of as the map assigning to each point $p \in U$ the point on the unit 2-sphere $S^2 \in \mathbf{R}^3$, by 'translating' the unit normal vector $N(p)$ to the origin [Kuhn02]. The endpoints of normals, therefore, will cover a certain region on S^2. If a neigbourhood $U(p)$ is small such that the map $N(U(p))$ is one-to-one and orientation-preserving (outward normals at corresponding points on S and S^2 correspond), then the area $N(U(p))$ is considered positive, and the corresponding region $U(p)$ is said to be strictly *convex* and the Gaussian curvature at p defined as $|K(p)| = \lim_{U(p)\downarrow p} \frac{AreaN(U(p))}{AreaU(p)}$, is positive, i.e. $K(p) > 0$. If the map $N(U(p))$ is one-to-one but orientation reversing, then the area $N(U(p))$ is considered to be negative, p is a saddle point and $K(p) < 0$. Of course, different regions of S can be mapped to the same region on the unit sphere, which results in multiplicities of the Gauss map.
To compute directly the image of the Gauss map of a given vertex, we need to construct the outward vector normal for each of the facets around a vertex and then draw geodesics arcs between the images of neighbouring faces to obtain a graphic image. The union of the Gauss maps for all vertices is the Gauss map of a polyhedral surface.
An orientation of the contour around the vertex on a polyhedral surface induces the orientation on the boundary of the spherical polygon. Thus we can evaluate the curvature around the vertex by computing the area of the spherical image with the sign + (plus) in the case that the orientation is preserved, and with the sign − (minus) otherwise.

3 Results

To characterise a polyhedral model we have developed algorithms that have the following functionalities:

1. *Determination of the Gauss Map* for each of the vertices in V; and
2. *Curvature Visualisation*, which displays a graphical representation of the Gauss map.

We are able to divide the Gauss Map for a vertex into different spherical polygons, determine the orientation of each polygon and thus its sign. Therefore, we are able not only to separate $\omega(v)$ for a vertex v into positive and negative parts $\omega^+(v)$ and $\omega^-(v)$, but to separate into subparts of the same sign. The number of subparts together with their signs represents the *Gauss map signature* of a vertex. Each subpart of the negative sign represents a potential (hidden) saddle region.
The main advantage of our method is that it allows the determination of incorporated curvatures of various types of vertices, including all the above-mentioned ones and much more complex such as the *monkey saddle*, or vertices with *self-intersections*,

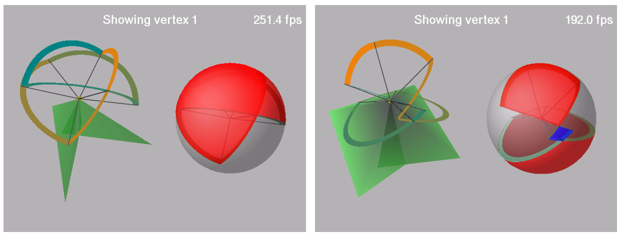


Figure 4: Pinch vertex and a reverse pinch vertex with corresponding Gauss maps

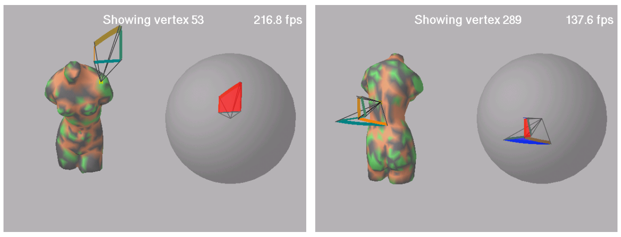


Figure 5: Torso and its Gauss map visualisations

which don't fit exactly in the category 'mixed', described in the previous section. Eventually, we can determine the curvatures of a vertex of any type (of an oriented polyhedral surface P). It is also possible to display the Gauss Map for all the vertices of the object simultaneously, or select only one of the vertices for its Gauss Map to be shown exclusively (or, correspondingly, to visualise the Gauss map of a region on the surface). The method is interactive, and we can visualise the regions of positive curvature separately from the regions of negative curvature.

Examples of Gauss map visualisations are given below. The display of the Gauss Map is done in two different views, or scenes, and is implemented using OpenGl. The left scene shows the model of the original object and, in the right scene, the areas for the Gauss Map are drawn on top of a sphere. Positive areas are shown in red, while negative areas are displayed in blue (lighter and darker grey in the black-white print). The corresponding areas on the object are coloured in green and red respectively (dark and light grey in the black-white print).

Figure 4 shows the Gauss map visualisation of two complex vertices with self-intersections, which we call a *pinch vertex* and a *reverse pinch vertex*. In order to understand the difference between a pinch vertex and a reverse pinch vertex, imagine a walk along the link of the star of a vertex ν. In the case of the pinch vertex the walk makes two full turns around the vertex, both turns have the same orientation (for example, counter clockwise). In the case of the reverse pinch point, the walk makes also two full turns, one is, for example, in the counter clockwise direction, and the second one - in the 'reverse' direction (i.e. clockwise). The Gauss map of the pinch vertex has two overlapping areas, each of positive sign. One area is equal to the curvature of the convex star of the pinch vertex. The Gauss map of the reverse pinch vertex has also two areas of positive curvature, separated by the area of negative curvature.

A more complex object and its Gauss map visualisation are presented in Figure 5.

4 Future work

Current on-going research includes the visualisation of the processes of mesh simplification and optimisation by using the GMS method, as well as to use it for developing subdivisions schemes based on curvature estimations.

References

[Alb96] Alboul, L. and van Damme, R. Polyhedral metrics in surface reconstruction. In: Mullineux, G. (ed.), *The Mathematics of Surfaces VI*, pp. 171-200, Oxford, 1996.

[Alb03] Alboul, L., Optimising triangulated polyhedral surfaces with self-intersections. In: Wilson, M. J., Martin, R. R., *Mathematics of Surfaces*, LNCS 2768, pp. 48-72, 2003.

[Banch70] Banchoff, T.F., Critical points and curvature for embedded polyhedral surfaces. *Amer. Math. Monthly* 77, pp. 475-485, 1970.

[Bor03] Borelli, V., Cazals, F, and Morvan J.-M., On the angular defect of triangulations and the pointwise approximation of curvature, *Comp. Aided Geom. Design* 20, pp. 319-341, 2003.

[Br79] Brehm, U., and Kühnel, W., Smooth approximation of polyhedral surfaces with respect to curvature measures. In: *Global differential geometry*, pp. 64-68, 1979.

[Dyn01] Dyn N. et al., Optimising 3D triangulations using discrete curvature analysis. In: Lyche, T., and Schumaker, L. L. (eds.), *Mathematical Methods in CAGD*, pp. 1-12, Vanderbilt University Press, 2001.

[Kuhn02] Kühnel, W., Differential geometry. Curves-Surfaces-Manifolds, *Amer Math Society*, 2002.

[Low02] Lowekamp, B., Rheingans, P., and Yoo, T.S., Exploring surface characteristics with interactive Gaussian images: a case study. *Proc. of the conference on Visualization '02*, pp. 553-556, 2002.

[Malt02] Maltret, J.-L., and Daniel, M., Discrete curvatures and applications: a survey. *Research report LSIS-02-004*, March 2002.

[Mey03] Meyer, M., Desbrun, M., Schröder, P., and Barr, A. H., Discrete differential-geometry operators for triangulated 2-manifolds. In: Hege, H.-Ch., and Polthier, K. (eds.), *Visualization and Mathematics III*, pp. 35-59, Springer-Verlag, 2003.

On the Number of Facets of Three-Dimensional Dirichlet Stereohedra III: Cubic Group

Pilar Sabariego [*] Francisco Santos [†]

Abstract

We prove that Dirichlet stereohedra for cubic crystallographic groups cannot have more than 105 facets. This improves a previous bound of 162 [3].

1 Introduction

A *stereohedron* is any bounded convex polyhedron which tiles the space by the action of a crystallographic group. A particular case is the Voronoi region of a point P in the Voronoi diagram of its orbit GP under the action of a crystallographic group. These stereohedra are called *Dirichlet stereohedra* and are the object of study in this paper.

The study of the possible combinatorial types of stereohedra and, in particular, of their maximum number of facets, is related to Hilbert's 18th problem [9]. The two main previous results are:

- The *fundamental theorem of stereohedra* (Delone, 1961 [5]) asserts that a stereohedron of dimension d for a crystallographic group G with a aspects cannot have more than $2^d(a+1)-2$ facets, where the number of *aspects* of G is the number of translational lattices in which a generic orbit of G decomposes. 3D crystallographic groups can have a maximum of 48 aspects, so 3D stereohedra cannot have more than 390 facets.

- P. Engel (see [6] and [7, p. 964]), using a computer search, found in 1980 a 3-dimensional Dirichlet stereohedron with 38 facets, for a cubic group with 24 aspects. This is the stereohedron with the maximum number of facets known.

In previous papers, the second author together with D. Bochiş has initiated an exhaustive study of the number of facets of Dirichlet stereohedra for the different 3D crystallographic groups. They divided the 219 affine conjugacy clases of 3-dimensional crystallographic groups in three blocks, and gave upper bounds for the number of facets of Dirichlet stereohedra in them:

- Within the 100 crystallographic groups which contain reflection planes, the exact maximum number of facets is 18 [1].

- Within the 97 non-cubic crystallographic groups without reflection planes, there are Dirichlet stereohedra with 32 facets and no Dirichlet stereohedron can have more than 80 [2].

For cubic groups without reflection planes (there are 22 of them), Bochiş and Santos were only able to prove an upper bound of 162 facets [3]. Here we improve this bound, and hence the general upper bound for the number of facets of 3D Dirichlet stereohedra, to 105. More precisely, our bound goes "group by group" and it lies below 38 except in the eight so-called "quarter groups" [4]. Our bounds for these eight groups are respectively 42, 43, 53, 66, 73, 74, 73 and 105. Curiously enough, the last (and biggest) one is precisely for the crystallographic group that produces Engel's Dirichlet stereohedron with 38 facets.

2 Outline of the method

The sketch of the method is as follows:

1. We choose a tessellation of the 3-dimensional Euclidean space "adapted" to the group G under study. We call the tiles *fundamental subdomains*. By "adapted" we mean that the tiles are in a finite (and small) number of classes modulo the normalizer of G. We choose one fundamental subdomain of each class, and call them *basic fundamental subdomains*. If two points lie in the same orbit of the normalizer of G then the Dirichlet stereohedra based on them are affinely equivalent. Hence, every Dirichlet stereohedra for G is affinely equivalent to one with basis point in a basic fundamental subdomain.

2. For each basic fundamental subdomain, say D_0, we compute an *extended Voronoi region*, i.e., a region that is guaranteed to contain the union of the Dirichlet stereohedra generated by all the points in D_0. We do this cutting out parts of space that are guaranteed not to belong to any Voronoi region with basis in D_0 because of the presence of certain rotations or translations in G. The precise method is the same used in 2D in [2], except here we do it on the computer because of the extra complexity of the problem.

[*]Departamento de Matematicas, Estadistica y Computacion, Universidad de Cantabria, Santander, Spain, pilar.sabariego@unican.es

[†]Departamento de Matematicas, Estadistica y Computacion, Universidad de Cantabria, Santander, Spain, santosf@unican.es

Of course, the extended Voronoi region is not uniquely defined, and the smaller the one we get is, the better the final bound will result.

3. The extended Voronoi region of a non-basic fundamental subdomain D is now trivial to compute: Find the basic fundamental subdomain D_0 related to $D = \rho D_0$ by a motion ρ in the normalizer of G, and apply ρ to the extended Voronoi region of D_0. We call *influence region* of a basic subdomain D_0 the union of all the subdomains whose extended Voronoi regions intersect the extended Voronoi region of D_0.

Theorem 1 *For every $p \in D_0$, the neighbors of p in the Voronoi diagram of the orbit of p are contained in the influence region of D_0.*

Corollary 2 *The number of facets of Dirichlet stereohedra with base point in D_0 is bounded above by the number of fundamental subdomains in the influence region of D_0 "counted with multiplicity" (i.e., each one counted as many times as the number, perhaps zero, of transformations in G that send it to D_0. In particular only those in the same class of D_0 modulo the action of G are counted).*

All of the above actually follows the [Bochiş-Santos]'s approach, but with two new ingredients:

- [Bochiş-Santos] compute 2-dimensional influence regions for certain planar subgroups of G, and take as 3D influence region the intersection of the rectangular prisms over the 2D influence regions. We bound directly in dimension 3, with the aid of a computer program,resulting in a smaller region.
- Our understanding of cubic groups is greatly simplified by a new classification of 3D crystallographic groups given by Thurston et al. [4].

Let us briefly describe this classification. Thurston et al. first divide crystallographic groups into reducible and irreducible, were *irreducible groups* are those that do not have any invariant direction. They coincide with the cubic groups.

For an irreducible subgroup G, they define its *odd subgroup* as the one generated by the rotations of order three, and they observe that there are only two possible odd subgroups, that they denote T_1 and T_2. The odd subgroup T of a group G is normal, and so G lies between T and its normalizer $N(T)$. This is a powerful property, because it reduces the enumeration of irreducible space groups to the enumeration, up to conjugacy, of subgroups of two finite groups $N(T_1)/T_1$ and $N(T_2)/T_2$.

Hence, we study the cubic groups in two blocks. The 27 groups with odd subgroup T_1 are called "full groups" in [4], and they include all the cubic groups with reflections. The 8 groups with odd subgroup T_2 are called "quarter groups".

3 The 27 "full groups"

These are the groups between T_1 and $N(T_1)$. $N(T_1)$ is the automorphism group $I\frac{4}{m}\bar{3}\frac{2}{m}$ of the body centered cubic lattice (here and elsewhere we use the International Crystallographic Notation for crystallographic groups, see [8]). T_1 is the crystallographic group $F23$, generated by the triad rotations whose axes are the diagonals of the unit cube and by the translations of length 2 in the three edge directions of the cube. T_1 coincides the crystallographic group $F23$.

We take as fundamental subdomains of any group between T_1 and $N(T_1)$ the Delaunay tetrahedra of a body centered cubic lattice. They have two opposite perpendicular edges with length 1/2 (in relation to the fundamental translations of T_1) and four edges parallel to the four diagonals of the cube, with length $\frac{\sqrt{3}}{4}$. $N(T_1)$ has diad rotations over the first type of edges, and triad rotations over the second type of edges. The extended Voronoi region consists of five fundamental subdomains: a central one and its four neighbors. The influence region is made up of these tetrahedra plus their 10 neighbors. These 15 tetrahedra fall into two classes modulo T_1, with 11 and 4 elements.

Now let G be one of the full groups. Let s be the number of elements of G that preserve a fundamental subdomain. Then, the bound for G derived from the influence region we have calculated is $15s - 1$ or $11s - 1$ depending on whether G contains elements that exchange the two classes of subdomains or not. Since the order of $N(T_1)/T_1$ is 16, the values of s for groups other than $N(T_1)$ (which has reflections) are 1, 2, 4 and 8. This, in principle, gives a bound of 119 facets for these groups.

But we can do better. The worse this bound is the more motions we have in G, not present in T_1, that can be used to cut the extended Voronoi region further and produce better influence regions. Doing this, we get the upper bounds of the following table for the 14 full crystallographic groups without reflections.

Group	Our bound	Previous bound
$F23$	10	102
$F432$	5	21
$F\bar{4}3c$	14	44
$F\frac{2}{d}\bar{3}$	14	69
$P23$	21	102
$F4_132$	21	72
$P432$	12	15
$I23$	29	102
$P\frac{2}{n}\bar{3}$	24	79
$F\frac{4_1}{d}\bar{3}\frac{2}{n}$	29	89
$P\bar{4}3n$	29	72
$P4_232$	33	79
$P\frac{4}{n}\bar{3}\frac{2}{n}$	26	30
$I432$	26	33

4 The 8 "quarter groups"

The normalizer $N(T_2)$ of the second odd group consists of the automorphisms of the following arrangement of lines: the lines $x = y = z$, $1 + y = z = -x$, $1+z = x = -y$, and $1+x = y = -z$, together with all their translates by vectors with integer even coordinates. The odd subgroup T_2 itself is generated by the triad rotations on all these lines. The index of T_2 in its normalizer is eight. In crystallographic notation, $N(T_2)$ is $I\frac{4_1}{g}\bar{3}\frac{2}{d}$ and $T2$ is $P2_13$. There are 8 groups between (and including $N(T_2)$ and T_2. We can see their graphic representations [8] in Figures 1 to 8.

For quarter groups we choose a more complicated tesselation of space into fundamental subdomains. They are of two types, with different volumes. In type A the basic subdomain is the convex hull of the following five points: $(0, 0, 0)$, $(1/4, 0, 0)$, $(1/4, 1/4, 1/4)$, $(1/4, 1/8, 0)$ and $(1/4, 0, 1/8)$. For type B we use the convex hull of $(0, 0, 0)$, $(1/4, 1/4, -1/4)$, $(1/4, 0, 0)$, $(1/4, 0, -1/4)$, $(1/4, 1/4, 0)$, $(1/8, 0, -1/4)$ and $(1/8, 1/4, 0)$. Replicating these two bodies by the motions in $N(T_2)$ (Figure 1), tesselates space.

To calculate the extended Voronoi region we cut using the translations $(1/2, 0, 0)$, $(-1/2, 0, 0)$, $(0, 1/2, 0)$, $(0, -1/2, 0)$, $(0, 0, 1/2)$ and $(0, 0, -1/2)$, and the following list of triad rotations, which belong to T_2, hence to all the groups. The two entries in each row of the list are a point and the direction of the rotation axis:

Triad rotations

Point	Vector
$(0, 0, 0)$	$(1, 1, 1)$
$(0, 1/2, 0)$	$(-1, 1, 1)$
$(-1/2, 0, 0)$	$(-1, -1, 1)$
$(-1/2, 1/2, 0)$	$(1, -1, 1)$
$(1/2, 0, 0)$	$(-1, -1, 1)$
$(1/2, 1/2, 0)$	$(1, -1, 1)$
$(1/2, -1/2, 0)$	$(1, -1, 1)$
$(1/2, 1, 0)$	$(-1, -1, 1)$
$(0, -1/2, 0)$	$(-1, 1, 1)$
$(-1, 3/2, 0)$	$(-1, 1, 1)$
$(1, 1, 0)$	$(1, 1, 1)$
$(-3/2, -1, 0)$	$(-1, -1, 1)$
$(-1/2, -1, 0)$	$(-1, -1, 1)$
$(3/2, -1/2, 0)$	$(1, -1, 1)$
$(-1, -1, 0)$	$(1, 1, 1)$
$(-1, 0, 0)$	$(1, 1, 1)$
$(0, -1, 0)$	$(1, 1, 1)$
$(0, 1, 0)$	$(1, 1, 1)$
$(1, 0, 0)$	$(1, 1, 1)$
$(1, -1/2, 0)$	$(-1, 1, 1)$
$(1, 1/2, 0)$	$(-1, 1, 1)$
$(-1, 1/2, 0)$	$(-1, 1, 1)$

The resulting bounds with these extended Voronoi regions are in the first column of Table 1. They are not very good, the biggest being above 500. But, as in the case of full groups, the worse bounds are in groups where additional motions can be used to cut the extended Voronoi region. In particular, some of the groups have diad rotations parallel to the coordinate axes or to the diagonals of the faces of the unit cube, which we use too where we can. The list of rotations is the following, and the resulting bounds are in columns 2 and 3 of Table 1:

Diad rotations parallel to the coordinate axes

Point	Vector
$(1/2, 0, 1/4)$	$(0, 1, 0)$
$(0, 1/4, 0)$	$(1, 0, 0)$
$(1/4, 1/2, 0)$	$(0, 0, 1)$
$(0, 0, 1/4)$	$(0, 1, 0)$
$(-1/4, 1/2, 0)$	$(0, 0, 1)$
$(0, -1/4, 0)$	$(1, 0, 0)$
$(1/4, 0, 0)$	$(0, 0, 1)$
$(1/2, 0, -1/4)$	$(0, 1, 0)$
$(0, 1/4, -1/2)$	$(1, 0, 0)$
$(-1/4, 0, 0)$	$(0, 0, 1)$
$(0, 0, -1/4)$	$(0, 1, 0)$
$(0, -1/4, -1/2)$	$(1, 0, 0)$
$(0, 1/4, 1/2)$	$(1, 0, 0)$
$(0, -1/4, 1/2)$	$(1, 0, 0)$
$(0, 3/4, 0)$	$(1, 0, 0)$
$(0, 3/4, 1/2)$	$(1, 0, 0)$
$(0, 3/4, -1/2)$	$(1, 0, 0)$
$(-1/2, 0, 1/4)$	$(0, 1, 0)$
$(-1/2, 0, -1/4)$	$(0, 1, 0)$
$(-1/2, 0, -3/4)$	$(0, 1, 0)$
$(1/2, 0, -3/4)$	$(0, 1, 0)$
$(0, 0, -3/4)$	$(0, 1, 0)$
$(-1/4, -1/2, 0)$	$(0, 0, 1)$
$(1/4, -1/2, 0)$	$(0, 0, 1)$
$(3/4, 1/2, 0)$	$(0, 0, 1)$
$(3/4, 0, 0)$	$(0, 0, 1)$
$(3/4, -1/2, 0)$	$(0, 0, 1)$
$(-1/2, 0, 3/4)$	$(0, 1, 0)$
$(0, 0, 3/4)$	$(0, 1, 0)$
$(1/2, 0, 3/4)$	$(0, 1, 0)$

Point	Vector
$(3/4, 0, 3/8)$	$(-1, 1, 0)$
$(1/4, 0, 1/8)$	$(1, 1, 0)$
$(1/4, 0, 5/8)$	$(1, 1, 0)$
$(-3/4, 0, 1/8)$	$(1, 1, 0)$
$(-3/4, 0, 5/8)$	$(1, 1, 0)$
$(7/4, 0, 3/8)$	$(-1, 1, 0)$
$(3/8, 3/4, 0)$	$(0, -1, 1)$
$(1/8, 1/4, 0)$	$(0, 1, 1)$
$(5/8, 1/4, 0)$	$(0, 1, 1)$
$(1/8, -3/4, 0)$	$(0, 1, 1)$
$(5/8, -3/4, 0)$	$(0, 1, 1)$
$(3/8, 7/4, 0)$	$(0, -1, 1)$
$(0, 3/8, 3/4)$	$(1, 0, -1)$
$(0, 1/8, 1/4)$	$(1, 0, 1)$
$(0, 5/8, 1/4)$	$(1, 0, 1)$
$(0, 1/8, -3/4)$	$(1, 0, 1)$
$(0, 5/8, -3/4)$	$(1, 0, 1)$
$(0, 3/8, 7/4)$	$(1, 0, -1)$

Finally, in order to get better bounds in some of the groups, we intersect the final influence region

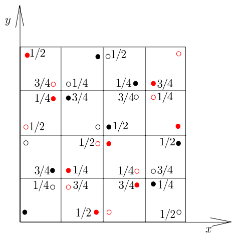


Figure 1: $I\frac{4_1}{g}\bar{3}\frac{2}{d}$

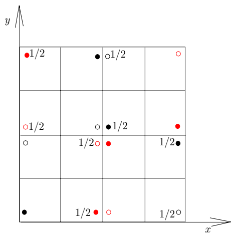


Figure 2: $I\frac{2}{g}\bar{3}$

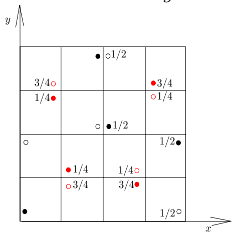


Figure 3: $I\bar{4}3d$

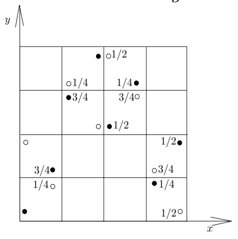


Figure 4: $I4_132$

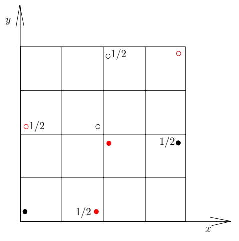


Figure 5: $P\frac{2_1}{a}\bar{3}$

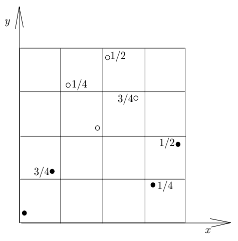


Figure 6: $P4_132 \approx P4_332$

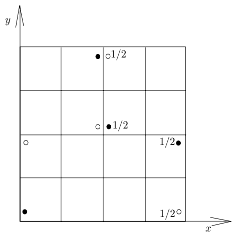


Figure 7: $I2'3$

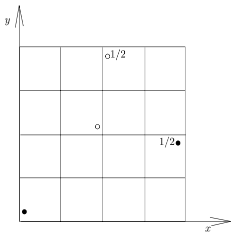


Figure 8: $P2_13$

that we obtain with the one obtained in [3] with the method of intersecting prisms over 2-dimensional regions. Our final bounds are in column four, compared tothe bounds in [3] in the last column.

Group	Our bounds				Bound from [3]
	(1)	(2)	(3)	(4)	
$I\frac{4_1}{g}\bar{3}\frac{2}{d}$	530	166	138	73	161
$I4_132$	246	94	70	53	162
$I\bar{4}3d$	276	87		74	135
$I\frac{2}{g}\bar{3}$	268	81		43	103
$P4_132$	129		105	105	89
$I2'3$	130	47		42	102
$P\frac{2_1}{a}\bar{3}$	137			73	94
$P2_13$	66			66	102

(1) Bounds after applying the triad rotations.
(2) Bounds after applying the diad rotations with axes parallel to the coordinate axes.
(3) Bounds after applying the diad rotations with axes parallel to the diagonals of the faces of the cube.
(4) Bounds after intersecting with planar projections in [3].

Table 1: Results for quarter groups.

References

[1] D. Bochiş and F. Santos. On the number of facets of 3-dimensional Dirichlet stereohedra I: groups with reflections. *Discrete Comput. Geom.* ***25:3*** (2001), 419–444.

[2] D. Bochiş and F. Santos. On the number of facets of 3-dimensional Dirichlet stereohedra II: non-cubic groups. *Preprint December 2000, revised April 2002, 27 pages.* http://arxiv.org/abs/math.CO/0204231

[3] D. Bochiş. Estereoedros de Dirichlet en 2 y 3 dimensiones. *Ph. D. Thesis, Universidad de Cantabria*, 1999.

[4] J. H. Conway, O. Delgado Friedrichs, D.H. Huson and W.P. Thurston. On Three-Dimensional Space Groups. *Contributions to Algebra and Geometry. Vol 42. No 2* (2001), 475–507.

[5] B. N. Delone (or Delaunay). A proof of the fundamental theorem of the theory of stereohedra. In *Dokl. Akad. Nauk. SSSR* ***138*** (1961), 1270–1272. (In Russian)

[6] P. Engel, *Über Wirkungsbereichsteilungen von kubischer Symmetrie*, *Z. Krist.* **154** (1981), no. 3-4, 199–215.

[7] B. Grünbaum and G. C. Shephard, Tilings with congruent tiles, *Bul. Amer. Math. Soc.* **3** (1980), 951–973.

[8] E. H. Lockwood and R. H. Macmillan. Geometric Symmetry. *Cambridge University Press*, Cambridge, 1978.

[9] J. Milnor. Hilbert's 18 problem: on crystallographic groups, fundamental domains, and on sphere packing. In *Proceedings of Symposium in Pure Mathematics of American Mathematical Society*, pages 491–507. Northern Illinois University, Dekalb, Illinois, 1974.

A Pointed Delaunay Pseudo-Triangulation of a Simple Polygon

Günter Rote[*] André Schulz[†‡]

Abstract

We present a definition of a pointed Delaunay pseudo-triangulation of a simple polygon. We discuss why our definition is reasonable. Our approach will be motivated from maximal locally convex functions, and extends the work of Aichholzer et al.[1]. Connections between the polytope of the pointed pseudo-triangulations and the Delaunay pseudo-triangulation will be given.

1 Introduction

Since the Delaunay triangulation is an important concept in computational geometry it is natural to ask for an equivalent in the pseudo-triangulation world. More precisely we are interested in a pointed pseudo-triangulation which is Delaunay-like. A pseudo-triangulation is pointed if every point is incident to an angle greater than π. Without the restriction to pointedness the Delaunay triangulation itself would be the most Delaunay-like pseudo-triangulation. In the following we skip the term *pointed* for the pointed Delaunay pseudo-triangulation.

As a simple case we consider the pseudo-triangulations of a simple polygon. For simple polygons the Delaunay triangulation must contain the boundary edges. Thus we are interested in generalizing the *constrained Delaunay triangulation* (CDT)[2] for simple polygons.

Delaunay-like is a quite soft expression. What we are looking for is a pseudo-triangulation which shares many of the properties of the CDT. For a survey for properties of the Delaunay triangulation see [4]. As the most important criterion we demand that for a convex polygon Delaunay triangulation and Delaunay pseudo-triangulation must coincide.

For the rest of the paper we assume that the points are given in general position.

[*]Institut für Informatik, Freie Universität Berlin,Takustraße 9, D-14195 Berlin, Germany, email:`rote@inf.fu-berlin.de`

[†]Supported by the Deutsche Forschungsgesellschaft (DFG) under grant RO 2338/2-1

[‡]Institut für Informatik, Freie Universität Berlin,Takustraße 9, D-14195 Berlin, Germany, email:`schulza@inf.fu-berlin.de`

2 A reasonable definition of the Delaunay pseudo-triangulations of a simple polygon

Constrained regularity is a concept which can be applied on triangulations and pseudo-triangulations. A triangulation or pseudo-triangulation is said to be *regular* if it can be represented by a downward projection of a convex lifting. Since the Delaunay triangulation is the projection of the lower convex hull of the paraboloid lifting the Delaunay triangulation is regular [3]. Moreover the constrained Delaunay triangulation is constrained regular. In [1] constrained regular pseudo-triangulations were introduced. We use this approach to find a lifting map which is similar to the paraboloid lifting.

From now on we are considering a polygon P. The n vertices of P are given in a clockwise order and named $p_1, p_2, \ldots, p_n$. A vertex is called *corner* if its internal angle is smaller than π. It is called a *reflex* vertex otherwise.

Our goal is to define the Delaunay pseudo-triangulation with help of a certain maximal locally convex function. These functions were studied in [1] and we are using the *optimality theorem* of this paper for our definition. A function is locally convex on P, if it is convex on every line segment in P. Let h_i be a given height for every vertex p_i. f^* is the maximal locally convex function which satisfies $f^*(p_i) \leq h_i$. The lifting induced by f^* is piecewise linear and projects down to a pseudo-triangulation $\mathcal{PT}(h)$. f^* and $\mathcal{PT}(h)$ are unique. If we set $h_i = |p_i|^2$ the CDT of P is $\mathcal{PT}(h)$.

The pseudo-triangulation $\mathcal{PT}(h)$ is not necessaryly pointed. Since we require pointedness we have to choose the heights in such a way that f^* will induce a pointed pseudo-triangulation. The corners of P are pointed because their outer angle is greater than π. The reflex vertices must contain the big angle inside. Thus they must part of a reflex chain of some pseudo-triangle (in [1] these vertices are called *incomplete*). The *optimality theorem* says that in this case $f^*(p_i) > h_i$. Their height is defined by the 3 corners of the pseudo-triangle which contains the reflex angle of the reflex vertex.

To assure that the reflex vertices are part of a reflex chain, we let their h_i be very high. With such a lifting it is not possible to obtain local convexity when $f^*(p_i) = h_i$ for any reflex vertex. Hence they will be pointed.

Now we discuss how an appropriate lifting will look like. We define

$$h_d := \begin{cases} f_1(p_i) & \text{if } p_i \text{ is corner in } P \\ f_2(p_i) & \text{otherwise} \end{cases}$$

For f_1 we choose $f_1(p_i) = |p_i|^2$. This guarantees that for convex polygons the Delaunay triangulation and the Delaunay pseudo-triangulation coincide. f_2 can be any strictly concave function with $\min_P(f_2(p_i)) > \max_P(f_1(p_i))$.

Lemma 1 *The maximal locally convex function in the domain of P defined by h_d projects down to a pointed pseudo-triangulation.*

Proof. Let p_c be a reflex vertex of P and let l be a line in P that crosses the neighborhood of p_c. The line l ends at the boundary of P. We name its endpoints e_1 and e_2. The point e_1 lies on the boundary segment (b_1, b_2) of P; the point e_2 on the segment (b_3, b_4). Figure 1 illustartes the situation.

We know that for $i \in \{1, 2, 3, 4\} f^*(b_i) \leq f_2(b_i)$ by the definition of f^*. Since f_2 is concave and f^* convex we know that $f^*(e_1) \leq f_2(e_1)$ and $f^*(e_2) \leq f_2(e_2)$. Thus we have $f^*(e_1) < f_2(p_c)$ and $f^*(e_2) < f_2(p_c)$ (f_2 is strictly concave). But since f^* must be convex on l we deduce that $f^*(p_c) < f_2(p_c)$. Thus $f^*(p_c) < h_c$ for all reflex vertices and as discussed above $\mathcal{PT}(h_d)$ is pointed. □

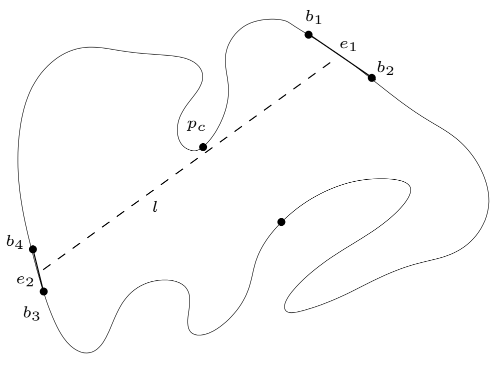


Figure 1: Illustration of the proof of Lemma 1

Since we have defined a unique pointed pseudo-triangulation by a maximal locally convex function, we are able to define the Delaunay pseudo-triangulation with help of h_d. It turns out that the height of the reflex vertices are not relevant.

Definition 1 *Let P be a simple polygon and h_d the lifting defined above. We name $\mathcal{PT}(h_d)$ the Delaunay pseudo-triangulation of P.*

Figure 2.a shows an easy example. We have a polygon with 10 vertices, 3 of them are reflex. The locally convex surface induced by the paraboloid lifting of the complete vertices is shown in Figure 3.

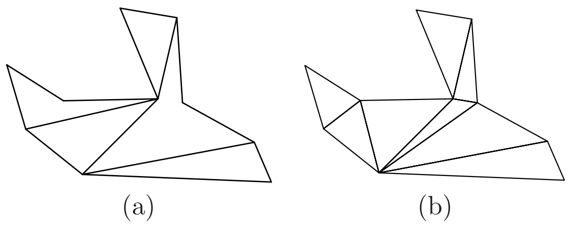


Figure 2: A Delaunay pseudo-triangulation of a polygon (a) and the CDT of the same polygon (b)

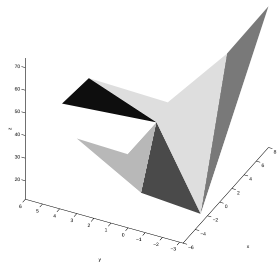


Figure 3: The lifted polygon from Figure 2

The $\mathcal{PT}(h_d)$ can be constructed by flipping to optimality [1]: We start with any triangulation of P and then flip away all reflex edges by convexifying and edge removing flips. After at most $O(n^2)$ flips we will reach the optimum. For every flip a linear system of equations must be solved. This can be done in $O(n + i^3)$ time, where i is the number of incomplete vertices. By flipping to optimality the $\mathcal{PT}(h_d)$ can be computed in $O(n^5)$.

3 Relations to the PPT Polytope

This section presents some evidence why our choice of definition is reasonable. The Delaunay triangulation can be expressed as an optimal vertex of a high-dimensional polytope. We will show a similar interpretation for the Delaunay pseudo-triangulation.

The secondary polytope of a point set S represents every regular triangulation of S as a vertex in $\mathbf{R}^{|S|}$. Moreover flips corresponds to edges in this polytope. The scalar product of a vertex of the secondary polytope and a height vector gives the volume under the lifted triangulation. If the height vector lifts the points to the paraboloid the minimization of the volume will give us the Delaunay triangulation. Recently a polytope for regular pseudo-triangulations of simple polygons was introduced [1, Lemma 9.2]. Again we can represent the volume under the lifted surface as a scalar product with some height vector. Thus $\mathcal{PT}(h_d)$ can be defined as an optimal vertex in this polytope. For practical computation this approach is not useful, because the polytope is not given by a set of inequalities.

Even more interesting is the connection between $\mathcal{PT}(h_d)$ and the polytope of pointed pseudo-triangulations (PPT Polytope) defined in [6]. For a class of simple polygons we present a formulation of $\mathcal{PT}(h_d)$ as an objective function of the PPT polytope. This gives us a completely independent way to define $\mathcal{PT}(h_d)$ as a generalization of a Delaunay triangulation for this class.

Our starting point is the objective function for the Delaunay triangulation in the PPT polytope. Of course this can only be done for points in convex position. Fortunately, for points in convex position, there is a connections between the secondary polytope and the PPT polytope (in this case the two polytopes are combinatorially equivalent)[6].

We abbreviate the the signed area of the triangle p_i, p_j, p_k with $[p_i, p_j, p_k]$ and the point $(0, 0)$ with 0. A possible formulation of the PPT polytope is the following:

$$\begin{array}{lrcl} \forall(p_i,p_j) & \langle p_i - p_j, v_i - v_j\rangle + d_{ij} & = & [p_i,p_j,0]^2 \\ & d_{ij} & \leq & 0 \end{array}$$

The PPT polytope is originally expressed by the variables v_i and lives in $\mathbf{R}^{2n}$. We added the slack variables d_{ij} to simplify further calculations.

A vertex is represented by a maximal set of inequalities which hold with equality, i.e. $d_{ij} = 0$. The edges of a pointed pseudo-triangulation of a vertex can be expressed as the set $\{(i,j) \mid d_{ij} = 0\}$.

Let $a \in \mathbf{R}^n$ a point of the secondary polytope. As discussed above, the minimization of $\sum_i a_i|p_i|^2$ leads to the Delaunay triangulation. We can express a_i in terms of d_{ij}, i.e.

$$a_i = -\frac{d_{i-1,i+1}}{[p_{i-1}, p_i, p_{i+1}]} + [p_{i-1}, p_i, p_{i+1}]$$

Let us abbreviate the signed area $[p_{i-1}, p_i, p_{i+1}]$ with E_i and let

$$c_k := -\frac{|p_k|^2}{E_k}$$

This gives us for a minimization problem the objective function

$$\textit{minimize} \qquad \sum_{i\in\{1,\ldots,n\}} c_i d_{i-1,i+1} \qquad (1)$$

over the PPT polytope.

We will now perturb the convex polygon. Convexity is not necessary anymore, but we forbid heavy deformations. More precisely we demand the following:

Definition 2 *We call a polygon P neighborly visible, if the two neighbors of any corner of P can see each other.*

Figure 4.a shows a neighborly visible polygon, Figure 4.b a polygon which is not neighborly visible.

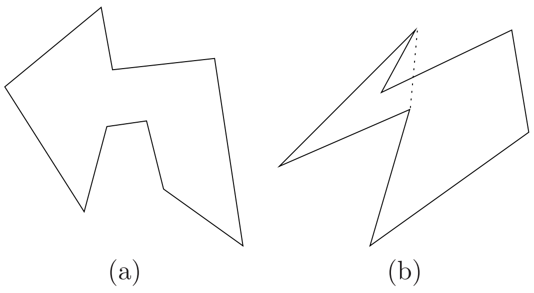


Figure 4: Examples of neighborly and not neighborly visible polygons

Theorem 2 *Let P be a neighborly visible polygon. Furthermore let all corners of P lie on its convex hull. Then minimizing the objective function (1) over the PPT polytope will induce $\mathcal{PT}(h_d)$ as solution of the linear program.*

Proof. Due to the limited space we will only sketch the proof of the theorem. We construct a pointed pseudo-triangulation of the point set P as the extension of the polygon P. For this we set all d_{ij} belonging to the boundary of P to 0. Furthermore we triangulate the concave chains of P and set the used d_{ij} to 0 if necessary. If we would allow corners which are not part of the convex hull of P, we have to add pseudo-triangles. This would immediately lead to dependencies which result in a different solution of the LP. To give an idea of the proof we first observe the

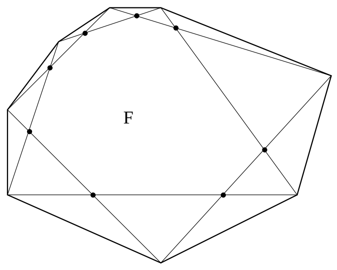


Figure 5: The construction of an equilibrium stress used in the proof

case where P is a convex polygon. The dual of the LP leads to the an stressed framework which is not in equilibrium condition. The stress for each edge is given by ω^*_{ij}. It holds

$$\forall i \;\sum_j \omega^*_{ij}(p_i - p_j) = -c_{i+1}(p_i - p_{i+2}) - c_{i-1}(p_i - p_{i-2})$$

We can adjust this stress by introducing additional arcs between every pair (p_i, p_{i+2}) (see Figure 5). Then

$$\omega_{ij} := \begin{cases} \omega^*_{ij} + c_{i+1} & \text{if } (i+2) \bmod n = j \\ \omega^*_{ij} & \text{otherwise} \end{cases}$$

forms an equilibrium stress. With help of the technique used in [5] we can turn the constructed graph into a planar one. Now we are able to apply the reverse of the Maxwell-Cremona theorem [8]. This leads to a lifting of the stressed graph. The heights of all points induced by the lifting can be calculated with help of the newly added edges and *without* knowing the Delaunay triangulation of P. First we fix the location of the central face F; then we tilt the points upwards. The stress $\omega_{i,i+2}$ gives us the information how high the point p_i will be tilted. Figure 6 gives an idea how the tilting process looks like. After some calculation it turns out that the height is exactly $|p_i|^2$. Furthermore the sign of the stress guarantees that the edges of the Delaunay triangulation span a convex surface.

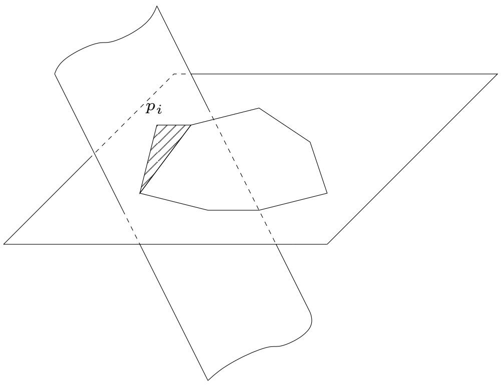


Figure 6: The tilting process

For a general neighborly visible polygon P we can now deduce the following. Since P is neighborly visible, there exists a single central face F. All corners of P will lifted to the paraboloid, because the arguments for the convex case still hold. The reflex vertices corresponds to 2 vertices in the lifting. The first one has height 0 and the second one is shifted vertically. The second vertex is the result of a tilting process outside P and belongs to the lifted pseudo-trinagulation surface. Again the pseudo-triangulation surface is convex due to the signs of the stress. Therefore the solution of the LP coincides with $\mathcal{PT}(h_d)$. □

4 Future work

Defining a Delaunay pseudo-triangulation for a simple polygon is just a first step towards a pointed Delaunay pseudo-triangulation of a general augmented polygon. Since we give a number of arguments why the definition of the Delaunay pseudo-triangulation is reasonable we are convinced that our approach can be generalized in a natural way.

One can think of using the concept of complete and incomplete vertices introduced in [1] for non-pointed Delaunay pseudo-triangulation of point sets. If we define which vertices of a point set are complete and incomplete in advance, we can define a maximal locally convex function which preserves the state of the vertices. Lifting all pointed vertices to the paraboloid would give us a Delaunay-like pseudo-triangulation. This approach needs further investigation and seems to be promising.

Since pointed pseudo-triangulations have no lifting it is completely unclear how to define a pointed Delaunay-like pseudo-triangulations. In [6] a "canonical" pseudo-triangulation has been defined as optimal vertex of the PPT polytope with the help a certain canonical objective function. This canonical pseudo-triangulation has several nice properties like an interesting lifting function for the convex case (see[7]). But it turned out that it is not a good candidate for the pointed Delaunay pseudo-triangulation. It has several properties which are not characteristic for the Delaunay Triangulation. One possible way to overcome the "non regularity" of pointed pseudo-triangulations could be to find a different objective function over the PPT polytope.

Acknowledgements
We thank Thomas Hackl for supporting us with his software package for spatial liftings of pseudo-triangulations.

References

[1] O. Aichholzer, F. Aurenhammer, H. Krasser, and P. Brass. Pseudotriangulations from surfaces and a novel type of edge flip. *SIAM J. Comput.*, 32(6):1621–1653, 2003.

[2] L. P. Chew. Constrained Delaunay triangulations. *Algorithmica*, 4(1):97–108, 1989.

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

[4] O. R. Musin. Properties of the Delaunay triangulation. In *Proceedings of the thirteenth annual symposium on Computational geometry*, pages 424–426. ACM Press, 1997.

[5] G. Rote, R. Connelly, and E. D. Demaine. Straightening polygonal arcs and convexifying polygonal cycles. Technical Report B 02-02, Discrete and Computational Geometry, February 2002.

[6] G. Rote, F. Santos, and I. Streinu. Expansive motions and the polytope of pointed pseudo-triangulations. *Discrete and Computational Geometry–The Goodman-Pollack Festschrift*, 25:699–736, 2003.

[7] A. Schulz. über die Extrempunkte des PPT Polytopes. In *Doktoranden-Workshop der FU Berlin*, number B-03-17, 2003. Technical Report.

[8] W. Whiteley. Motion and stresses of projected polyhedra. *Structural Topology*, 7:13–38, 1982.

Transforming Spanning Trees and Pseudo-Triangulations

Oswin Aichholzer[*] Franz Aurenhammer[†] Clemens Huemer[‡] Hannes Krasser[§]

Abstract

Let T_S be the set of all crossing-free straight line spanning trees of a planar n-point set S. Consider the graph $\mathcal{T}_S$ where two members T and T' of T_S are adjacent if T intersects T' only in points of S or in common edges. We prove that the diameter of $\mathcal{T}_S$ is $O(\log k)$, where k denotes the number of convex layers of S. Based on this result, we show that the flip graph $\mathcal{P}_S$ of pseudo-triangulations of S (where two pseudo-triangulations are adjacent if they differ in exactly one edge – either by replacement or by removal) has a diameter of $O(n \log k)$. This sharpens a known $O(n \log n)$ bound. Let $\widehat{\mathcal{P}}_S$ be the induced subgraph of pointed pseudo-triangulations of $\mathcal{P}_S$. We present an example showing that the distance between two nodes in $\widehat{\mathcal{P}}_S$ is strictly larger than the distance between the corresponding nodes in $\mathcal{P}_S$.

1 Introduction

Let S be a set of n points in general position in the plane (no three points of S are on a common line). We denote by T_S the set of all crossing-free straight line spanning trees of S. Several authors investigated the question of whether, and how fast, two members of T_S can be transformed into each other by means of predefined rules. Avis and Fukuda [5] considered the graph with node set T_S where two spanning trees are adjacent if they have all but one edge in common (i.e., differ by a single edge move). They showed that the diameter of this graph is at most $2n-4$. The impact of several more involved transformations, including length-reducing edge moves and so-called edge slides, has been studied in Aichholzer, Aurenhammer, and Hurtado [1]. Recently, Aichholzer and Reinhardt [4] proved that the edge slide distance between two trees in T_S is $O(n^2)$.

Define the graph $\mathcal{T}_S = (T_S, A)$ whose set of arcs A consists of pairs of trees in T_S that intersect each other only in points of S or in common edges. In other words, A contains all pairs (T, T') such that no edge of T crosses any edge of T'. The arcs of $\mathcal{T}_S$ correspond to rather powerful transformations, namely, the replacement of a tree by some 'compatible' tree. Not surprisingly, a bound of $O(\log n)$ on the diameter of $\mathcal{T}_S$ is easily obtained. A core result in [1] states that $\mathcal{T}_S$ contains a path of size $O(\log n)$ from every member of T_S to the *minimum* spanning tree of S such that tree lengths decrease along this path.

In this note we prove that the diameter of $\mathcal{T}_S$ is $O(\log k)$, where k is the number of convex layers $L_1, \ldots, L_k$ of S. That is, L_1 is the boundary of the convex hull of S, and L_i is defined recursively as the boundary of the convex hull of $S \setminus \bigcup_{j<i} L_j$, for $2 \leq i \leq k$ and $k = \min\{i \mid L_{i+1} = \emptyset\}$. We do not know whether this bound is asymptotically tight. In particular, we do not have any example where the diameter of $\mathcal{T}_S$ is not a constant.

Interestingly, the diameter of $\mathcal{T}_S$ is related to flip distances in pseudo-triangulations. A *pseudo-triangle* is a planar polygon with exactly three interior angles less than π. A pseudo-triangulation of S is a partition of the convex hull of S into pseudo-triangles whose vertex set is S. The flip graph of pseudo-triangulations, $\mathcal{P}_S$, has as its set of nodes all possible pseudo-triangulations of S. Two pseudo-triangulations are connected in $\mathcal{P}_S$ by an arc if they differ in exactly in one edge, either by replacement or removal. In other words, each arc of $\mathcal{P}_S$ corresponds to an exchanging or a removing edge flip. Aichholzer, Aurenhammer, and Krasser [3] (see also [2]) proved that the diameter of $\mathcal{P}_S$ is $O(n \log n)$. Let $\widehat{\mathcal{P}}_S$ be the induced subgraph of elements of $\mathcal{P}_S$ having exactly $2n-3$ edges (the minimum number of edges a pseudo-triangulation of S can have). The elements of $\widehat{\mathcal{P}}_S$ are called minimum, or pointed, pseudo-triangulations; each vertex of such a pseudo-triangulation is *pointed*, that is, all its incident edges lie in an angle less than π. Bereg [6] showed that the diameter of $\widehat{\mathcal{P}}_S$ is still bounded by $O(n \log n)$, and in [2] it is proved that if the diameter of $\widehat{\mathcal{P}}_S$ is $O(n)$ then the same is true for $\mathcal{P}_S$. On the other hand, the flip distance for triangulations (i.e. pseudo-triangulations having the maximum number of edges) is known to be $\Theta(n^2)$ in the worst case. In [8], Hurtado, Noy, and Urrutia refined the bound for triangulations to $O(nk)$.

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

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

[‡]Departament de Matematica Aplicada II, Universitat Politecnica de Catalunya, Barcelona, Spain. Research partially supported by Projects MCYT BFM2003-00368 and Accion Integrada Espan a Austria HU2002-0010, Huemer.Clemens@upc.edu

[§]Institute for Theoretical Computer Science, Graz University of Technology, Graz, Austria, hkrasser@igi.tugraz.at

For both graphs $\mathcal{P}_S$ and $\widehat{\mathcal{P}}_S$ it is an open problem to determine tight asymptotic bounds on the diameter. We prove that the diameter of $\mathcal{P}_S$ is $O(n \log k)$, using our result for the graph $\mathcal{T}_S$. In particular, if the diameter of $\mathcal{T}_S$ is constant then the diameter of $\mathcal{P}_S$ is $\Theta(n)$. We also demonstrate that the distance between certain nodes in $\widehat{\mathcal{P}}_S$ is strictly larger than the distance between the corresponding nodes in $\mathcal{P}_S$. A more comprehensive study of the diameters of $\mathcal{T}_S$ and $\mathcal{P}_S$ can be found in Huemer [7].

2 An upper bound on the diameter of $\mathcal{T}_S$

We show that the diameter of $\mathcal{T}_S$ can be related to the number k of convex layers of the point set S.

Observation 1 *Let Δ be a triangulation of S. Let x be a point of S which lies on some layer L_i of S, for $i \geq 2$. Then Δ contains an edge xy that does not cross L_i and such that y lies on a layer L_j with $j < i$.*

Proof. If such an edge does not exist then x must be a pointed vertex of Δ. But a triangulation does not contain pointed vertices, except on layer L_1. □

Theorem 1 *Let a point set S be given whose number of convex layers is k. The diameter of $\mathcal{T}_S$ is $O(\log k)$.*

Proof. Define a *layer tree* to be a non-crossing spanning tree of S which contains all but one edge of each layer L_i of S and which connects consecutive layers (by single edges); Figure 1 gives an example. We show below that any given spanning tree $T \in \mathcal{T}_S$ can be transformed into some layer tree using $O(\log k)$ transformations, as defined by the arcs of $\mathcal{T}_S$. This implies the theorem, because two layer trees clearly can be made to coincide by applying at most two transformations.

Consider some triangulation Δ that contains T. Due to Observation 1, for every point $x \in S \setminus L_1$ there is an edge in Δ to some layer with lower index. We select one such edge per point in $S \setminus L_1$, and in addition, all edges but one of L_1. The selected edges constitute a new spanning tree, T', of S. As T' and T live in the same triangulation, a single transformation is capable of replacing T by T'.

For a point $x \in S$, let $g_k(x)$ be the shortest path from x to a point on L_k such that $g_k(x)$ does not cross T'. As no edge of T' crosses any layer twice, $g_k(x)$ visits points on layers with increasing index. On the other hand, by Observation 1, there is a path $g_1(x)$ from x to L_1 that does not cross T' and that visits points on layers with decreasing index. Now, for all points x on layers $L_1, \ldots, L_{k/2}$ take the path $g_1(x)$, and for all points x on layers $L_{k/2}, \ldots, L_k$ take the path $g_k(x)$. The union of all these paths with L_1 is a connected graph, G. By construction, G neither crosses the tree T' nor the layer $L_{k/2}$. We select

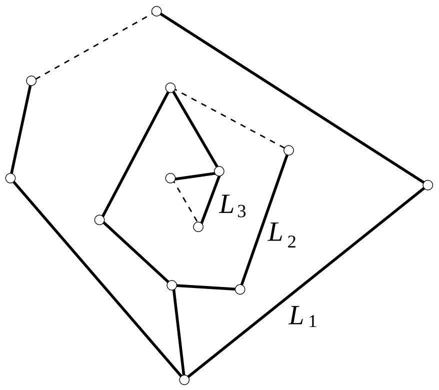


Figure 1: A layer tree for $k = 3$

from G a spanning tree, T'', which contains L_1 minus one edge. T' can be transformed into T'' in one step, and T'' can be transformed into a spanning tree T_1 that contains, in addition, $L_{k/2}$ minus one edge, in one more step.

In summary, after a constant number of transformations we arrive at two independent subproblems of size $k/2$. Therefore, with the same effort, we can transform the tree T_1 into a tree that contains, in addition to L_1 and $L_{k/2}$, from *both* layers $L_{k/4}$ and $L_{3k/4}$ all edges but one. We conclude that $O(\log k)$ tranformations suffice to generate a layer tree for S. □

3 Bounding the diameter of $\mathcal{P}_S$

Next we show that an upper bound on the diameter of $\mathcal{T}_S$ also gives an upper bound on the diameter of the flip graph $\mathcal{P}_S$ of pseudo-triangulations. We make use of a lemma from [3] on flip distances in simple polygons.

Lemma 2 *Let Q be a simple polygon with m edges. The flip distance between any two triangulations of Q is $O(m)$, if exchanging, removing, and inserting edge flips are allowed.*

Theorem 3 *Let a set S of n points be given. If the diameter of $\mathcal{T}_S$ is d then the diameter of $\mathcal{P}_S$ is $O(nd)$.*

Proof. Every pseudo-triangulation of S can be completed to a triangulation by applying $O(n)$ inserting edge flips. It thus suffices to show that any two triangulations Δ_1 and Δ_2 are connected in $\mathcal{P}_S$ by a sequence of $O(nd)$ flips. Let Δ_1 and Δ_2 contain spanning trees T_1 and T_2 of S, respectively. There is a path of length d in $\mathcal{T}_S$ which connects T_1 and T_2. We show that for consecutive trees T and T' on this path, the distance in $\mathcal{P}_S$ between any triangulation Δ containing T and any triangulation Δ' containing T' is $O(n)$.

Let us 'cut' the triangulation Δ along the edges of T. This gives several triangulated polygons. Note that no edge of T' crosses any edge of such a polygon, because T' and T are adjacent in $\mathcal{T}_S$. By Lemma 2, we can modify the triangulation within each such polygon Q_i so as to contain all the edges of T' within Q_i in $O(m_i)$ flips, if Q_i has m_i edges. Thus Δ can be transformed into a triangulation Δ'' that contains T' in $O(\sum m_i)$ flips. This sum is bounded by $n+2(n-1)$, counting the edges of L_1 plus two times the edges of T. Similarly, in a second step, we cut Δ'' along T' and transform Δ'' into the desired triangulation Δ' using another $O(n)$ flips. $\square$

Corollary 4 *The diameter of the flip graph $\mathcal{P}_S$ is bounded by $O(n \log k)$, where k is the number of convex layers of S.*

Lemma 2 also holds for pointed pseudo-triangulations [3]. Thus, we are also interested in the graph $\widehat{\mathcal{T}_S}$ of pointed spanning trees of S, where two trees are adjacent if there exists a pointed pseudo-triangulation which contains them both. If we can bound the diameter of $\widehat{\mathcal{T}_S}$ by d then the diameter of $\widehat{\mathcal{P}_S}$ is $O(nd)$, applying the argumentation of Theorem 3. Moreover, the bound $O(nd)$ carries over to the flip graph $\mathcal{P}_S$ by a result in [2].

4 Comparing distances in $\mathcal{P}_S$ and $\widehat{\mathcal{P}_S}$

In a pointed pseudo-triangulation the number of edges is minimum. This suggests that this type of pseudo-triangulation is most flexible as far as adaption by flips is concerned. In other words, one might conjecture that the distance between two nodes of $\mathcal{P}_S$ does not increase when we are required to stay within the subgraph $\widehat{\mathcal{P}_S}$ of $\mathcal{P}_S$. In the following we present an example that refutes this conjecture. Assume $n > 9$ in the following.

The underlying set S of n points is shown in Figures 2 and 3. It consists of three subsets $P = \{p_1, .., p_{n/3}\}$, $Q = \{q_1, .., q_{n/3}\}$, and $R = \{r_1, .., r_{n/3-1}\}$ in convex position, and one interior point m. The last point is chosen to lie to the left of $p_1 q_{n/3}$, of $q_1 r_{n/3-1}$, and of $r_1 p_{n/3}$. Two pointed pseudo-triangulations PT_1 (Figure 2) and PT_2 (Figure 3) are drawn on S.

Lemma 5 *The distance between PT_1 and PT_2 in $\widehat{\mathcal{P}_S}$ is at least $n-3$.*

Proof. Every pointed pseudo-triangulation of S has exactly $2n-3$ edges. PT_1 and PT_2 have the $n/3-1$ edges $r_i m$ in common, plus the $n-1$ edges of the convex hull of S. Thus $2n/3-1$ edges are different. We will show that to obtain the very first edge of PT_2 that is not in PT_1, at least $n/3-1$ (exchanging) edge flips are required. Then, when the first edge is present, there still remain at least $2n/3-2$ different edges. For each such edge, at least one flip is needed in addition. This gives a lower bound of $n-3$ flips.

PT_1 contains edges mq_i and $q_1 p_i$ which must be transformed into edges $p_{n/3} q_i$ and mp_i of PT_2. If the first edge of PT_2 that is created is of type $p_{n/3} q_i$ then this edge crosses $n/3-1$ edges $q_1 p_i$. All these edges must be replaced beforehand. If, otherwise, the first edge of PT_2 that is created is of type mp_i then either all edges mq_i or all edges mr_i must be replaced first, because the point m has to stay pointed. So, in any case, at least $n/3-1$ flips are necessary to obtain the first edge of PT_2. $\square$

Lemma 6 *The distance between PT_1 and PT_2 in $\mathcal{P}_S$ is at most $2n/3$.*

Proof. We construct a sequence of $2n/3$ flips that transforms PT_1 into PT_2. The edge flip that inserts the edge mp_1 into PT_1 is applied first. Now edge $q_1 p_i$ is flipped into edge mp_{i+1} by an exchanging flip, for $i = 1, \dots, n/3-1$. Next, edge mq_i is exchanged by edge $p_{n/3} q_{i+1}$, for $i = 1, \dots, n/3-1$. Finally, we apply the edge flip that removes $mq_{n/3}$ and gives PT_2. $\square$

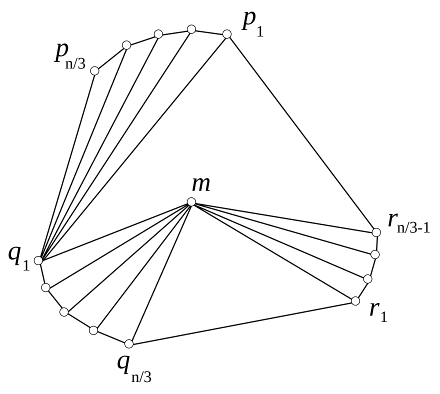


Figure 2: The pseudo-triangulation PT_1

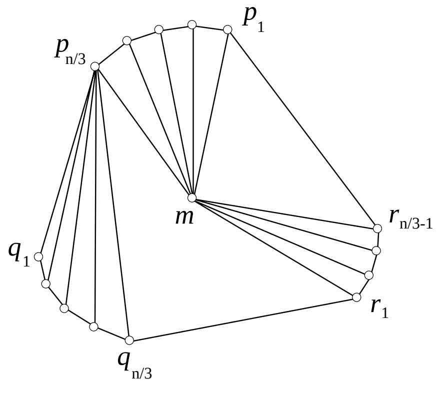


Figure 3: The pseudo-triangulation PT_2

Corollary 7 *There exist pointed pseudo-triangulations PT_1 and PT_2 whose distance in the graph $\widehat{\mathcal{P}}_S$ can only be realized by a flip sequence that affects edges common to PT_1 and PT_2.*

Proof. To see this, let the subset R in Figures 2 and 3 consist of a single point r_1. Then the edge mr_1 is common to PT_1 and PT_2. If flipping mr_1 is not allowed then at least $n-3$ flips are needed to transform PT_1 into PT_2, by the same arguments as in the proof of Lemma 5. Otherwise, we change mr_1 (in PT_1) to the edge mp_1 first. Then we apply the same sequence of $2n/3-2$ exchanging flips as in the proof of Lemma 6. Finally, $mq_{n/3}$ is changed to mr_1 which gives PT_2. This sequence consists of only $2n/3$ flips. □

Note that the reduction in flip distance in Lemma 6 and Corollary 7, respectively, stems from creating the edge mp_1. This edge is outruled in Lemma 5 by the required pointedness of m, and in Corollary 7 by being the result of flipping the common edge mr_1.

5 Conclusion and open problems

We gave a bound on the diameter of the graph $\mathcal{T}_S$ of non-crossing spanning trees and related this result to transforming pseudo-triangulations. The problem of bounding the diameter of $\mathcal{T}_S$ is also of interest on its own. So far we were not able to find two non-crossing spanning trees on the same point set S whose distance in $\mathcal{T}_S$ is more than constant.

Conjecture 1 *The diameter of $\mathcal{T}_S$ is sublogarithmic.*

We also restate the well known problem of determining the diameter of the flip graph $\mathcal{P}_S$ of pseudo-triangulations.

Conjecture 2 *The diameter of $\mathcal{P}_S$ is $\mathbf{o}(n \log n)$.*

References

[1] O. Aichholzer, F. Aurenhammer, F. Hurtado, Sequences of spanning trees and a fixed tree theorem. Computational Geometry: Theory and Applications 21 (2002), 3-20.

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

[3] O. Aichholzer, F. Aurenhammer, H. Krasser, Adapting (pseudo)-triangulations with a near-linear number of edge flips. Springer Lecture Notes in Computer Science 2748 (2003), 12-24.

[4] O. Aichholzer, K. Reinhardt, A quadratic distance bound on sliding between crossing-free spanning trees. To appear in Computational Geometry: Theory and Applications.

[5] D. Avis, K. Fukuda, Reverse search for enumeration. Discrete Applied Mathematics 65 (1996), 21-46.

[6] S. Bereg, Transforming pseudo-triangulations. Information Processing Letters 90 (2004), 141-145.

[7] C. Huemer, Flip operations for geometric and combinatorial objects. Master Thesis, Graz University of Technology, Graz, Austria, 2003.

[8] F. Hurtado, M. Noy, J. Urrutia. Flipping edges in triangulations. Discrete and Computational Geometry 22 (1999), 333-346.

Pseudo-Tetrahedral Complexes

Franz Aurenhammer and Hannes Krasser*

1 Introduction

A pseudo-triangulation is a cell complex in the plane whose cells are pseudo-triangles, i.e., simple polygons with exactly three convex vertices (so-called corners). Being an interesting and flexible generalization of triangulations, pseudo-triangulations have found their place in computational geometry; see e.g. [8, 11, 7, 1] and references therein.

Unlike triangulations, pseudo-triangulations eluded a meaningful generalization to higher dimensions so far. In this paper, we define pseudo-simplices and pseudo-simplicial complexes in d-space in a way consistent to pseudo-triangulations in the plane. Flip operations in pseudo-complexes are specified, as combinations of flips in pseudo-triangulations [11, 1], and of bistellar flips in simplicial complexes [9, 5, 4]. Our results are based on the concept of maximal locally convex functions on polyhedral domains [1], that allows us to unify several well-known structures, namely pseudo-triangulations, constrained Delaunay triangulations [3, 14], and regular simplicial complexes [2, 5]. Several implications of our results exist, and challenging open questions arise.

2 Polytopes and Corners

We give some notation concerning polytopes in d-space $\mathbb{R}^d$. A connected, bounded, and closed subset P of $\mathbb{R}^d$ is called a *d-polytope* if P is a d-manifold, with piecewise linear boundary $\mathsf{bd}\ P$ that is structured as a $(d-1)$-dimensional cell complex. The components of $\mathsf{bd}\ P$ of dimension j are called the *j-faces* of P. Faces of dimensions $d-1$, 1, and 0, respectively, are also called *facets*, *edges*, and *vertices*. We denote with $\mathsf{vert}\ P$ the set of vertices of P. P is called *simple* if P is homeomorphic to a closed ball in $\mathbb{R}^d$.

A *terminal* of P is a point $x \in P$ such that no line segment $L \subset P$ contains x in its relative interior. All terminals of P belong to $\mathsf{vert}\ P$. A terminal v of P is called a *corner* of P if there exists a hyperplane through v that has all edges of P incident to v on a fixed side. (For $d = 2$, terminals automatically fulfill this requirement and therefore are corners.) All vertices of the convex hull $\mathsf{conv}\ P$ are corners of P. So every d-polytope P has at least $d+1$ corners. In Figure 1, x is a corner, y is a terminal but not a corner, and z is not a terminal.

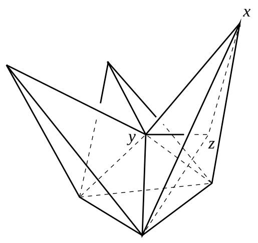


Figure 1: 3-polytope with different vertex types

A boundary-connected d-polytope with exactly $d+1$ corners is termed a *pseudo-simplex*. Every simplex is a pseudo-simplex. For $d = 2$ and $d = 3$, respectively, pseudo-simplices will be also called *pseudo-triangles* and *pseudohedra*. Our definition of a pseudo-triangle is equivalent to the classical definition, see e.g. [11, 1], which requires exactly 3 polygon vertices with an internal angle smaller than π. A *pseudo-complex* is a cell complex in $\mathbb{R}^d$ all whose cells are pseudo-simplices of dimension d. For $d = 2$, pseudo-complexes are pseudo-triangulations.

3 Local Convexity

The theory of maximal locally convex functions is the key to a derivation of pseudo-complexes. For $d = 2$, the relationship between these two concepts has been observed in [1]. Consider a real-valued function f on $\mathbb{R}^d$ whose domain is a simple d-polytope D. Function f is called *locally convex* if f is convex on each line segment $L \subset D$.

Let $\boldsymbol{h}$ be a real-valued vector that assigns a $(d+1)$st coordinate h_i (called *height*) to each vertex $v_i \in \mathsf{vert}\ D$. Our interest is in the maximal locally convex function f^* on D which fulfills $f^*(v_i) \le h_i$ for each $v_i \in \mathsf{vert}\ D$. The function f^* is unique, because f^* is the pointwise maximum of all locally convex functions which satisfy the constraints in $\boldsymbol{h}$. Moreover, f^* is continuous in the interior $\mathsf{int}\ D$ of D, by its local convexity. For $d \ge 3$, f^* need not be continuous on $\mathsf{bd}\ D$, however.

We can show that f^* is a piecewise linear function on D. In fact, f^* induces a face-to-face cell complex in D, whose cells are maximal connected subset of D where f^* is linear. The continuity of f^*

*Institute for Theoretical Computer Science, Graz University of Technology, Graz, Austria. Research supported by the FWF [Austrian Fonds zur Förderung der Wissenschaftlichen Forschung] {auren,hkrasser}@igi.tugraz.at

on int D implies that the faces of f^* have piecewise linear boundaries, and therefore are j-polytopes, for $0 \le j \le d$. As D is a simple d-polytope, the boundary of each cell of f^* is connected: The existence of a hole H in a cell contradicts the convexity of f^* on each line segment $L \subset D$ that crosses the (relative) boundary of H twice.

Each vertex x of f^* either belongs to vert D, or x is the intersection of a j-face of f^* with a $(d-j)$-face of D. Accordingly, x will be termed a *primary* or a *secondary* vertex. We extend this terminology to the cells of f^*, by distinguishing whether or not all corners of a cell are primary vertices.

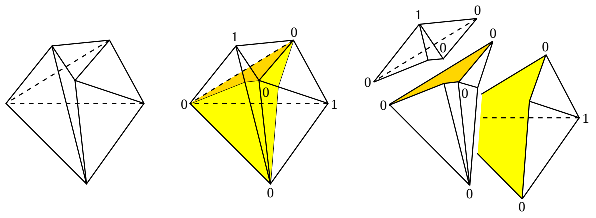


Figure 2: Splitting the Schönhardt polytope with f^*

Figure 2 illustrates a three-dimensional cell complex induced by f^* when D is the Schönhardt polytope [13]. All six vertices of D are corners. Numbers denote vertex heights. The complex consists of three non-tetrahedral primary cells. Note the occurrence of secondary vertices in the relative interior of certain edges of D.

A vertex $v_i \in$ vert D is termed *complete* if $f^*(v_i) = h_i$. A vertex of f^* that is a corner of all its incident cells is called an *allcorner*. For instance, each corner of D is an allcorner of f^*.

Lemma 1 *(a) All terminals of D are complete. (b) A vertex where f^* is discontinuous cannot be a corner of any cell. (c) Allcorners of f^* are characterized by being vertices of D that are complete and where f^* is continuous. (d) No vertex of f^* lies in* int D*, or in the relative interior of a facet of D.*

The polytope D in Figure 1 serves as an example where f^* is discontinuous. As heights for D we choose 1 for vertex y and 0 for the remaining vertices. Then $f^*(p) = 0$ for all $p \in$ int D. By Lemma 1(a), we have $f^*(y) = 1$ because y is a terminal. Thus f^* is discontinuous at y.

Maximal locally convex functions constitute a natural generalization of convex hulls. Let H be the point set in $\mathbb{R}^{d+1}$ that results from lifting vert D by its height vector $\boldsymbol{h}$. Denote by low_H the (convex) function whose graph is the lower convex hull of H, i.e., the part of bd conv H visible from $-\infty$ on the $(d+1)$st coordinate axis.

Theorem 2 *If the domain D is convex then we have $f^* = \mathsf{low}_H$, for every height vector $\boldsymbol{h}$.*

4 Pseudo-Complexes

For a given simple d-polytope D and a height vector $\boldsymbol{h}$ for vert D, let $\mathcal{PC}(D, \boldsymbol{h})$ denote the polytopal cell complex induced by f^* in D. We call $\boldsymbol{h}$ *generic* (for D) if there exists some $\epsilon > 0$ such that $\mathcal{PC}(D, \boldsymbol{h}_\epsilon) = \mathcal{PC}(D, \boldsymbol{h})$ holds for all possible ϵ-perturbations $\boldsymbol{h}_\epsilon$ of $\boldsymbol{h}$. To ease the exposition, this property of $\boldsymbol{h}$ will be implicitly assumed henceforth.

We call $\boldsymbol{h}$ *convex* if $\mathsf{low}_H(v_i) = h_i$ holds for each $v_i \in$ vert D. In particular, $\boldsymbol{h}$ is called *parabolic* if $h_i = v_i^2$ for each i. If $\boldsymbol{h}$ is convex then $\mathcal{PC}(D, \boldsymbol{h})$ shows several nice properties.

Theorem 3 *Let $\boldsymbol{h}$ be a convex height vector. Then all cells of $\mathcal{PC}(D, \boldsymbol{h})$ are primary cells, pseudo-simplices, and simple d-polytopes. All primary vertices of $\mathcal{PC}(D, \boldsymbol{h})$ are allcorners. Moreover, f^* is continuous on the entire domain D.*

By Theorem 3, locally convex functions generate pseudo-complexes if the height vector $\boldsymbol{h}$ is convex. In fact, this is the case for arbitrary $\boldsymbol{h}$; see Section 5. If D (and with it, $\boldsymbol{h}$) is convex then all cells are simplices, and $\mathcal{PC}(D, \boldsymbol{h})$ is a regular simplicial complex [2, 5] in D, by Theorem 2. If, in addition, $\boldsymbol{h}$ is parabolic then the well-known Delaunay simplicial complex [6] for D is obtained. When $\boldsymbol{h}$ is convex but D is not, then $\mathcal{PC}(D, \boldsymbol{h})$ need not be simplicial; see Figure 2. The case $d = 2$ has been treated in [1]. $\mathcal{PC}(D, \boldsymbol{h})$ then is a constrained regular pseudo-triangulation of the simple polygon D. In particular, if $\boldsymbol{h}$ is parabolic, then the constrained Delaunay triangulation [3] of D is obtained, which is the Delaunay triangulation [6] provided D is a convex polygon.

5 Bistellar Pseudoflips

We now investigate the properties of $\mathcal{PC}(D, \boldsymbol{h})$ for arbitrary height vectors $\boldsymbol{h}$, by defining flip operations in pseudo-complexes that result from controlled changes in $\boldsymbol{h}$. These operations generalize both the d-dimensional Lawson flip [9, 4] and the flips in 2-dimensional pseudo-triangulations [11, 1]. From now on, let $n = |\text{vert } D|$.

Moving Heights Let $\boldsymbol{h}_0$ and $\boldsymbol{h}_1$ be two (generic) height vectors for vert D. Assume that $\boldsymbol{h}_0$ is convex, and that $\boldsymbol{h}_0 > \boldsymbol{h}_1$ (elementwise). We continuously deform $\mathcal{PC}(D, \boldsymbol{h}_0)$ into $\mathcal{PC}(D, \boldsymbol{h}_1)$ and study the changes in the structure of cells.

To this end, let $\boldsymbol{h}_\lambda = \lambda \boldsymbol{h}_1 + (1-\lambda)\boldsymbol{h}_0$, for λ increasing from 0 to 1. By Theorem 3, in $\mathcal{PC}(D, \boldsymbol{h}_0)$ all primary vertices are complete and are allcorners, and all cells are primary and are pseudo-simplices. $\mathcal{PC}(D, \boldsymbol{h}_\lambda)$ changes its shape exactly at values λ where $\boldsymbol{h}_\lambda$ is not generic. Fix such a value λ. Consider a cell U of $\mathcal{PC}(D, \boldsymbol{h}_\lambda)$ which is not a cell of $\mathcal{PC}(D, \boldsymbol{h}_{\lambda-\varepsilon})$, for sufficiently small $\varepsilon > 0$. Denote with $\mathcal{PC}_{\lambda-\varepsilon}$ the restriction of $\mathcal{PC}(D, \boldsymbol{h}_{\lambda-\varepsilon})$ to U. The crucial observation is that, for $\lambda - \varepsilon$, f^* on $\mathsf{int}\, U$ is determined by its values at the allcorners of $\mathcal{PC}_{\lambda-\varepsilon}$. This follows from Lemma 1(b). Therefore $\mathcal{PC}_{\lambda-\varepsilon}$ has exactly $d+2$ allcorners (apart from special cases which can be avoided by perturbing $\boldsymbol{h}_0$ slightly). In particular, U has at most $d+2$ corners.

In $\mathcal{PC}(D, \boldsymbol{h}_{\lambda+\varepsilon})$, the polytope U is restructured into a cell complex $\mathcal{PC}_{\lambda+\varepsilon}$. The replacement of $\mathcal{PC}_{\lambda-\varepsilon}$ by $\mathcal{PC}_{\lambda+\varepsilon}$ is termed a *pseudoflip*.

Anatomy of Pseudoflips To study the structure of pseudoflips, let us consider any complex $PC(U, \boldsymbol{h})$ with exactly $d+2$ allcorners $v_1, \ldots, v_{d+2}$. By Lemma 1(c), full height is assumed at and only at $v_1, \ldots, v_{d+2}$. W.l.o.g., let $\boldsymbol{h}$ contain entries ∞ for all other vertices of U. Let $\boldsymbol{h}^-$ be the vector obtained from $\boldsymbol{h}$ by changing the signs of finite entries. Then, for any generic choice of heights $h_1, \ldots, h_{d+2}$ for $v_1, \ldots, v_{d+2}$, one of the complexes $PC(U, \boldsymbol{h})$ or $\mathcal{PC}(U, \boldsymbol{h}^-)$ has to arise, because the relative position of the $d+2$ points $\binom{v_i}{h_i}$ in $\mathbb{R}^{d+1}$ already determines f^*.

As a consequence, each pseudoflip can be simulated by replacing $PC(U, \boldsymbol{h})$ by $\mathcal{PC}(U, \boldsymbol{h}^-)$. Moreover, as $\mathcal{PC}(U, \boldsymbol{h}^-)$ has at most $d+2$ allcorners, the cells of $\mathcal{PC}(U, \boldsymbol{h}^-)$ are pseudo-simplices, provided the same holds for $PC(U, \boldsymbol{h})$. (If $\mathcal{PC}(U, \boldsymbol{h}^-)$ has $d+1$ allcorners then its only cell is the pseudo-simplex U.) Recalling that the original complex $\mathcal{PC}(D, \boldsymbol{h}_0)$ was a pseudo-complex, we get an inductive argument showing that the final complex $\mathcal{PC}(D, \boldsymbol{h}_1)$ is a pseudo-complex.

We face two different types of pseudoflips. An *exchanging* pseudoflip transforms $PC(U, \boldsymbol{h})$ into a complex with the same number of allcorners, whereas a *removing* pseudoflip transforms $PC(U, \boldsymbol{h})$ into a single cell. (The inverse of a removing pseudoflip is also considered a valid pseudoflip; we call it an *inserting* pseudoflip.) $PC(U, \boldsymbol{h})$ contains primary cells and, in general, also secondary cells, because secondary vertices may arise in the relative interior of faces of U. Unfortunately, neither the number of primary cells nor the number of secondary cells is bounded by a function of d. Already for $d = 3$ there are examples where $\Theta(k)$ primary cells and $\Theta(k^2)$ secondary cells occur, for $k = |\mathsf{vert}\, U|$. An upper bound for primary cells in this case is $O(k^2)$, see Theorem 4.

For general d and arbitrary complexes $\mathcal{PC}(D, \boldsymbol{h})$, the number of primary cells is bounded by Theorem 2 and the observation that this number is maximal if D is convex. The number of secondary cells can be shown to be finite but remains unclear. Concerning the total number of pseudoflips, we can show that each $(d+2)$-tuple of vertices of D gives rise to at most one pseudoflip when heights are moved as above. We conclude:

Theorem 4 *$\mathcal{PC}(D, \boldsymbol{h})$ is a pseudo-complex for arbitrary (generic) $\boldsymbol{h}$. The number of primary cells of $\mathcal{PC}(D, \boldsymbol{h})$ is $O(n^{\lceil d/2 \rceil})$. Given two height vectors $\boldsymbol{h}$ and $\boldsymbol{h}'$, the distance between $\mathcal{PC}(D, \boldsymbol{h})$ and $\mathcal{PC}(D, \boldsymbol{h}')$ by pseudoflips is $O(n^{d+2})$.*

A challenging open question is whether pseudoflip sequences between simplicial complexes do exists such that cell sizes are bounded by $O(d)$. The complexity of pseudoflips does not depend on n in this case.

Examples We illustrate some pseudoflips for $d = 3$. The 3-polytope U where a flip takes place has at most 5 corners; they are labeled by numbers in the following figures. The pseudoflips are viewed best when imagining that the height of vertex 4 is lowered.

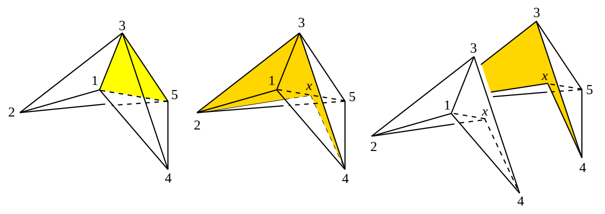


Figure 3: Exchanging pseudoflip

Figure 3 shows an exchanging pseudoflip. Before the flip, U contains the two tetrahedral cells 1235 and 1345. They are are adjacent in the triangular facet 135. (The tetrahedron 1245 avoids the interior of U and yields no cell.) After the flip, which destroys the facet 135 and creates the pseudo-triangular facet $234x$, two pseudohedra arise as cells. Their corners are $1, 2, 3, 4$ and $2, 3, 4, 5$, respectively. The secondary vertex x arises as a noncorner of both cells. All involved cells are primary cells.

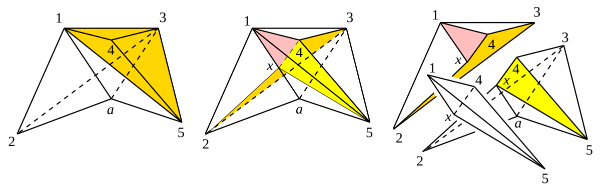


Figure 4: Pseudoflip generating a secondary cell

The exchanging pseudoflip in Figure 4 is more complicated. Again, U contains only two cells before the flip, the tetrahedron 1345 and the pseudohedron with corners $1, 2, 3, 5$ and the noncorner a. Both cells are primary cells. Their common triangular facet 135 is destroyed in the flip. After the flip, two new primary cells are present, namely, the pseudohedra with corners $1, 2, 3, 4$ and $2, 3, 4, 5$, respectively. In addition, the tetrahedron $145x$ arises as a secondary cell. Its corner x is the secondary vertex where the pseudo-triangular facet with corners $2, 3, 4$ intersects the edge $1a$.

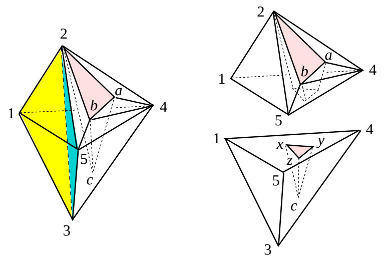


Figure 5: Pseudoflip that creates a tunnel

Figure 5 depicts an exchanging pseudoflip that creates a non-simple cell. Three primary cells are present before the flip: The tetrahedra 1234 and 1235, and the pseudohedron with corners $2, 3, 4, 5$ and non-corners a, b, c. These cells are pairwise adjacent in triangular facets which are destroyed in the flip. A single facet F with corners $1, 4, 5$ and the secondary vertices x, y, z as noncorners is created. As xyz is a hole, F is not a valid pseudo-triangle, but rather a polygonal region with three corners. Two primary cells are adjacent in F. The cell with corners $1, 2, 4, 5$ contains a tunnel, defined by the edges $2x$, ay, and bz.

6 Extensions

Several extensions of our results exist. For fixed D, consider the class of pseudo-complexes

$$\mathcal{R}(D) = \{\mathcal{PC} \mid \exists\, \boldsymbol{h} \text{ with } \mathcal{PC} = \mathcal{PC}(D, \boldsymbol{h})\}.$$

The existence of a convex n-polytope can be established that represents all the members of $\mathcal{R}(D)$. This generalizes the polytope constructions in [10] (the associahedron) and in [2] (the secondary polytope), which concern the regular simplicial complexes for convex domains D, as well as the polytope in [1], for constrained regular pseudo-triangulations of simple polygons D. In particular, the subclass

$$\mathcal{M}(D) = \{\mathcal{PC} \mid \exists\, \boldsymbol{h} \text{ convex with } \mathcal{PC} = \mathcal{PC}(D, -\boldsymbol{h})\}$$

constitutes a generalization to $\mathbb{R}^d$ of minimum (or pointed) pseudo-triangulations [11].

A pseudohedron need not be tetrahedrizable. Still, pseudo-complexes for convex height vectors provide a way of decomposing a given nonconvex polytope in 3-space into $O(n^2)$ tetrahedra, when secondary vertices are used as Steiner points.

Pseudo-complexes give rise to several burning questions. A question of major interest in motion planning applications [8] is whether visible space between polyhedral objects can be tiled and maintained with pseudo-simplices and pseudoflips, respectively. A related important problem is whether any two simplicial complexes in d-space can be transformed into each other by pseudoflips. For Lawson flips [9], the answer is negative for $d \geq 6$, and unknown for $3 \leq d \leq 5$; see [12].

Acknowledgements Thanks go to Günter Rote and Paco Santos for discussions on this topic.

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 (2003), 1621-1653.

[2] L.J.Billera, P.Filliman, B.Sturmfels. Constructions and complexity of secondary polytopes. Advances in Mathematics 83 (1990), 155-179.

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

[4] J.A.de Loera, F.Santos, J.Urrutia. The number of geometric bistellar neighbors of a triangulation. Discrete & Computational Geometry 21 (1999), 131-142.

[5] H.Edelsbrunner, N.R.Shah. Incremental topological flipping works for regular triangulations. Algorithmica 15 (1996), 223-241.

[6] 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.

[7] 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. Proc. 19th Ann. ACM Sympos. Computational Geometry, 2003, 154-163.

[8] D.Kirkpatrick, J.Snoeyink, B.Speckmann. Kinetic collision detection for simple polygons. Int'l. J. Computational Geometry & Applications 12 (2002), 3-27.

[9] C.L.Lawson. Properties of n-dimensional triangulations. Computer Aided Geometric Design 3 (1986), 231-246.

[10] C.W.Lee. The associahedron and triangulations of the n-gon. European J. Combinatorics 10 (1989), 173-181.

[11] G.Rote, F.Santos, I.Streinu. Expansive motions and the polytope of pointed pseudo-triangulations. In: Discrete & Computational Geometry - The Goodman-Pollack Festschrift, B.Aronov, S.Basu, J.Pach, M.Sharir (eds.), Algorithms and Combinatorics, Springer, Berlin, 2003, 699-736.

[12] F.Santos. A point configuration whose space of triangulations is disconnected. J. Amer. Math. Soc. 13 (2000), 611-637.

[13] E.Schönhardt. Über die Zerlegung von Dreieckspolyedern in Tetraeder. Math. Ann. 98 (1928), 309-312.

[14] J.R.Shewchuk, Updating and constructing constrained Delaunay and constrained regular triangulations by flips. Proc. 19th Ann. ACM Sympos. Computational Geometry, 2003, 181-190.

On Pseudo-Convex Decompositions, Partitions, and Coverings

Oswin Aichholzer* Clemens Huemer† Sarah Renkl‡ Bettina Speckmann§ Csaba D. Tóth¶

Abstract

We introduce pseudo-convex decompositions, partitions, and coverings for planar point sets. They are natural extensions of their convex counterparts and use both convex polygons and pseudo-triangles. We discuss some of their basic combinatorial properties and establish upper and lower bounds on their complexity.

1 Introduction

Let S be a set of n points in general position in the plane. The *convex cover number* of S, $\kappa_c(S)$, is the minimum number of convex polygons spanned by S and covering all points of S. The study of convex cover numbers is rooted in the classical work of Erdös and Szekeres [3, 4] who showed that any set of n points contains a convex subset of size $O(\log n)$. More recent results include the work by Urabe [9].

Together with convex coverings also *convex partitions* and *convex decompositions* have received much recent attention [9, 7, 5, 8, 10]. Here the *convex partition number* of S, $\kappa_p(S)$, is the minimum number of *disjoint* convex polygons spanned by S and covering all vertices of S; the *convex decomposition number* of S, $\kappa_d(S)$, is the minimum number of faces in a subdivision of the convex hull of S into convex polygons whose vertex set is exactly S.

Whether a chain of points is considered convex or reflex depends only on the point of view. Therefore, when studying convex chains and polygons contained in a set of points one might also consider reflex chains or polygons. See for example the work by Arkin et al. [2] who study questions related to convex coverings and partitions by examining the reflexivity of point sets. The 'most reflex' polygon possible is the *pseudo-triangle* which has exactly three convex vertices with internal angles less than π. A pseudo-triangle is the natural counterpart of convex polygons.

*Institute for Software Technology, Graz University of Technology, oaich@ist.tugraz.at

†Departament de Matemtica Aplicada II, Universitat Politcnica de Catalunya, huemer.clemens@upc.es. Research partially supported by Projects MCYT BFM2003-00368 and Acción Integrada España Austria HU2002-0010.

‡Department of Mathematics, TU Berlin, renkl@math.TU-Berlin.de

§Department of Mathematics and Computer Science, TU Eindhoven, speckman@win.tue.nl

¶Department of Mathematics, Massachusetts Institute of Technology, toth@math.mit.edu

In this paper we introduce pseudo-convex decompositions, partitions, and coverings which use both convex polygons and pseudo-triangles. Pseudo-convex decompositions and partitions are significantly sparser than their convex counterparts while pseudo-convex and convex coverings have asymptotically the same complexity.

Definitions. A *pseudo-triangulation* for S is a partition of the convex hull of S into pseudo-triangles whose vertex set is exactly S. A vertex is called *pointed* if it has an adjacent angle greater than π. A planar straight line graph is pointed if every vertex is pointed.

The *pseudo-convex cover number* $\psi_c(S)$ of S is the minimum number of convex polygons and/or pseudo-triangles spanned by S and covering all points of S. The pseudo-convex cover number for all sets of fixed size n is $\psi_c(n) := max_S \psi_c(S)$.

The *pseudo-convex partition number* $\psi_p(S)$ of S is the minimum number of *disjoint* convex polygons and/or pseudo-triangles spanned by S and covering all vertices of S. The pseudo-convex partition number for all sets of fixed size n is $\psi_p(n) := max_S \psi_p(S)$. Note that disjoint here implies empty (of points): neither a convex nor a pseudo-convex partition contains nested polygons.

A pseudo-convex decomposition of S is a partition of the convex hull of S into convex polygons and/or pseudo-triangles spanned by S. For instance every triangulation or pseudo-triangulations of S is a pseudo-convex decomposition. The minimum number of polygons needed for a pseudo-convex decomposition of S is the *pseudo-convex decomposition number* $\psi_d(S)$. The pseudo-convex decomposition number for all sets of fixed size n is $\psi_d(n) := max_S \psi_d(S)$.

We denote the convex cover number (and equivalently the convex partition and decomposition number) for all sets of fixed size n with $\kappa_c(n) := max_S \kappa_c(S)$.

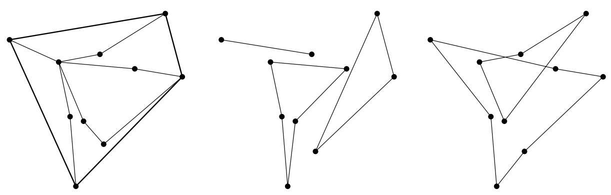


Figure 1: From left to right: Pseudo-convex decomposition, partition, and covering.

Previous work and results. The convex decomposition number $\kappa_d(n)$ is bounded by

$$n - 3 + \lfloor \sqrt{2(n-3)} \rfloor \leq \kappa_d(n) \leq \frac{10n - 18}{7}$$

(left: García-López et al. [5], right: Neumann-Lara et al. [8]). We show that the pseudo-convex decomposition number is bounded by

$$\frac{3}{5}n \leq \psi_d(n) \leq \frac{7}{10}n \ .$$

The convex partition number $\kappa_p(n)$ is bounded by

$$\left\lceil \frac{n-1}{4} \right\rceil \leq \kappa_p(n) \leq \left\lceil \frac{5n}{18} \right\rceil$$

(left: Urabe [9], right: Hosono and Urabe [7]). We show that the pseudo-convex partition number $\psi_p(n)$ is bounded by

$$\left\lfloor \frac{n}{6} \right\rfloor + 1 \leq \psi_p(n) \leq \frac{n}{4} \ .$$

The convex cover number $\kappa_c(n)$ is bounded by

$$\frac{n}{\log_2 n + 2} < \kappa_c(n) < \frac{2n}{\log_2 n - \log_2 e},$$

for $n \geq 3$ [9]. There is an easy connection between the pseudo-convex cover number and the convex cover number, namely $\psi_c(n) \leq \kappa_c(n) \leq 3\psi_c(n)$ (all points which can be covered by a pseudo-triangle can be covered by at most three convex sets). Thus both numbers have the same asymptotic behavior, which implies $\psi_c(n) \in \Theta(\frac{n}{\log n})$.

The upper bound construction for $\psi_d(n)$ depends on exact results for small point sets. These are related to a combinatorial geometry problem posed by Erdös. For $n(k) \geq 3$ find the smallest integer $n(k)$ such that any set S of $n(k)$ points contains the vertex set of a convex k-gon whose interior does not contain any points of S. Klein [3] showed that every set of 5 points contains an empty convex quadrilateral, that is $n(4) = 5$. Urabe proved in [9] that every set of 7 points can be partitioned into a triangle and a disjoint convex quadrilateral. Hosono and Urabe [7] showed that every set of 9 points contains two disjoint empty convex quadrilaterals. Harborth [6] proved that every set of 10 points contains an empty convex pentagon, that is $n(5) = 10$. We prove the following two Ramsey-type results:

Theorem 1 *Every set of* 8 *points in general position contains either an empty convex pentagon or two disjoint empty convex quadrilaterals.*

Theorem 2 *Every set of* 11 *points in general position contains either an empty convex hexagon or an empty convex pentagon and a disjoint empty convex quadrilateral.*

Both results were established with the help of the order type data base [1]. In the full paper we also provide a surprisingly intuitive geometric proof of Theorem 1 that requires only a moderate number of case distinctions.

Furthermore, we establish some basic combinatorial properties of $\psi_d(n)$, $\psi_p(n)$, and $\psi_c(n)$ and we also prove that $\psi_d(n)$ is monotonically increasing.

2 Basic Properties

Our first (trivial) observation is that $\psi_d(n) \leq \kappa_d(n)$, $\psi_p(n) \leq \kappa_p(n)$, and $\psi_c(n) \leq \kappa_c(n)$. It is well known that $\kappa_c(n) \leq \kappa_p(n) \leq \kappa_d(n)$. For pseudo-convex faces we trivially have $\psi_c(n) \leq \psi_p(n)$. $\psi_p(n) \leq \psi_d(n)$ follows from the bounds given in the previous section.

Next we observe that $\psi_d(n+1) \leq \psi_d(n)+1$, $\psi_p(n+1) \leq \psi_p(n) + 1$, and $\psi_c(n+1) \leq \psi_c(n) + 1$. This follows by induction when inserting the points in x-sorted order. For covering and partitioning the last inserted vertex is a singleton, for decomposing it forms a corner of a pseudo-triangle similar to the last step in a Henneberg construction.

The following lemma establishes an interesting connection between the convex partition number and the pseudo-convex decomposition number.

Lemma 3 *For any point set S we have $\psi_d(S) \leq 3\kappa_p(S) - 2$ and thus $\psi_d(n) \leq 3\kappa_p(n) - 2$.*

Table 1 shows the exact values of $\psi_c(n)$, $\psi_p(n)$, and $\psi_d(n)$ for small sets of points. There is one intriguing open case: $\psi_p(13) \in \{3, 4\}$: $\psi_p(13) = 3$ would imply an improved upper bound of $\psi_p(n) \leq 3n/13$.

The pseudo-convex decomposition, partition, and covering numbers for a particular point set S are not necessarily monotone. Consider the examples in Figure 2: (left) A set S with 9 points and $\psi_d(S) = 3$. Removing the bottom most point of S results in a set S' with 8 points and $\psi_d(S') = 4$. (right) A set S with 6 points and $\psi_c(S) = \psi_p(S) = 1$. Removing the topmost point of S results in a set S' with 5 points and $\psi_c(S') = \psi_p(S') = 2$.

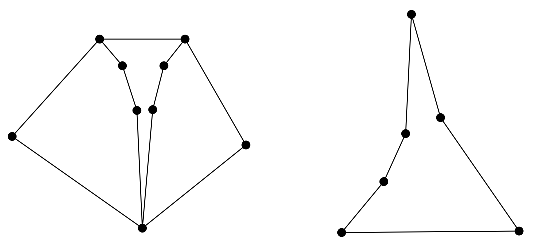


Figure 2: Sets with non-monotone behavior.

3 Pseudo-Convex Decompositions

We first give a formula for the number of faces in a pseudo-convex decomposition:

n	3	4	5	6	7	8	9	10	11	12	13	14	15
$\psi_c(n)$	1	1	2	2	2	2	2	2..3	2..3	2..3	2..3	2..3	2..4
$\psi_p(n)$	1	1	2	2	2	2	3	3	3	3	3..4	3..4	4
$\psi_d(n)$	1	2	2	3	4	4	5	6	6	7	8	8..9	8..9

Table 1: Bounds on the pseudo-convex cover number $\psi_c(n)$, partition number $\psi_p(n)$, and decomposition number $\psi_d(n)$ for small point sets.

Lemma 4 *Let S be a set of n points in general position. Let P be a pseudo-convex decomposition of S, n_k the number of convex k-gons in P, and p the number of pointed vertices. Then the number of faces of P is*

$$|P| = 2n - p - 2 - \sum_{k=4}^{n} n_k(k-3)$$

Corollary 5 *The number of faces in a pointed pseudo-convex decomposition is*

$$|P| = n - 2 - \sum_{k=4}^{n} n_k(k-3)$$

Although the pseudo-convex decomposition number for a particular point set S might not be monotone (recall Figure 2), $\psi_d(n)$ nevertheless increases monotonically with n.

Theorem 6 *The pseudo-convex decomposition number increases monotonically with the number of points.*

Proof. We have to show that $\psi_d(n) \leq \psi_d(n+1)$ which is equivalent to show that for all point sets S, $|S| = n$, $\psi_d(S) \leq \psi_d(n+1)$ holds. So let S be some point set with n vertices and let $q \in S$ be an extreme point of S. We place a new vertex q^+ arbitrarily close to q to get the set $S^+ = S \cup q^+$ such that both, q and q^+, are extreme vertices of S^+. Note that $S^+ \setminus q$ has the same order type as S, that is, for any two points $p_1, p_2 \in S \setminus q$ the triples p_1, p_2, q and p_1, p_2, q^+ have the same orientation.

As S^+ has $n+1$ points it can be pseudo-decomposed with at most $\psi_d(n+1)$ faces. Let D^+ be such a decomposition. Note that the face F of D^+ which contains the edge qq^+ has to be convex, as otherwise q and q^+ would lie on different sides of at least one edge of the pseudo-triangle F. Now contract the edge qq^+ until q and q^+ coincide. By this transformation the face F loses one edge, but all other faces of D^+ remain combinatorially unchanged, that is, either convex polygons or valid pseudo-triangles. Thus we obtain a pseudo-decomposition D of S which has either the same number of faces as D^+ or, in the case that F was a triangle, one less. Therefore $\psi_d(S) \leq \psi_d(S^+) \leq \psi_d(n+1)$. $\square$

The general lower bound construction as well as a detailed analysis of upper and lower bounds for small point sets can be found in the full paper.

3.1 Upper Bound

Our upper bound construction is based on exact bounds for small point sets. Assume that we are given a set S with n points and that we know the value of $\psi_d(k)$ for some $k < n$. We choose an extremal point p. Now we take a line through p that has the whole point set on one side and perform a circular sweep around p, splitting off point sets of size $k-1$. Together with p each of these *petals* contains k points. We have a total of $\frac{n}{k-1}$ petals which each can be decomposed into at most $\psi_d(k)$ faces. Two adjacent petals can be combined with a pseudo-triangle into one larger convex set. We apply this method until all of them are combined and so obtain an upper bound of

$$\psi_d(n) \leq \frac{\psi_d(k)+1}{k-1} n \; .$$

The best current upper bound can be achieved by combining Theorem 2 with Corollary 5. We construct a decomposition for $k = 11$ points by pseudo-triangulating in a pointed way around the convex polygons guaranteed by Theorem 2. Then Corollary 5 states that $\psi_d(11) = 11 - 2 - 3 = 6$ if the point set contains an empty convex hexagon and $\psi_d(11) = 11 - 2 - 1 - 2 = 6$ if the point set contains an empty convex pentagon and a disjoint empty convex quadrilateral. This implies

$$\psi_d(n) \leq \frac{\psi_d(11)+1}{11-1} n = \frac{6+1}{10} n = \frac{7}{10} n \; .$$

4 Pseudo-Convex Partitions

An upper bound of $\psi_p(n) \leq n/4$ can be easily established: Any four points form either a pseudo-triangle or a convex quadrilateral and grouping them in x-sorted order guarantees disjointness.

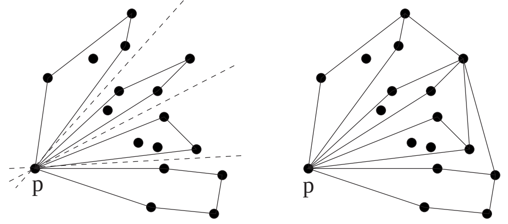


Figure 5: Petals of size 5.

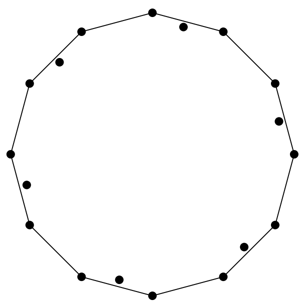


Figure 3: A point set S with $\psi_p(S) = \frac{n}{6} + 1$.

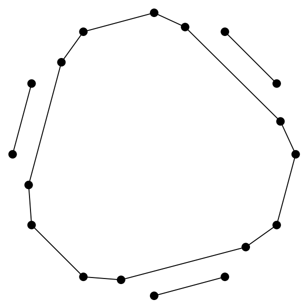


Figure 4: A partition of S consisting of $\frac{n}{6}+1$ faces.

4.1 Lower Bound

Lemma 7 *Let S be set of points in convex position with $|S| = 2m$, $m \geq 1$. We partition S into m pairs of consecutive points (along the convex hull). Let $\psi'_p(S)$ denote the minimum number of faces in a pseudo-convex partition of S in which no face contains a pair. Then $\psi'_p(S) = \lceil \frac{m}{2} \rceil + 1$.*

Theorem 8 $\psi_p(n) \geq \lfloor \frac{n}{6} \rfloor + 1$, $n \geq 3$.

Proof. We consider the point set S shown in Figure 3. S contains $\lfloor \frac{n}{3} \rfloor$ interior points which are placed very close to every second convex hull edge. Let P be a pseudo-convex partition of S using the minimum number $\psi_p(S)$ of faces. We say that two consecutive points p and q of the convex hull form a *pair* if there is an interior point close to the edge pq. There are $\lfloor \frac{n}{3} \rfloor$ such pairs. We partition the faces of P into two classes: Class A denotes faces containing at least one pair. Class B consists of faces of P containing no pair. Observe that a face of class A contains points of at most two pairs. Thus, when drawing the faces of A, there remain at least $\lfloor \frac{n}{3} \rfloor - 2|A|$ unused pairs. There might also remain additional interior points and other convex hull points. Since all faces of B are convex removing these additional points from the optimal partition P only can decrease the number of faces of B. Hence, the number of faces of B is at least the number of faces needed for the $\lfloor \frac{n}{3} \rfloor - 2|A|$ remainng unused pairs. By Lemma 7 we need at least $(\lfloor \frac{n}{3} \rfloor - 2|A|)/2+1$ faces for these pairs. Hence, $|B| \geq (\lfloor \frac{n}{3} \rfloor - 2|A|)/2+1$, and $\psi_p(S) = |A|+|B| \geq |A|+\lfloor \frac{n}{6} \rfloor - |A|+1 \geq \lfloor \frac{n}{6} \rfloor + 1$. A partition of S consisting of $\frac{n}{6}+1$ faces is shown in Figure 4. □

Note: In the proof of Theorem 8 we did not use ceiling and floor functions to their utmost limit to simplify the resulting formula. For example with $n = 9$ we get from Lemma 7 a lower bound of $2 + |A|$ and thus $\psi_p(9) \geq 3$. Similar we get $\psi_p(15) \geq 4$.

Acknowledgements. The first two authors want to thank Ferran Hurtado and Hannes Krasser for valuable discussions on the presented subject.

References

[1] O. Aichholzer, F. Aurenhammer, and H. Krasser. Enumerating order types for small point sets with applications. In *Proc. 17th Annu. ACM Sympos. Comput. Geom.*, pages 11–18, 2001.

[2] E. M. Arkin, S. P. Fekete, F. Hurtado, J. S. B. Mitchell, M. Noy, V. Sacristán, and S. Sethia. On the reflexivity of point sets. In B. Aronov, S. Basu, J. Pach, and M. Sharir, editors, *Discrete and Computational Geometry: The Goodman-Pollack Festschrift*, volume 25, pages 139–156. Springer-Verlag, 2003.

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

[4] P. Erdös and G. Szekeres. On some extremeum problem in geometry. *Ann. Univ. Sci. Budapest*, 3-4:53–62, 1960.

[5] J. García-López and M. Nicolás. A counterexample about convex partitions. In *IV Jornadas de Matemática Discreta y Algorítmica*, page 213, 2004.

[6] H. Harborth. Konvexe fünfecke in ebenen punktmengen. *Elemente der Mathematik*, 33(5):116–118, 1978.

[7] K. Hosono and M. Urabe. On the number of disjoint convex quadrilaterals for a planar point set. *Computational Geometry: Theory and Applications*, 20:97–104, 2001.

[8] V. Neumann-Lara, E. Rivera-Campo, and J. Urrutia. A note on convex decompositions of a set of points in the plane. *Graphs and Combinatorics*, 20(2):223–231, 2004.

[9] M. Urabe. On a partition into convex polygons. *Discrete Applied Mathematics*, 64:179–191, 1996.

[10] J. Urrutia. Algunos problemas abiertos. In *Actas de los IX Encuentros de Geometría Computacional, Girona, España*, pages 13–23, 2001. English version: `http://www.matem.unam.mx/~urrutia/online_papers/latimopOL.pdf`.

Pointed Binary Encompassing Trees: Simple and Optimal

Michael Hoffmann[*] Csaba D. Tóth[†]

Abstract

For n disjoint line segments in the plane we construct in optimal $O(n \log n)$ time an encompassing tree of maximal degree three such that every vertex is *pointed*. Moreover, at every segment endpoint all incident edges lie in a halfplane defined by the incident input segment.

1 Introduction

Interconnection graphs of disjoint line segments in the plane are fundamental structures in computational geometry. Often more complex objects can be modeled by their boundary segments or polygons. One particularly well-studied example is a crossing-free spanning graph: the *encompassing graph* for disjoint line segments in the plane is a connected planar straight line graph (Pslg) whose vertices are the segment endpoints and that contains every input segment as an edge. One well-known example of encompassing graphs are constrained (Delaunay) triangulations [9].

Bose et al. [4, 3] showed that any finite set of disjoint line segments in the plane admits an encompassing tree of maximum degree *three*. Moreover, they gave an $O(n \log n)$ time algorithm to construct such an encompassing tree for n given segments. Both the degree bound and the runtime are best possible (the latter in the algebraic computation tree model).

Hoffmann, Speckmann, and Tóth [5] extended the result of Bose et al. by showing that for n disjoint segments in the plane a *pointed* binary encompassing tree can be constructed in $O(n^{4/3} \log n)$ time. A Pslg is *pointed* iff for every vertex v all edges incident to v lie in a halfplane whose boundary contains v.

Here, we improve this result in several aspects: We construct an encompassing tree in optimal $O(n \log n)$ time and guarantee a stronger sense of pointedness where all edges incident to a vertex v lie in a halfplane aligned with the input segment whose endpoint is v. As an additional benefit, the presented algorithm is also considerably simpler (to understand and to implement) than the existing approach.

Theorem 1 *Let S be a set of n disjoint line segments in the plane. There exists an encompassing tree $T(S)$ of maximum degree three such that for every vertex v all incident edges lie in a halfplane bounded by the line through the segment from S that is incident to v. Moreover, $T(S)$ can be constructed in $O(n \log n)$ time*

Motivation *Pointed* Pslgs are closely related to minimum pseudo-triangulations, which have numerous applications in motion planning [11], kinetic data structures [8], collision detection [1], and guarding [10]. Streinu [11] showed that a minimum pseudo-triangulation of V is a pointed Pslg on the vertex set V with a maximal number of edges. As opposed to triangulations, there is always a bounded degree pseudo-triangulation of a set of points in the plane [7]. A bounded degree pointed encompassing tree for disjoint segments leads to a bounded degree pointed encompassing pseudo-triangulation, due to a result of Aichholzer et al. [2].

A simple construction (Figure 1a) shows that not every set of n disjoint segments in the plane admits an *encompassing path*. But there is always a path that encompasses $\Theta(\log n)$ segments and does not cross any other input segment [6].

2 Definitions

Polygons. A *polygon* P is a sequence $(p_1, p_2, \ldots, p_k)$ of points in the plane. Denote the set of vertices of P by $V(P) = \{p_1, p_2, \ldots, p_k\}$, and the set of edges by $E(P) = \{p_1p_2, p_2p_3, \ldots, p_{k-1}p_k, p_kp_1\}$.

A *weakly simple polygon* is a polygon without self-crossings. Any weakly simple polygon P partitions $\mathbb{R}^2 \setminus P$ into an interior and exterior.

The boundary of every simply connected polygonal set D can be covered by a weakly simple polygon ∂D. In particular, every planar straight line tree A can be covered by a weakly simple polygon ∂A. Note, however, that a vertex of the tree A can occur several times among the vertices of ∂A. One way to distinguish distinct occurrences of the same point along ∂A is by the *angles* (three consecutive vertices) along ∂A.

Faces of a PSLG. The complement of a connected Pslg A can have several connected components, which we call the *face*s of A. The boundary of each face F can be covered by a weakly simple polygon ∂F. We say that a vertex v_i of the weakly simple polygon ∂F is convex (reflex) if the angle $\angle v_{i-1}v_iv_{i+1}$

[*]Theoretical Computer Science, ETH Zürich, hoffmann@inf.ethz.ch

[†]Department of Mathematics, MIT, Cambridge, toth@math.mit.edu

whose angular domain contains F is less than (more than) 180°. This angle is the exterior angle of ∂F for the outer face, and the interior angle of ∂F for all bounded faces.

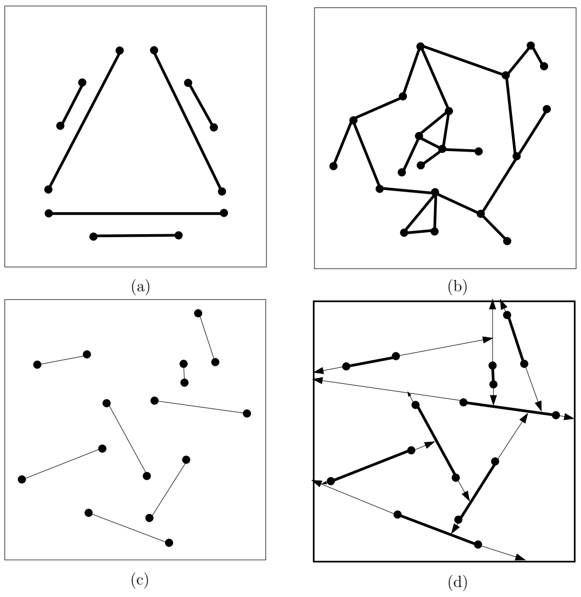


Figure 1: Six segments not admitting an encompassing path (a), a connected PSLG with 4 faces including the outer face (b), disjoint segments (c), and their convex partition (d).

Convex partition and cells. The free space around n disjoint line segments in the plane can be partitioned into $n+1$ convex cells by the following well known partitioning algorithm. (For simplicity, we assume that no three segment endpoints are collinear.) For every segment endpoint p of every input segment s_p, extend s_p beyond p until it hits another input segment, a previously drawn extension, or to infinity. There may be many different partitions depending of the order in which we consider the segment endpoints, but the number of convex cells is always $n+1$.

3 Tunnel Graphs

Consider a set of disjoint segments S in the plane and a convex partition $P(S)$ obtained by the above algorithm. Let us assign every segment endpoint p to an incident cell $\tau(p)$ of the partition. We define the *tunnel graph* $T(S, P(S), \tau)$ for S, a partition $P(S)$, and an assignment τ as follows: The nodes of T correspond to the convex cells of $P(S)$. Two nodes a and b are connected by an edge iff there is a segment $pq \in S$ such that $\tau(p) = a$ and $\tau(q) = b$. The tunnel graph is clearly planar; and T has $n+1$ nodes and n edges, therefore it is connected iff it is a tree.

Theorem 2 *For any set S of n disjoint line segments, we can construct in $O(n \log n)$ time a convex partition $P(S)$ and an assignment τ such that the tunnel graph $T(S, P(S), \tau)$ is a tree.*

We note that the choice of the convex partition is important in Theorem 2: Figure 2(d) shows seven disjoint line segments and a convex partition such that there is no assignment for which the tunnel graph is connected. We obtain Theorem 1 as a corollary of Theorem 2.

Proof of Theorem 1. Consider a partition $P(S)$ and an assignment τ provided by Theorem 2. We construct a binary encompassing tree as follows: In each cell connect all segment endpoints assigned to it by a simple path; for example, connect them in the order in which they appear along the boundary of the cell.

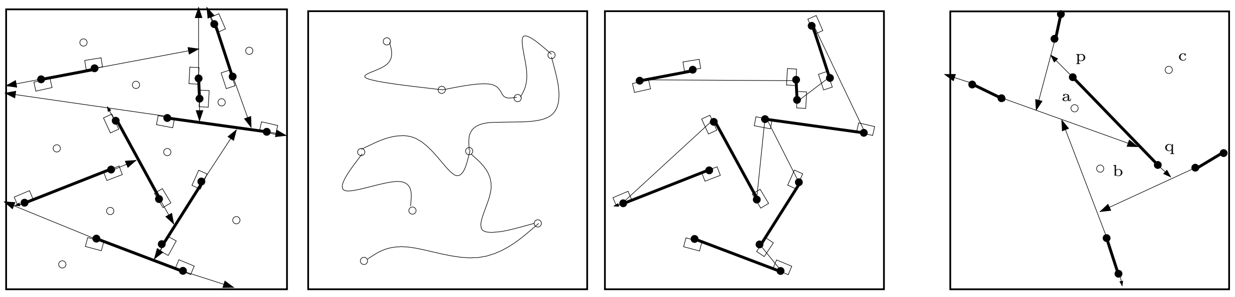
The resulting graph is clearly a PSLG that encompasses the input segments. The maximal degree is three because we add at most two new edges at every segment endpoint. It remains to prove connectivity. Let p and r be two segment endpoints. We know that the tunnel graph is connected, so there is an alternating sequence of cells and segments $(a_1 = \tau(p), p_1q_1, a_2, \ldots, p_{k-1}q_{k-1}, a_k = \tau(r))$ such that $\tau(p_i) = a_i$ and $\tau(q_i) = a_{i+1}$, for every i. As all segment endpoints assigned to the same cell are connected, this path corresponds to a path in the encompassing graph. □

4 Constructing the Convex Partition

This section is devoted to the proof of Theorem 2. Given n disjoint line segments in the plane, we partition the free space around the segments into $n+1$ convex cells and we assign an incident cell to every segment endpoint in $O(n \log n)$ time.

Let R be a bounding box of the input segments. We construct a convex partition of the free space in two phase line sweep algorithm. In the first phase, we apply a left-to-right sweep: We extend every input segment beyond its right endpoint until the extension hits another segment, another extension, or the boundary of R. If two extensions meet, then an arbitrary one continues and the other one ends.

The free space of the input segments and their right extensions is a simply connected set $C_0 \subset R$. Order the segments $s_1, \ldots, s_n$ according to the order of their left endpoint along ∂C_0. Let p_i (q_i) denote the left (right) endpoint of s_i. We extend every input segment s_i, $i = 1, 2, \ldots, n$, in this order, beyond its left endpoint p_i until the extension hits another segment, another extension, or the boundary of R. Let γ_i denote the left extension of s_i. Every segment γ_i recursively partitions a cell of our cell complex into two subcells. Notice that all left extensions γ_i can be constructed in a single right-to-left sweep which gives priority to the segment of smaller index whenever two



(a) Partition. (b) Tunnel Graph. (c) Encompassing Tree. (d) Disconnected.

Figure 2: An example for a partition with an assignment (a), the corresponding tunnel graph (b), the resulting tree (c). A partition for which no assignment gives a connected tunnel graph (d).

extensions meet. Thus we can compute a convex partition P by two line sweeps in $O(n \log n)$ time.

For constructing the assignment τ, we assume that all right extensions are in place, and we insert the left extensions one by one (even though we have pre-computed all left extensions in a single sweep). We define the assignment τ on the endpoints of s_i, $i = 1, 2, \ldots, n$, as soon as γ_i has been inserted. When the first $i-1$ left extensions have been inserted, we have a partition P_{i-1} into i cells and a partial assignment τ_{i-1} on the endpoints of the first $i-1$ segments. P_{i-1} and τ_{i-1} define a tunnel graph T_{i-1} on i nodes. We choose the assignment at the endpoints of s_i inductively such that T_i (a graph with $i+1$ nodes) remains connected.

Assume that γ_i splits a cell C_i of P_{i-1} into two cells $C'_i, C''_i \in P_i$. The node $v(C) \in T_{i-1}$ corresponding to cell C is split into two nodes C' and C'', which lie in different components of the resulting graph T'_{i-1}. The left endpoint p_i of s_i is incident to both C'_i and C''_i because $p_i \in \gamma_i$. The right endpoint q_i, however, may be incident to neither C'_i nor C''_i. We always assign q_i to the cell lying above q_i. We can always assign p_1 to C'_i or C''_i, whichever lies in the other component of T'_{i-1} as $\tau(q_i)$, thus ensuring that T_i is a tree.

We have shown that there exists an assignment $\tau = \tau_n$ for which the tunnel graph $T = T_n$ is connected. It remains to prove that such an assignment can be computed in $O(n \log n)$ time. That is, we need to decide efficiently whether two cells are in the same connected component of the current tunnel graph T_i.

Data structure. For each cell C of P_{i-1}, we maintain a doubly linked list of all segment endpoints and vertices along ∂C. The assignments τ_i carries one bit information for each segment endpoint r: It assigns r to the cell lying *below* or *above* r. We can insert a splitting segment γ_i by splitting the doubly connected list of of C_i into C'_i and C''_i in constant time. We also note an interval $g(v) \subset [1, n]$ for each vertex v of the right extension tree such that the descendants of v contain every left segment endpoint p_j, $j \in g(v)$. We maintain a *coloring* on the segments and their left and right extensions: Every input segment and every right extension is *blue*. The color of right extensions is defined recursively: γ_i is blue if its left endpoint hits a blue segment, otherwise it is *red*. We also maintain an index ind(e) for every blue input segment or blue extension. The index of s_i or its right extension is i. If γ_i hits a segment of index j then ind$(\gamma_i) = j$.

Assignment rule. We assign p_i according to the following rule: If γ_i is blue and $v_i \notin s_i$ where v_i is the deepest vertex in the right extension tree such that $[\text{ind}(\gamma_i), i] \subseteq g(v_i)$, then we assign p_i to the cell above it, otherwise to the cell below it. It takes $O(\log n)$ time to find v_i in the right extension tree, and so $\tau(p_i)$ can be computed for all $i = 1, 2, \ldots, n$ in $O(n \log n)$ time.

Proposition 3 *Choosing $\tau(p_i)$ by the above rule maintains the connectivity of T_i*

Proof. We define an orientation on the input segments and their extensions. Every segment and every right extension is directed to the right, every left extension is directed to the left. Note that there are no cycles in this orientation. For every $i = 1, 2, \ldots n$, we define a curve β_i through p_i: two branches of β_i start out from p_i to the left along γ_i and to the right along s_i, they follow the above orientation until the two branches meet or until both hit the bounding box R. Curve β_i partitions R into two *regions* A_i and B_i such that p_i lies on their common boundary. Observe that the curve does not pass through any left segment endpoint, and recall that every right segment endpoint q_j, is assigned to the region above q_j.

By a case analysis, we can verify that s_i is the only segment whose left and right endpoints are assigned to regions A_i and B_i, respectively, and the assignment rule assigns p_i and q_i to distinct regions. (1) If γ_i is red then β_i is x-monotone and its two branches pass

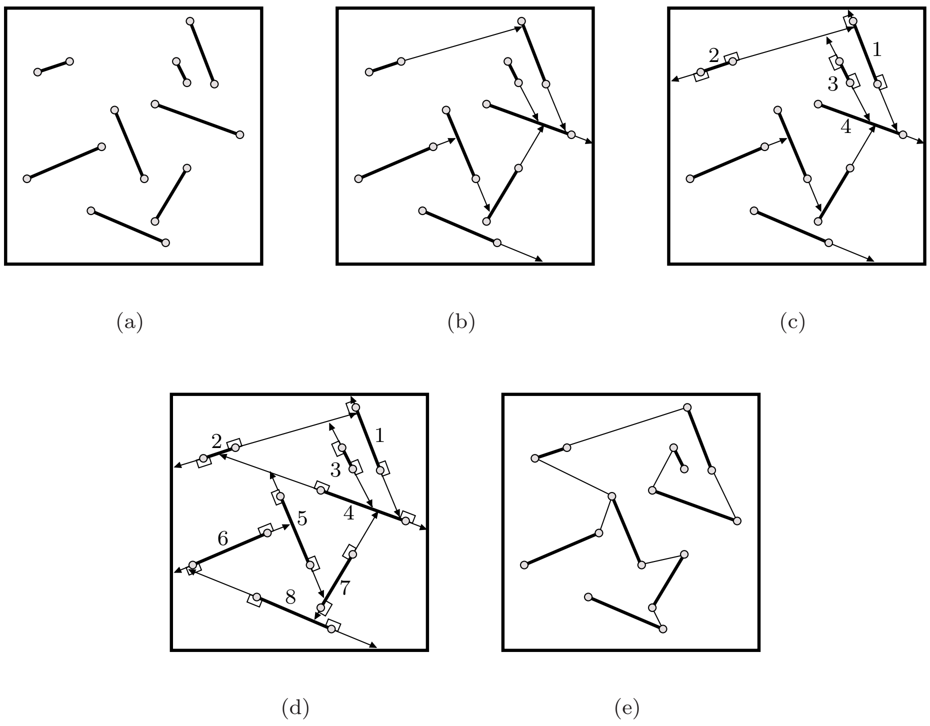


Figure 3: Constructing the partition: First all right extensions (b), then the left extensions are inserted one by one (c) and (d), and from the final partition together with the assignment we can construct the encompassing tree.

through right endpoints only, so p_i is the only vertex that might be assigned to the region below β_i. (2) Suppose that γ_i is blue: The left branch of β_i starts out x-monotone decreasing, then it hits a segment or a right extension and turns back in x-monotone increasing direction. Let A_i be the region not adjacent to the left side of R. A_i is an x-monotone region. (2a) If γ_i hits a segment s_j, $j > i$, or its right extension, then A_i must be below s_i. We know that B_i is above q_i and A_i is below p_i. (2b) If γ_i hits a segment s_j, $j < i$, or its blue extension, then A_i is above s_i, so we know that A_i is above p_i. The rightmost point of A_i is v_i. The only case where A_i does not lie above q_i is that $v_i \in s_i$. □

Acknowledgments We thank Bettina Speckmann for several helpful discussions.

References

[1] P. K. Agarwal, J. Basch, L. J. Guibas, J. Hershberger, and L. Zhang, Deformable free space tilings for kinetic collision detection, in *Proc. 4th WAFR*, 2001, 83–96.

[2] O. Aichholzer, M. Hoffmann, B. Speckmann, and Cs. D. Tóth, Degree bounds for constrained pseudo-triangulations, in: *Proc. 15th CCCG*, 2003, pp. 155–158.

[3] P. Bose, M. E. Houle, and G.T. Toussaint, Every set of disjoint line segments admits a binary tree, *Discrete Comput Geom.* **26** (2001), 387–410.

[4] P. Bose and G. T. Toussaint, Growing a tree from its branches, *J. Algorithms* **19** (1995), 86–103.

[5] M. Hoffmann, B. Speckmann, and Cs. D. Tóth, Pointed binary encompassing trees, in *Proc. 9th SWAT*, vol. 3111 of LNCS, Springer-Verlag, 2004, pp. 442–454.

[6] M. Hoffmann and Cs. D. Tóth, Alternating paths through disjoint line segments, *Inf. Proc. Letts.* **87** (2003), 287–294.

[7] L. Kettner, D. Kirkpatrick, A. Mantler, J. Snoeyink, B. Speckmann, and F. Takeuchi, Tight degree bounds for pseudo-triangulations of points, *Comput. Geom.* **25** (2003), 1–12.

[8] D. Kirkpatrick and B. Speckmann, Kinetic maintenance of context-sensitive hierarchical representations for disjoint simple polygons, in *Proc. 18th SoCG*, 2002, pp. 179–188.

[9] D. T. Lee and A. K. Lin, Generalized Delaunay triangulations for planar graphs, *Discrete Comput. Geom.* **1** (1986), 201–217.

[10] B. Speckmann and Cs. D. Tóth, Allocating vertex π-guards in simple polygons, *14th SODA*, 2003, pp. 109–118.

[11] I. Streinu, A combinatorial approach to planar non-colliding robot arm motion planning, *41st FOCS*, 2000, pp. 443–453.

Approximate Multi-Visibility Map Computation

Narcís Coll[*] Marta Fort[*] J. Antoni Sellarès[*]

Abstract

A multi-visibility map is the subdivision of the domain of a terrain into different regions that, according to different criteria, encode the visibility with respect to a set of view elements. We present an algorithm for computing approximate multi-visibility maps for a terrain, modeled as a TIN, with respect to a set of view segments. Our approach is based on an algorithm that reconstructs an approximation of an unknown planar subdivision from information gathered from linear probes of the subdivision.

1 Introduction

Visibility information of terrain areas is necessary in many Geographic Information Systems applications, such as path planning, mobile phone networks design and environmental modeling.

The visibility map is an structure that encodes the visibility of a terrain with respect to a view element (point, segment, triangle, ...) belonging to or above the terrain. Algorithms for computing the visibility map of a point on a Triangulated Irregular Network (TIN) are available in [3, 4]. Visibility structures for several view elements, that we generically call multi-visibility maps, can be defined by combining the visibility map of such elements according to some operators, for example intersection, union and counting.

When the representation of a terrain is a rough approximation of the underlying terrain, the approximated computation of a multi-visibility map is often sufficient.

In this paper we address the problem of computing approximate multi-visibility maps for a terrain modeled by a TIN with respect to a set of view segments.

2 Preliminaries

2.1 TIN, visibility, multi-visibility map

A terrain can be modeled as the graph of a *Triangulated Irregular Network*, $(\mathcal{T},\mathcal{F})$, formed by a triangulation $\mathcal{T}=\{t_1,\cdots,t_n\}$ of the domain $D\subset\mathbb{R}^2$ and by a family $\mathcal{F}=\{f_1,\cdots,f_n\}$ of linear functions such that: a) function $f_i\in\mathcal{F}$ is defined on triangle t_i, $i=1..n$; b) for every pair of adjacent triangles t_i and t_j, $f_i(x,y)=f_j(x,y)$ for all points $(x,y)\in t_i\cap t_j$.

For any triangle $t_i\in\mathcal{T}$, $\bar{t}_i=f_i(t_i)$ is a triangle in $\mathbb{R}^3$ that we will call a *face* of the TIN, and the restriction of each function f_i to an edge or a vertex of $\mathcal{T}$ is an *edge* or a *vertex* of the TIN, respectively.

Given a terrain $(\mathcal{T},\mathcal{F})$, we will say that a point $Q=(x,y,z)$ is *above* the terrain if the domain point (x,y) belongs to some $t_i\in\mathcal{T}$ and $z>f_i(x,y)$. A *view element* (point, segment, triangle, ...) is an element belonging to or above the terrain. A point P on the terrain is called *visible* from a *view point* V if the interior points of the line segment joining V and P lie above the terrain. A point P belonging to the terrain is called *(weakly) visible* from a *view segment* v if P is visible at least from a point of v.

Given a set V of r view segments, a *multi-visibility map* is a subdivision of the domain D of the terrain into regions according to different visibility criteria. Some subdivision criteria examples are: the region visible from at least one segment and its complement; the region visible simultaneously from all the segments and its complement; the regions visible from exactly $0,1,\cdots,r$ segments.

2.2 Planar subdivision reconstruction

Given an unknown *target* bounded planar subdivision $\mathcal{S}$, in [2] a general purpose online reconstruction algorithm is presented that reconstructs an approximation of the unknown target based on information gathered from linear probes of the target. The goal is to recover the vertices, edges and faces of $\mathcal{S}$ based on the information acquired from the probes. Online means that the algorithm maintains an approximation of the planar subdivision after processing the information from each line probe. Since the set of line probes does not provide sufficient information to reconstruct $\mathcal{S}$ exactly, the algorithm progressively refines the approximation which converges to $\mathcal{S}$ as the number of probe lines increases. How the line probes are chosen plays an important role in how accurate the reconstruction is and how quickly it converges. The probing lines are generated uniformly distributed over a bounding box B of $\mathcal{S}$ so that the probability that a line intersects a piece of the boundary of and edge of $\mathcal{S}$, independent of its location and orientation, is proportional to its length [8]. A line probe L on $\mathcal{S}$ partitions L into a finite set of segments with each segment labelled

[*]Institut d'Informàtica i Aplicacions, Universitat de Girona, Spain, {coll,mfort,sellares}@ima.udg.es. Partially supported by grant TIN2004-08065-C02-02

with the face of $\mathcal{S}$ that contains it. The reconstruction algorithm maintains a triangulation of the endpoints of the segments from which the approximation of the unknown target can be extracted. The mean computational cost of the algorithm when k lines are processed is $O(k \log k)$, where the hidden constant depends on the quotient between the sum of the length of all the edges of $\mathcal{S}$ and the perimeter of B. The algorithm is particularly suited for the setting where computing the intersection of a line with an unknown target is much simpler than computing the unknown target itself.

3 Approximate multi-visibility map computation

We have designed an algorithm for computing approximate multi-visibility maps, according to different criteria, based on the planar subdivision reconstruction algorithm mentioned in previous section.

The input of our algorithm are a TIN $(\mathcal{T}, \mathcal{F})$ and a set of view segments V. The output is an approximate multi-visibility map $\mathcal{M}$. For representing $\mathcal{T}$ and $\mathcal{M}$ we are going to use a DCEL structure. This structure allows for all necessary traversal operations efficiently.

Next we give an overview of the major steps of our algorithm.

- Determine an axis-parallel rectangular bounding box B containing the domain D of the triangulation $\mathcal{T}$. Triangulate the region of B exterior to D to extend $\mathcal{T}$ to a triangulation $\mathcal{T}'$ of B. To this purpose it suffices to add $O(n)$ triangles, where n is the number of triangles in $\mathcal{T}$, with four of them having an edge that coincides with an edge of B. Total cost $O(n)$.

- Generate a set $\mathcal{L}$ of k lines uniformly distributed over B. Each line in $\mathcal{L}$ is obtained in constant time by the following process. Let p be a point selected uniformly at random from the boundary of B, and let ϕ be an angle selected uniformly at random from the interval $[0, \pi)$. A line uniformly distributed over B will be a line l going through p and such that the angle, interior to B, between the edge s of B that contains p and l measures ϕ. Cost $O(k)$.

- For each line $l \in \mathcal{L}$ and each segment $v \in V$, compute $\mathcal{M}_{l,v}$, the restriction on l of the visibility map from v. First we compute the intersection of l with $\mathcal{T}$. Let $a_1, \cdots, a_m$ be the resulting line sections, where $a_j \in t_j$. Cost $O(m)$ (See subsection 3.1). Next we determine the visibility of the segment $\overline{a}_j = f_j(a_j)$ on the face $\overline{t}_j$ of the TIN from the view segment v. Cost $O(n^2 \log n)$ (See subsection 3.2). Finally $\mathcal{M}_{l,j}$ is obtained as the subdivision of l in segments according to the visibility of $\overline{a}_j, j = 1..m$. Cost $O(mn^2)$.

- Denote $\mathcal{M}_l$ the restriction of the multi-visibility map on l with respect to the segments of V. $\mathcal{M}_l$ is obtained by merging the information in $\mathcal{M}_{l,v}$, for each $v \in V$. Naturally, the merging operation depends on the visibility criterion chosen for defining the multi-visibility map. Cost $O(mn^2 \log r)$, where r is the number of segments in V.

- Apply the reconstruction algorithm to the k lines $\mathcal{M}_l, l \in \mathcal{L}$. Cost $O(k \log k)$.

Consequently, the total cost of the algorithm in the worst case is $O(krmn^2 \log n) + O(k \log k)$, where we have supposed that $r \leq n$.

3.1 Line-triangulation intersection

The m line sections resulting from the intersection of l and $\mathcal{T}$ can be computed in $O(m)$ time. To facilitate the task we intersect l with the extended triangulation $\mathcal{T}'$. An edge of the first triangle of $\mathcal{T}'$ intersected by l is the edge of B containing the origin p of l. Starting from this triangle we can traverse $\mathcal{T}'$ following l through adjacent edges in constant time per edge. The whole intersection process takes $O(m)$ time.

From results of integral geometry we know that the mean number of triangles of $\mathcal{T}$ intersected by a line l uniformly distributed over B is $m = \sum_{i=1}^{n} \partial t_i / \partial B$ [8]. Consequently we can considered m as a function depending on the triangulation and the bounding box.

3.2 Segment-segment visibility

Given a view segment v and a segment s on a face of the TIN we want to compute the visible parts of s from v. We will concentrate our attention in the case in which v and s are non coplanar, so that they determine a tetrahedron $\mathbb{T}_{v,s}$. Therefore we will be just interested in the n' faces of the TIN intersecting $\mathbb{T}_{v,s}$. The study of the most simple case in which v and s are coplanar and determine a quadrilateral is omitted in this abstract.

We have adapted to our purpose an algorithm proposed by Bern et al. [1] for computing the visibility with a moving point of view in 3D polygonal scenes and we also use ideas given by S. Ghali in [5] to determine shadow points in 2D polygonal scenes. We will solve the problem by setting a moving point on v, and looking only what happens to s.

Let v be parameterized by "time" t from 0 to 1. We denote v_t the point of v with parameter value t. When points v_0, v_1 have different height we assume that v_0 is located below v_1. As well we consider the segment s parameterized by u from 0 to 1 and we denote s_u the point of s with parameter value u. We choose s_0 and s_1 in such a way that the orthogonal projections onto D of the segments $v_0 s_0$ and $v_1 s_1$ do not intersect.

While we are moving v_t along v, the triangle T_t determined by v_t and s, *the vision triangle*, sweeps the tetrahedron $\mathbb{T}_{v,s}$. During the sweep, for a given time t, the intersection point between an edge e and the vision triangle is the *pierce point*, $p_{e,t}$ (their position and existence depend on t). At time t, the ordered list of edges producing pierce points is the *transparent visibility cycle (tvc$_t$)*. The edges are ordered according to the angular order around v_t of the pierce points determined by them. The ordered list of edges producing visible pierce points from v_t is the *opaque visibility cycle (ovc$_t$)*. While we move v_t, no important changes in the visible parts of s are produced unless in a point where there is a change either in *tvc* or *ovc*. These points are the *critical points* and we say that a *transparent/opaque topology change* occurs, respectively. It is not difficult to see that a transparent topology change occurs at time t if and only if there are two edges e and e' such that there is a line segment that intersects v_t, e, e' and s. And it is an opaque topology change on s if, in addition, the line segment does not pass through the interior of any face.

We are interested in determining the opaque topology changes on s. Using *ovc* and *tvc* we are able to determine all the critical points on v, but we are not able to decide if the corresponding topology change is opaque or transparent. With a preprocess we can update *ovc* and determine where the opaque topology changes occur [1]. To avoid this preprocess we consider the *back face* of the edges producing pierce points. The back face of the edge e at time t, denoted $bf_{e,t}$, is the first face intersected by the ray originated in v_t passing through $p_{e,t}$. The face $bf_{e,t}$ may change when there is a transparent topology change. We will store with each edge e in tvc_t its back face $bf_{e,t}$. When moving v_t, in order to update the back face of an edge in $O(\log n')$ time, we use some "auxiliary" edges and faces. Let E be the set of segments obtained intersecting the TIN edges and $\mathbb{T}_{v,s}$. We take as auxiliary edges the vertical segments joining an endpoint of a segment in E with its orthogonal projection onto D. We take as auxiliary faces the quadrilaterals determined by each segment in E and its orthogonal projection onto D. Observe that for each TIN edge we introduce at most two new auxiliary edges and a face. From now on, faces and edges mean both, the ones from the TIN and the auxiliary ones.

In practice, during the sweep, we have three types of critical points characterized by the following events: (1) an endpoint of an edge is reached by T_t; (2) a non-auxiliary edge intersects the boundary of $\mathbb{T}_{v,s}$ (3) two edges exchange their order in *tvc*, what means that the pierce points they determine exchange their angular position. As $\mathbb{T}_{v,s}$ intersects with $O(n')$ edges, the number of critical points is at most $O(n'^2)$.

Let us focus in the algorithm. The input is the TIN $(\mathcal{T}, \mathcal{F})$, the view segment v and the segment s on a face of the TIN. The output is the set of the endpoints of the visible sub-segments of s ordered by u.

The algorithm has two main parts:

1. *Opaque topology changes on s:* We move v_t along v sweeping the tetrahedron $\mathbb{T}_{v,s}$ with the vision triangle T_t. During the sweep we determine the events corresponding to critical points along v. These points are characterized by 4-tuples (t, e, e', u) such that s_u is the intersection point between s and the line passing through v_t, e and e'. We only store the 4-tuples of critical points that corresponds to opaque topology changes on s, the *shadow points*, in an ordered list $\mathcal{C}_s$.

2. *Visible parts of s:* First we obtain the visible parts of v looking from point $s_0 \in s$. Next we determine the visible parts of s by traversing the set of 4-tuples (t, e, e', u) that characterize the opaque topology changes on s in an ordered way from $u = 0$ to $u = 1$.

3.2.1 Opaque topology changes on s

Events determination. The coordinates t and u that correspond to the two events of type (1) and (2) determined by and edge e can be easily computed in constant time. We will characterize them by a 4-tuple (t, e, e, u).

To obtain the events of type (3) we can consider all the pairs of edges intersecting $\mathbb{T}_{v,s}$ and compute the time were they would exchange their angular position. In spite of using brute force, we only consider the pairs of edges which are adjacent in tvc_t at some time t. For this reason we will obtain them during the sweep process. The coordinates t and u for the events of type (3) are obtained using the skew projection [1].

Events are stored in a priority queue ordered by t. Different events can occur at the same t, in this case the last events to handle are the events of type (2) with e as an auxiliary edge and $u \neq 1$. During all the process we will only store in the priority queue events with t and u between 0 and 1.

Obtaining tvc_0 and ovc_0. Once we have the priority queue initialized with events of type (1) and (2), we can obtain tvc_0 and ovc_0.

We take the set E of edges e corresponding to the events $(0, e, e, u)$ and we delete these events from the priority queue. The transparent visibility cycle tvc_0 contains the edges of E ordered by increasing u. We determine ovc_0 and the back face $bf_{e,0}$ of each edge $e \in E$ by an angular sweep around v_0 in $O(n' \log n')$ time. We store in the list $\mathcal{L}_0$ the faces intersecting the line $v_0 s_1$ by increasing distance from v_0.

Once tvc_0 has been computed we can obtain the events of type (3) by traversing tvc_0. For each pair of adjacent edges in tvc_0 we compute a 4-tuple (t, e, e', u) and we store it in the priority queue.

Handle events. Assume that for a critical point occurred at t' we have $tvc_{t'}$, $ovc_{t'}$, and $\mathcal{L}_{t'}$ (the ordered set of faces intersecting the line $v_{t'}s_0$). To handle the next event $(t, e, e', u), t \geq t'$, found during the sweep we must proceed as follows:

1) Obtain tvc_t and ovc_t. The updating process depends on the type of the event handled, there is one method for events of type (1) or (2) and another for events of type (3). We use the auxiliary edges in order that this update can be done in $O(\log n')$ time. The details are not given in this abstract.

2) Find new events of type (3). We check for new adjacencies in tvc_t and we add the new events in the priority queue.

3) Select opaque topology changes on s. If there has been a change in ovc and the face containing s belongs to $\{bf_{e,t'}, bf_{e',t'}, bf_{e,t}, bf_{e',t}\}$ we store the 4-tuple (t, e, e', u) in the list $\mathcal{C}_s$ of shadow points on s.

Complete the list $\mathcal{C}_s$ of shadow points. Finally we must complete $\mathcal{C}_s$ with the shadow points produced by v_0 and v_1. These points are obtained by traversing ovc_0 and ovc_1. For each edge e such that $s \subset bf_{e,j}, j = 0, 1$ we store in $\mathcal{C}_s$ the 4-tuple $(0, e, v_j^s, u)$ where v_j^s is the vertical segment joining v_j with its orthogonal projection onto D.

3.2.2 Visible parts of s

Visible parts of v from s_0. By working in the triangle defined by v an s_0 we partition v in segments according to the visibility from s_0. Each visible segment of v is determined by a pair of edges (e, e'). The visibility changes in v are obtained by making a sweep around s_0 similar to the sweep made to obtain tvc_0 and ovc_0 but considering v as a face. It takes at most $O(n' \log n')$ time (details omitted in this abstract). At the end of the process the pairs (e, e') are stored in the set of visible parts of v from s_0, $\mathcal{V}_{v,0}$.

Visible parts of s from v. For each 4-tuple in $\mathcal{C}_s$ we update in $O(\log n')$ time $\mathcal{V}_{v,u}$, the set of visible parts of v from s_u, and we obtain the set of endpoints of the the visible segments of s according to a process similar to the one given in [5].

4 Visibility from a polygon and inter-regions visibility

If we are interested in a multi-visibility map from a view polygon placed on or over a terrain we just have to consider the visibility from the segments forming the boundary of this polygon [9] and then merge the visibility information obtained for each single boundary segment to obtain information of the whole view polygon.

A region on a TIN is the subset of faces of the TIN determined by a connected subset of triangles of the domain triangulation. Our algorithm can be easily adapted to determine the approximate weak visibility from region R to region R' [7]. Just consider the region R as a set of view polygons and use the projection onto D of the region R', instead of the whole D, to take random lines.

References

[1] M. Bern, D. Dobkin, D. Eppstein, and R. Grossman. Visibility with a moving point of view, In *Algorithmica* 11, pages 360-378, 1994.

[2] N. Coll, F. Hurtado and J.A. Sellarès. Approximating planar subdivisions ans generalized Vornoi diagrams from random sections, In *19th European Workshop on Computational Geometry*, pages 27-30, 2003.

[3] L. De Floriani, P. Magillo, Visibility Algorithms on Triangulated Digital Terrain Models, In *International Journal of Geographic Information Systems* 8(1), pages 13-41, 1994.

[4] L. De Floriani, E. Puppo, P. Magillo, Applications of Computational Geometry to Geographic Information Systems, In *Handbook of Computational Geometry, J.R. Sack, J. Urrutia (Eds.)*, pages 333-388, Elsevier Science, 1999.

[5] S. Ghali, Computation and Maintenance of Visibility and Shadows in the Plane, In *Sixth Int. Conf. in Central Europe on Computer Graphics and Visualization*, pages 117-124, 1998.

[6] Marc J. van Kreveld, Digital Elevation Models and TIN Algorithms, In *Algorithmic Foundations of Geographic Information Systems, LNCS 1340*, pages 37-78, 1997.

[7] Marc J. van Kreveld, E. Moet, R. van Ostrum, Region inter-visibility in terrains, In *Proc. 20th European Workshop on Computational Geometry*, pages 155-158, 2004

[8] L.A. Santalò, Geometric Probability, In *Society for Industrial and Applied Mathematics*, 1976.

[9] C.An Wang,B.Zhu, Three-dimensional waeak visibility: Complexity and applications, In *Theoretical Computer Scinecce* 234, pages 219-232, 2000.

Approximation Schemes for the Generalized Geometric Problems with Geographic Clustering

Corinne Feremans[*] Alexander Grigoriev[†]

Abstract

This paper is concerned with polynomial time approximations schemes for the generalized geometric problems with geographic clustering. We illustrate the approach on the generalized traveling salesman problem which is also known as Group-TSP or TSP with neighborhoods. We prove that under the condition that all regions are non-intersecting and have comparable sizes and shapes, the problem admits PTAS. To derive a PTAS we extend the algorithm by Arora [2]. This extension involves the dissection mechanism and solution of the selection problem. We observe that the results are applicable to many generalized geometric problems, to other Minkowski norms, and to other fixed dimensional spaces.

1 Introduction

Problem Statement. We consider the following generalization of the classic Euclidean Traveling Salesman Problem (TSP). Assume that a salesman has to visit k customers. Each customer has a set of specified locations in the plane (referred to as a region) where the customer is willing to meet the salesman. The objective is to find a shortest salesman tour that visits each of these customers. If a region is a single point, the described problem becomes the classic TSP.

The described generalization of TSP is known as the Generalized TSP, or the Group-TSP, or the TSP with neighborhoods. For short we shall refer to this problem as to GTSP. In the similar way we can define the generalizations for many other geometric optimization problems, for instance, Minimum Spanning Tree, Minimum Steiner Tree, Minimum k-Connected Subgraph, and many others.

Related Work. Traditionally, TSP attracts attention of many researchers in combinatorial optimization. The problem is known to be NP-hard. In the late nineties it has been shown that TSP and many other geometric optimization problems admit polynomial time approximation schemes (PTAS), see Arora [1] and Mitchell [5].

For GTSP it is known that the problem cannot be efficiently approximated within any constant factor unless $P = NP$, see [6]. Constant factor approximations were developed for the special cases where neighborhoods are disjoint convex fat objects [3], and where the diameter of neighborhoods are comparable [4]. For the case where the neighborhoods are pairwise disjoint unit disks Dumitrescu and Mitchell developed a PTAS [4].

Our results. In this paper we consider the special case of GTSP where the regions are non-intersecting and have comparable sizes and shapes. More precisely, the clustering is defined by a collection of k simple polygons in the plane satisfying the following properties. Two polygons may intersect each other only on the boundaries. Every polygon contain a disk of diameter q and the perimeter of each polygon is bounded from above by cq where c is a constant. In the interiors of these polygons n points are distributed in such a way that each polygon contains at least one of the given points. The points belonging to the same polygon form a region. We associate the polygons with countries, the points with cities, and we refer to the regions as to *geographic clusters*.

We prove that GTSP restricted to the geographic clustering admits a PTAS. This result generalizes the PTAS for pairwise disjoint unit disks by Dumitrescu and Mitchell [4] in two directions. First, we allow a bigger variety of shapes and sizes of the regions. Second, we allow regions to be finite sets and for every region we solve the selection problem, namely, we choose the city to be visited.

Furthermore, our results are based on techniques different from *m-guillotine method* used in [4]. To derive a PTAS for GTSP with geographic clustering we extend the *randomized dissection method* presented in Arora [2]. We introduce a new dissection technique which can be used to construct PTAS for many generalized geometric problems with geographic clustering. In contrast to [2] where the level lines in the dissection are straight, the new dissection is based on the curved (piece-wise linear) level lines. We also adopt the dynamic programming routine from [2] to solve the selection problems in the regions.

Finally, we observe that our algorithm is applicable to generalized versions of many geometric optimization problems listed in [2]. We also observe that the methodology is applicable to other Minkowski norms and to other fixed dimensional spaces which also extends the results from [4].

[*]Department of Quantitative Economics, University of Maastricht, c.feremans@ke.unimaas.nl

[†]Department of Quantitative Economics, University of Maastricht, a.grigoriev@ke.unimaas.nl

2 PTAS for GTSP with geographic clustering

To obtain a PTAS for GTSP with geographic clustering we follow the Arora's PTAS [2] making the following changes:

1. **Curved dissection.** Straight level lines in the dissection could cut a polygon into parts. This causes the problem that for a cut polygon we cannot decide to which node of the dissection tree it belongs. We fix this problem by *curved dissection.* Assume that a horizontal line in the dissection cuts some polygon into two or more parts, say "north" and "south". In this case, we redefine the level line as follows. Let the level line follow the boundary of the "south" leaving the entire polygon in the "north", see Figure 1. Similarly, we change the vertical level lines leaving the cut polygons in the "west". Doing the curved dissection we uniquely define the places of cut polygons in the 4-ary dissection tree.

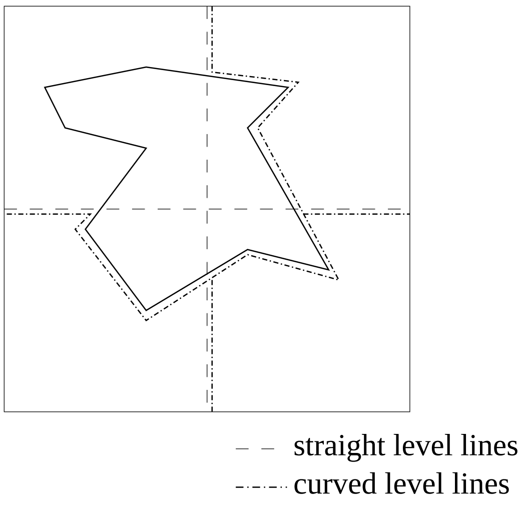


Figure 1: Curved dissection

2. **Early stop.** If the curved dissection goes too deep, two sequential curved level lines may have a common segment which again can cause the problem when deciding to which node of the dissection tree the polygon belongs. To avoid common segments we stop the dissection when the distance between two sequential straight level lines is cq. So, with the early stop and the curved dissection we uniquely define the places for all polygons in the 4-ary dissection tree.

3. **Reduced random shifts.** By early stop in the randomized dissection it is sufficient to consider the random horizontal shifts a and random vertical shifts b such that both, a and b, are divisible by cq.

4. **Portals multiplicities.** Since we allow curved level lines, the perimeter of the "squares" in the dissection increases. To keep the inter-portal distances unchanged on the curved level lines, we have to enlarge the number of portals on the perimeter of the "squares". From the condition that perimeter of any polygon is at most cq, we derive that the perimeter of any "square" in the curved dissection is at most $4c^2/\pi$ times bigger than the perimeter of the corresponding square in the straight dissection. Therefore, it is sufficient to multiply the portal parameter by the constant factor $4c^2/\pi$.

5. **Solution of the selection problem.** From early stop and condition that each polygon contains a disk of diameter q, we derive that in the leaves of the revised 4-ary dissection tree the number of polygons is at most $4c^2/\pi$. Since the number of cities in each polygon is finite and we have to pick up only one city per polygon, for any leaf of the dissection tree we can effectively enumerate all possible arrangements for the cities to be visited.

For the revised algorithm we prove the *Structure Theorem* as in [2], and consequently we have the following theorem.

Theorem 1 *With probability at least 1/2 over the choice of the shifts a and b, the revised algorithm provides a portal respecting tour of length* $(1+\varepsilon)OPT$ *in time* $O(n^{O(c^2/\varepsilon)})$*, where* $\varepsilon > 0$ *is a required accuracy parameter, and* OPT *is the optimum GTSP tour length.*

To derandomize the algorithm we can, for instance, go through all choices for a and b.

References

[1] S. Arora. Nearly linear time approximation schemes for Euclidean TSP and other geometric problems. JACM, 45, pages, 1–30, 1998.

[2] S. Arora. Approximation schemes for NP-hard geometric optimization problems: A survey. Mathematical Programming, 97, pages 43–69, 2003.

[3] M. de Berg, J. Gudmundsson, M. Katz, C. Levcopoulos, M. Overmars, A. van der Stappen. TSP with neighborhoods of varying size. In Proc. ESA, pages 187–199, 2002.

[4] A. Dumitrescu, J.S.B. Mitchell. Approximation algorithms for TSP with neighborhoods in the plane. J. of Algorithms, 48, pages 135 – 159, 2003.

[5] J.S.B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems. SIAM J. Comput., 28, pages 1298 – 1309, 1999.

[6] S. Safra, O. Schwartz. On the complexity of approximating TSP with neighborhoods and related problems. In Proc. ESA, pages 446–458, 2003.

Approximation Algorithm for the L_1-Fitting Circle Problem

Sariel Har-Peled[*]

Abstract

In this paper, we study the problem of L_1-fitting a circle to a set of points in the plane, where the target function is the sum of distances of the points to the circle. We show an $(1 + \varepsilon)$-approximation algorithm, with running time $O(n + \text{poly}(\log n, 1/\varepsilon))$, where $\text{poly}(\log n, 1/\varepsilon)$ is a constant degree polynomial in $\log n$ and $1/\varepsilon$. This is the first subquadratic algorithm for this problem.

1 Introduction

Motivated by a variety of applications, considerable work has been done on measuring various descriptors of the extent of a set P of n points in $\Re^d$. We refer to such measures as *extent measures* of P. Roughly speaking, an extent measure of P either computes certain statistics of P itself or it computes certain statistics of a (possibly nonconvex) geometric shape (e.g. sphere, box, cylinder, etc.) enclosing P. Examples of the former include computing the kth largest distance between pairs of points in P, and the examples of the latter include computing the smallest radius of a sphere (or cylinder), the minimum volume (or surface area) of a box, and the smallest width of a slab (or a spherical or cylindrical shell) that contain P.

Shape fitting, a fundamental problem in computational geometry, computer vision, machine learning, data mining, and many other areas, is closely related to computing extent measures. A widely used shape-fitting problem asks for finding a shape that best fits P under some "fitting" criterion. A typical criterion for measuring how well a shape γ fits P, denoted as $\mu(P, \gamma)$, is the maximum distance between a point of P and its nearest point on γ, i.e., $\mu(P, \gamma) = \max_{p \in P} \min_{q \in \gamma} d(p, q)$. This is the L_∞-*fitting problem*[1]. Here, one can define the extent measure of P to be $\mu(P) = \min_\gamma \mu(P, \gamma)$, where the minimum is taken over a family of shapes (such as points, lines, hyperplanes, spheres, etc.). For example, the problem of finding the minimum radius sphere (resp. cylinder) enclosing P is the same as finding the point (resp. line) that fits P best, and the problem of finding the smallest width slab (resp. spherical shell, cylindrical shell) is the same as finding the hyperplane (resp. sphere, cylinder) that fits P best.

The exact algorithm for computing extent measures are generally expensive, e.g., the best known algorithms for computing the smallest volume bounding box containing P in $\Re^3$ require $O(n^3)$ time. Consequently, attention has shifted to developing approximation algorithms [BH01, ZS02]. A general approximation technique was recently developed for such problems by Agarwal *et al.* [AHV04]. This implies among other things that one can $(1+\varepsilon)$-approximate the circle best L_∞-fit a set of points in the plane in $O(n + 1/\varepsilon^{O(1)})$ time (see [AAHS00] and [Cha02] and references therein for more information about this problem).

The main problem of the L_∞ measure in shape fitting is that it is very sensitive to outliers. Namely, a single outlying point can change the price of the optimal solution dramatically. There are two natural solutions. The first approach, is to change the target function to be less sensitive to outliers. For example, instead of considering the maximum distance, one can consider the sum of distances (i.e., L_1-fitting), or the sum of squared distance (i.e., L_2-fitting). The L_2-fitting in the case of a single linear subspace is well understood, and is no more than SVD (singular value decomposition). Fast approximation algorithms are known for this problem, see [FKV98, RVW04] and references therein. As for the L_1-fitting, this problem can be solved using linear programming techniques, in polynomial time in high dimensions, and linear time in constant dimension [YKII88]. Recently, Clarkson gave a faster algorithm for this problem [Cla05] which works via sampling.

The problem seems to be harder once the shape considered is not a linear subspace. There is considerable work on nonlinear regressions (i.e., extension of the least squares technique) for various shapes, but there not seems to be a guaranteed approximation algorithms for the circle case [SW89]. The hardness in the L_1-fitting of a circle seems to rise from the target function is a sum of terms, each term being an absolute value of a difference of a square root of a polynomial and a radius. It seems doubtful that analytical solution would exist for such a target function, as it is related to the Fermat-Weber problem [Wes93].

[*]Department of Computer Science; University of Illinois; 201 N. Goodwin Avenue; Urbana, IL, 61801, USA; sariel@uiuc.edu; http://www.uiuc.edu/~sariel/. Work on this paper was partially supported by a NSF CAREER award CCR-0132901.

[1]The L_∞-fitting comes from considering the distance of every point to the shape being a coordinate in a vector, and what metric we apply to this vector.

The second approach, is to specify the number k of outliers in advance, and find the best shape L_∞-fitting all but k of the input points. Har-Peled and Wang showed that there is a coreset for this problem [HW04], and as such it can be $(1+\varepsilon)$-approximated in $O(n + \text{poly}(k, \log n, \varepsilon))$ time, for a large family of shapes. The work of Har-Peled and Wang was mainly motivated by trying to solve the problem of L_1-fitting a circle to a set of points.

In this paper, we describe an $(1+\varepsilon)$-approximation algorithm for the L_1-fitting of a circle to a set of points in the plane. The solution has running time of $O(n + \text{poly}(\log n, 1/\varepsilon))$, and the result is correct with high probability. The only previous algorithm for this problem we are aware of, is due to Har-Peled and Koltun [HK04a], and it works in $O(n^2\varepsilon^{-2}\log^2 n)$ time.

Due to extreme space limitations, we only provide a sketch of the paper. A full version of this paper is available from the author's webpage `http://www.uiuc.edu/~sariel/papers/05/l1_fitting/`.

2 Approximate L_1 Fitting of a Circle to a Set of Points

2.1 Solution Outline.

The problem of best L_1-fitting a circle to a set of points in the plane, is equivalent to finding the points in 3D the minimizes the sum of distances to a set of cones.

As such, our solution is based on two steps. In the first step, we compute a small set of levels, and assign weights to them, such that solving the problem on those (weighted) levels would be equivalent to solving the problem on the original arrangement of cones. Unfortunately, this is by itself insufficient, as those levels by themselves might be of high complexity, and computing them would be prohibitly expensive. To this end, we replace the exact level by approximate level, using random sampling. This ensures that each such level is a shallow level (in the appropriate random sample).

This still fall short of solving the problem, because even a shallow level might have high complexity. As such, we simplify those levels in such a way that preserves the sum of vertical distances. The last step is done by applying a recent result of Har-Peled and Wang [HW04] that shows that one can do such a simplification, and it is of small size.

In the end of this process, we have a weighted arrangement of surface patches of small size (say, of complexity $O(\text{poly}(1/\varepsilon, \log n))$), such that we need to solve the problem in this arrangement. We can now solve the problem by using an reasonably efficient brute force approach. Indeed, we decompose the arrangement into vertical slabs where a vertical line intersect the surfaces in the same order. We need to solve the problem inside this prism. In the original settings, this corresponds to a (small) weighted set of points, where we want to best fit them to a circle. We use a slow (cubic time) approximation algorithm to do that.

2.2 Warmup Exercise – Slow Approximation

Lemma 1 *Let $\mathcal{G}$ be a set of n weighted cones in $\Re^3$ with total weight W, and $\varepsilon > 0$ a parameter. One can find a point $p \in \Re^3$ such that, $\nu_{\mathcal{G}}(p) \le (1+\varepsilon)\nu_{\text{opt}}(\mathcal{G})$, where $\nu_{\text{opt}}(\mathcal{G})$ is the price of the global minimum. The running time is $O(n^3\text{poly}(\log W, \varepsilon^{-1}))$.*

The same running time holds, if the feasible region is restricted to a region of constant complexity of space.

2.3 Second Warmup Exercise – the One Dimensional Case

In this section, we consider the one dimensional problem of approximating the distance function of a point x to a set of points $Z = \langle z_1, z_2, \ldots, z_n\rangle$, where $z_1 \le z_2 \le \ldots \le z_n$. Formally, we want to approximate the function $\nu_Z(z) = \sum_{z_i \in Z} |z_i - z|$. This is no more than the one median function for those points on the line. This corresponds to a vertical line in three dimensions, where each z_i represents the intersection of the vertical line with the surface γ_i. The one dimensional problem is well understood, since it is no more than the 1-median problem, and there exists a coreset for it, see [HM04, HK04b]. Unfortunately, it is unclear how to perform the operations that corresponds to those coreset construction in a global fashion, so that the construction would hold for all vertical lines.

Our first step, is to do *chunking*. Formally, we partition Z symmetrically into subsets, such that the size of the subsets increase in size as one comes toward the middle of the set. Formally, the set $L_i = \{z_i\}$ contains the ith point on the line, for $i = 1, \ldots, M$, where $M \ge 10/\varepsilon$ is a parameter to be determined shortly. Similarly, $R_i = \{z_{n-1+1}\}$, for $i = 1, \ldots, M$. Next, let $\alpha_M = M$, and let $\alpha_{i+1} = \min(\lceil(1+\varepsilon/10)\alpha_i\rceil, n/2)$, for $i = M, \ldots, N$, where α_N is the first number in this sequence equal to $n/2$. Now, let $L_i = \{z_{\alpha_{i-1}+1}, \ldots, z_{\alpha_i}\}$ and $R_i = \{z_{n-\alpha_{i-1}}, \ldots, z_{n-\alpha_i+1}\}$, for $i = M+1, \ldots, N$. Consider the partition of Z formed by $L_1, L_2, \ldots, L_N, R_N, \ldots, R_2, R_1$. Clearly, this is a partition of Z into "exponential sets". The margin M sets on the boundary are singletons, and all the other sets grow exponentially in cardinality, till the cover the whole set Z.

We next pick a point an arbitrary point $l_i \in L_i$ and $r_i \in R_i$, and we also assign weight $|R_i| = |L_i|$ to each such point. for $i = 1, \ldots, N$. Let $\mathcal{S}$ be the resulting weighted set of points. We claim that this is a coreset for the 1-median function.

Lemma 2 *Let A be a set of n real numbers, and let a be any number of A. We have that $|\nu_A(z) - |A|\|az\|| \leq \nu_A(a)$.*

Lemma 3 *$\nu_Z(z) \approx_{\varepsilon/5} \nu_{\mathcal{S}}(z)$, for any $z \in \Re$.*

Next, we "slightly" perturb the points of the coreset $\mathcal{S}$. Formally, assume that we have points $l'_1, \ldots, l'_N, r'_1, \ldots, r'_N$ such that $\|l'_i l_i\|, \|r'_i r_i\| \leq (\varepsilon/20)\|l_i r_i\|$, for $i = 1, \ldots, N$. Let $\mathcal{S}' = \{l'_1, \ldots, l'_N, r'_N, \ldots, r'_1\}$ be the resulting weighted set. We claim that $\mathcal{S}'$ is still a good coreset.

Lemma 4 *$\nu_Z(z) \approx_{\varepsilon/3} \nu_{\mathcal{S}'}(z)$, for any $z \in \Re$.*

2.4 Additional Tools

2.4.1 Approximating a Level by a Shallow Level in a Random Sample

Lemma 5 *Let $\mathcal{G}$ be a set of n surfaces in $\Re^d$, $\varepsilon > 0$, and let k be a number between 0 and $n/2$. Let $\rho = \min\left(ck^{-1}\varepsilon^{-2}\log n, 1\right)$, and pick each surface of $\mathcal{G}$ into a random sample $\mathcal{R}$ with probability ρ. Then, with high probability, the $L = k\rho = O(\varepsilon^{-2}\log n)$ level of $\mathcal{A}(\mathcal{R})$ lies between the $(1-\varepsilon)k$-level to the $(1+\varepsilon)k$-level of $\mathcal{A}(\mathcal{G})$. This holds with high probability.*

2.4.2 Approximating the Extent of Shallow Levels

We need the result of Har-Peled and Wang [HW04]. It states that for well behaved set of functions, one can find a small subset of the functions such that the vertical extent of the subset approximates the extents of the whole set. This holds only for "shallow" levels $\leq k$. In our application k is going to be about $O(\varepsilon^{-2}\log n)$.

3 The approximation algorithm

We are now ready to put everything together. Let the input be a set P of n points in the plane. We define, as in Section 2.1, for each point of P a surface in $\Re^3$. Each such surface is a cone. Let $\mathcal{G}$ be the resulting set of cones.

We decompose the arrangement $\mathcal{A}(\mathcal{G})$ into levels. Next, we chunk the levels into sets, as done in Section 2.3, where $M = O(\varepsilon^{-2}\log n)$ and $N = M + O(\varepsilon^{-1}1\log n) = O(\varepsilon^{-2}\log n)$. Let $L_1, \ldots, L_N, R_N, \ldots, R_1$ denote the resulting chunks of levels. Next, let $l_i = \mathbf{L}_{\mathcal{G},i-1}$ and $r_i = \mathbf{U}_{\mathcal{G},i-1}$ for $i = 1, \ldots, M$. For $i = N+1, \ldots, M$, we generate a random sample $\mathcal{R}_i$ of $\mathcal{G}$, according to Lemma 5, and levels $l_i = \mathbf{L}_{\mathcal{R}_i,k_i}$ and $r_i = \mathbf{U}_{\mathcal{R}_i,k_i}$, where $k_i = O(\varepsilon^{-2}\log n)$. We are guaranteed, with high probability, that l_i lies between the lowest and highest levels defined by L_i, and similarly that r_i lies between the highest and lowest levels of R_i, for $i = M+1, \ldots, N$.

Of course, computing the surfaces l_i and r_i explicitly is going to be prohibitively expensive. However, l_i and r_i are the bottom/top k_i-level in the arrangement $\mathcal{A}(\mathcal{R}_i)$. Since, k_i is relatively small, this means that those levels are shallow. As such, we can compute a subset $V_i \subseteq \mathcal{R}_i$, such that $V_i|_{k_i}^{k_i}(x) \geq (1-\varepsilon/10)\mathcal{R}_i|_{k_i}^{k_i}(x)$, for all $x \in \Re^2$, using [HW04]. Here, $s = 2$ and $|V_i| = O(k_i/\varepsilon^4) = O(\mathrm{poly}(1/\varepsilon, \log n))$.

Next, let l'_i and r'_i the k_i-bottom/top levels in $\mathcal{A}(V_i)$, respectively, for $i = M+1, \ldots, N$. Note that for $i = 1, \ldots, N$ we can use the same sample. As such, $V_1 = \cdots = V_M$. We assign the surfaces l'_i and r'_i the weight $|L_i| = |R_i|$, for $i = 1, \ldots, N$. Next, consider the set of the weighted surfaces $\mathcal{G}' = \{l'_1, \ldots, l'_N, r'_1, \ldots, r'_N\}$. It follows by Lemma 4 that

$$\nu_{\mathcal{G}'}(p) \approx_\varepsilon \nu_{\mathcal{G}}(p),$$

where p is any point in $\Re^3$. This holds with high probability. We still remain with the question of finding the global minimum of $\nu_{\mathcal{G}'}(p)$. To this end, we decompose $\Re^3$ into vertical prisms, such that inside such prism a vertical line intersect exactly the same surface patches in the same order. It is now straightforward to show that the number of such prisms is $O(\mathrm{poly}(\log n, 1/\varepsilon))$. Inside each such prism, the arrangement $\mathcal{A}(\mathcal{G}')$ has the same surfaces intersecting it in the same ordering. Namely, we have a portion of the parametric space we have to find the minimum for (i.e., the prism), while having a small number of weighted surfaces we have to consider.

Using the slow algorithm inside every prism, gives us an $(1+\varepsilon)$-approximation inside each such prism. Since the running time inside each such prism is $O(\mathrm{poly}(\log n, 1/\varepsilon))$, there are $O(\mathrm{poly}(\log n, 1/\varepsilon))$ prisms. Thus the overall running time is $O(n + \mathrm{poly}(\log n, 1/\varepsilon))$. We summarize:

Theorem 6 *Given a set P of n points in the plane, and parameter ε, one can compute in $O(n + \mathrm{poly}(\log n, 1/\varepsilon))$ time the circle minimizing the L_1 fitting price to P. The running time is $O(n + \mathrm{poly}(\log n, 1/\varepsilon))$. The result is correct with high probability.*

4 Conclusions

We had described in this paper an $(1+\varepsilon)$-approximation algorithm for the problem of L_1-fitting of a circle to a set of points in the plane. The running time of the new algorithm is $O(n + \mathrm{poly}(\log n, 1/\varepsilon))$, which is a linear running time for fixed ε. The constant powers hiding in the polylogarithmic term are too embarrassing to be explicitly stated, but are probably somewhere between 20 to 60. As such, this algorithm is only of theoretical interest. As such, the first open problem raised by this work is to improve this constants. A considerably more interesting problem is to develop a practical algorithm for this problem.

It is the author's belief that the techniques described in this paper, can be also applied to the problem of L_2-fitting of a circle to a set of points (i.e., best circle fitting a set of points minimizing the sum of square distances of the points to the circle). More importantly, it seems that the technique should be applicable to any of the fitting problems handled by the algorithm of Agarwal *et al.* [AHV04]. This includes the L_1-fitting of a sphere or a cylinder to a set of points.

A natural question is whether one can use the techniques of Har-Peled and Wang directly, to compute a coreset for this problem, and solve the problem on the coreset directly (our solution did a similar thing, by breaking the parametric space into a small number regions, and constructing a coreset inside each such region). There is unfortunately a nasty technicality that requires that a coreset for the L_1-fitting of linear function, is also coreset if we take the square root of the functions. It seems doubtful that this claim holds in general, but maybe a more careful construction of a coreset for the linear functions case would still work. The author leaves this as an open problem for further research.

Acknowledgments

The author would like to thank Pankaj Agarwal, Arash Farzan, Vladlen Koltun, Bardia Sadri, Kasturi Varadarajan, and Yusu Wang for useful discussions related to the problems studied in this paper.

References

[AAHS00] P. K. Agarwal, B. Aronov, S. Har-Peled, and M. Sharir. Approximation and exact algorithms for minimum-width annuli and shells. *Discrete Comput. Geom.*, 24(4):687–705, 2000.

[AHV04] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Approximating extent measures of points. *J. Assoc. Comput. Mach.*, 51(4):606–635, 2004.

[BH01] G. Barequet and S. Har-Peled. Efficiently approximating the minimum-volume bounding box of a point set in three dimensions. *J. Algorithms*, 38:91–109, 2001.

[Cha02] T. M. Chan. Approximating the diameter, width, smallest enclosing cylinder and minimum-width annulus. *Internat. J. Comput. Geom. Appl.*, 12(2):67–85, 2002.

[Cla05] K. L. Clarkson. Subgradient and sampling algorithms for l1 regression. In *Proc. 16th ACM-SIAM Sympos. Discrete Algorithms*, 2005. to appear.

[FKV98] A. Frieze, R. Kannan, and S. Vempala. Fast monte-carlo algorithms for finding low-rank approximations. In *FOCS '98: Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, page 370. IEEE Computer Society, 1998.

[HK04a] S. Har-Peled and V. Koltun. Approximate l_1 and l_2 circle fitting in (easy) polynomial time. manuscript, 2004.

[HK04b] S. Har-Peled and A. Kushal. Smaller coresets for k-median and k-means clustering. http://www.uiuc.edu/~sariel/papers/04/small_coreset/, 2004.

[HM04] S. Har-Peled and S. Mazumdar. Coresets for k-means and k-median clustering and their applications. In *Proc. 36th Annu. ACM Sympos. Theory Comput.*, pages 291–300, 2004.

[HW04] S. Har-Peled and Y. Wang. Shape fitting with outliers. *SIAM J. Comput.*, 33(2):269–285, 2004.

[RVW04] L. Rademacher, S. Vempala, and G. Wang. Matrix approximation and projective clustering via adaptive sampling. manuscript, 2004.

[SW89] G.A.F. Seber and C.J. Wild. *Nonlinear regression*. Jonh Wiley & Sons, 1989.

[Wes93] G. Wesolowsky. The Weber problem: History and perspective. *Location Science*, 1:5–23, 1993.

[YKII88] P. Yamamoto, K. Kato, K. Imai, and H. Imai. Algorithms for vertical and orthogonal l1 linear approximation of points. In *Proc. 4th Annu. ACM Sympos. Comput. Geom.*, pages 352–361. ACM Press, 1988.

[ZS02] Y. Zhou and S. Suri. Algorithms for a minimum volume enclosing simplex in three dimensions. *SIAM J. Comput.*, 31(5):1339–1357, 2002.

Competitive Online Searching for a Ray in the Plane

Andrea Eubeler* Rudolf Fleischer† Tom Kamphans* Rolf Klein* Elmar Langetepe*
Gerhard Trippen‡

Abstract

We consider the problem of a searcher that looks for a lost flashlight in a dusty environment. The search agent finds the flashlight as soon as it crosses the ray emanating from the flashlight, and in order to pick it up, the searcher has to move to the origin of the light beam.

First, we give a search strategy for a special case of the ray search—the window shopper problem—, where the ray we are looking for is perpendicular to a known ray. Our strategy achieves a competitive factor of ≈1.059, which is optimal. Then, we consider the search for a ray with an arbitrary position in the plane. We present an online strategy that achieves a factor of ≈22.513, and give a lower bound of ≈16.079.

Keywords: Online motion planning, competitive ratio, searching, ray search

1 Introduction

Searching in an unknown environment is a basic task in robot motion planning and well-studied in many settings. For example, Gal and independently Baeza-Yates et. al. [7, 2] considered the task of finding a point on an infinite line using a searcher, that starts in the origin and neither knows the distance nor the direction towards the goal. They introduced the so called *doubling strategy* that is, the agent moves alternately to the left and to the right, doubling it's exploration depth in every iteration step. Searching on the line was generalized to searching on m concurrent rays starting from the searcher's origin, see [7, 2]. Many other variants were discussed since then, for example m-ray searching with restricted distance (Hipke et. al. [9], Langetepe [13], Schuierer [14]), m-ray searching with additional turn costs (Demaine et. al. [4]), parallel m-ray searching (Hammar et. al. [8]) or randomized searching (Kao et. al. [11]).

The quality of a strategy that deals with incomplete information —an *online strategy*—is usually measured by the cost of the online solution compared to the optimal solution. More precisely, let $|S|$ denote the cost of an online strategy, S, and $|S_{\text{Opt}}|$ the cost of the optimal solution, then we call S *competitive with factor* C, iff there exists a constant A such that $|S| \leq C \cdot |S_{\text{Opt}}| + A$ holds for every input to S. In our case, the costs incurred by a search strategy is given by the length of the path covered by the searcher, and the optimal solution is the length of the shortest path from the searcher's origin to the goal. The competitive framework was introduced by Sleator and Tarjan [17] and used for many settings, see e. g. the survey by Fiat and Woeginger [5]. For a general overview of online motion planning problems and its analysis see the surveys [3, 15, 16, 10]. Another measure is the *search ratio*, see Koutsoupias et. al. [12] and Fleischer et. al. [6]

In this paper, we consider the search for the origin of a ray in the plane. The searcher has no vision, but recognizes the ray and it's origin as soon as it enters it. First, we consider a simplified version of this problem: the origin of the ray, r, we are looking for is located on another ray, r', perpendicular to r. The searcher's start point and r are located on the same side of r'. Moreover, r' is known. We call this problem the *window shopper problem*, since we can imagine r' as a line of shopping windows. A buyer walks along these windows, looking e. g. for a present, and walks towards the window as soon as an appropriate item is spoted. We give a search strategy for this problem, that achieves an optimal competitive factor of 1.059 . . . Then we consider the general case, and give a search strategy that achieves a factor of 22.513 . . . and give a lower bound of 16.079 . . .

2 The window shopper problem

First, we consider the problem of finding a gift along a line of shopping windows. W. l. o. g. we assume, that the line of sight, i. e. the ray r we are looking for, is parallel to the X-axis, starting in $(1, y_r)$ for $y_r \geq 0$, and emanating to the left hand side of r'. The searcher starts in the origin $(0, 0)$, see Figure 1. The

*University of Bonn, Institute of Computer Science I, D-53117 Bonn, Germany.
†Fudan University, 220 Handan Road, Shanghai 200433
‡Hong Kong University of Science and Technology Clear Water Bay, Kowloon Hong Kong.

The work described in this paper was partially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. HKUST6010/01E) and by a grant from the Germany/Hong Kong Joint Research Scheme sponsored by the Research Grants Council of Hong Kong and the German Academic Exchange Service (Project No. G-HK024/02).

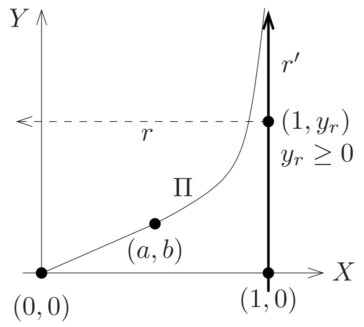


Figure 1: A strategy for the window shopper problem.

goal is discovered as soon as we reach its height, i. e. its Y-coordinate y_r. In order to reach the goal we finally have to move towards it (if we haven't reached it by coincidence).

Theorem 1 *There exists a strategy with an optimal competitive factor of* 1.059... *for searching the origin of a ray, r, that emanates from a known ray r' perpendicular to r.*

Proof. Apparently a good search path moves simultaneously along and towards the wall, i. e. in positive X- and Y-direction. It is obvious that the competitive factor converges to 1 for goals with very small Y-coordinate and also for goals with a large Y-coordinate. Using this properties, we construct the following search strategy, Π: First, we follow a line segment. In the second part we follow a curve $f(x)$ that converges to the wall and maintains the value of the competitive factor given by the line segment at the beginning of the search path, see Figure 1.

By specifying the first part of the search path as the line segment from the origin to a point (a, b) we can describe the competitive factor as a monotone increasing function $\phi(t)$ with

$$\phi(t) = \frac{t\sqrt{a^2+b^2}+1-ta}{\sqrt{1+t^2b^2}},$$

and $\phi'(t) \geq 0 \;\; \forall t \in [0, 1]$. Hence, $b \leq \sqrt{1-2a}$ follows.

From now on we assume $b = \sqrt{1-2a}$ and therefore $a = \frac{1-b^2}{2}$.

Now we are looking for a curve $f(x)$ that maintains the value

$$\delta = \frac{\sqrt{a^2+b^2}+1-a}{\sqrt{1+b^2}} = \sqrt{1+b^2}$$

for the competitive factor.

This means that the length of the whole search path, i. e. the line segment(s) and the curve, is δ times the Euclidean distance from the origin to the goal.

Solving the resulting differential equation of the form

$$\sqrt{a^2+b^2}+1-x+\int_a^x \sqrt{1+f'(t)^2}\,dt = \delta\cdot\sqrt{1+f(x)^2}$$

yields the values $b = 0.349\ldots$ and $\delta = 1.059\ldots$

To show the optimality of the given strategy, we observe an arbitrary curve $g(x)$ starting at the origin. Then we have to consider two cases.

Case 1: g reaches height b left to a.
If the goal is at height b then g has a competitive factor which is worse than our competitive factor. This also holds for goals above b, because the curve f has already reached its maximum value at (a, b).

Case 2: g reaches height b right to a.
Then the curve g intersects the curve f at some intersection point A; at the latest when the curve f reaches the wall. In this case the length of the path on g is longer than the path on f (since f is monotone increasing and convex), so the competitive factor of g is again worse than the factor of f in A. □

3 Searching for a ray

Now, we suppose that we are positioned somewhere in the plane and we want to find an arbitrary ray. It seems to be a good strategy to search for the ray by moving on a logarithmic spiral. Since we have no sight the ray is only found when we cross it. Our aim is to reach the origin of the ray—simply by following the ray after we have found it.

We are interested in the worst case. This means that we want to construct a position of the ray that maximizes the competitive factor. Since we want to search the ray by moving on a spiral, we are looking for a spiral that minimizes the worst case in the plane.

The spiral is given by equation

$$f(\theta) = ae^{b\theta}, \qquad -\infty < \theta < \infty.$$

Due to the properties of the spiral, we only have to consider positions of the ray in one turn (2π).

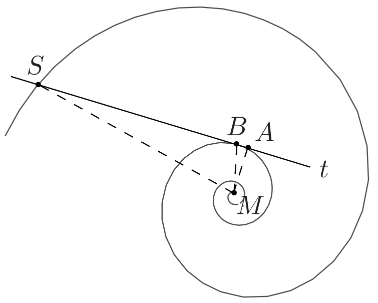


Figure 2: The tangent t to the spiral in point B.

One can easily see that the competitive factor reaches a local maximum if the ray is a tangent t to the spiral: Let B now be the tangent point of t, see Figure 2. The ray is missed in point B and detected in point S. We examine different positions of the ray's origin on the tangent. The starting point can not be on the left side of B, because otherwise the ray would no longer be a tangent. The competitive factor reaches a higher value for A than for B, see Figure 2, because A is farther away from S than B and $|\overline{MA}|$ is shorter than $|\overline{MB}|$.

First, we will find the best spiral for a point A, so that $|\overline{MA}|$ is perpendicular to tangent t.

Lemma 2 *The competitive factor δ_A for the point A, so that $|\overline{MA}|$ is perpendicular to the tangent t, only depends on the spiral parameter b and is given by*

$$\delta_A(b) = \frac{e^{b(2\pi+\beta(b))}}{\sin\alpha \cdot \cos\alpha} + \frac{e^{b(2\pi+\beta(b))} \cdot \sin(\beta(b))}{\sin^2\alpha} + b\,.$$

Its minimum value is $22.4908\ldots$ for $b = 0.1137\ldots$, where α is the tangent angle and β the angle $\angle BMS$.

Now we will show, that this b also gives us the optimal spiral for all tangents and all origins. This can be shown as follows. The adversary can move the point A to A' which gives a factor of $\delta_{A'}(b,\gamma) = \cos\gamma \cdot \delta_A(b) + \sin\gamma$ by simple geometry (see Figure 3).

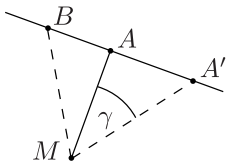


Figure 3: The triangle MAA'.

Therefore in general we have to minimize $\delta_A(b)$ whereas the adversary can choose the worst case γ. By simple analysis we have:

Theorem 3 *The best worst case competitive factor is $22.5130\ldots$ This value is reached by point A' which is specified by $\gamma = 0.4443\ldots$ and $b = 0.1137\ldots$*

Due to space limitations we omit the proofs of Lemma 2 and Theorem 3. See the full version of the paper.

4 A lower bound for searching a ray

We discuss a subproblem and consider a subset of rays, such that the extension of every ray goes through the starting point s.

If we consider the full bundle of lines passing through s, the given problem is equivalent to the problem of searching for a point in the plane as presented by Alpern and Gal [1]. We assume that the goal is detected, if it is swept by the radius vector of the trajectory. Alpern and Gal [1] showed that among all monotone and periodic strategies, a logarithmic spiral represented by polar coordinates $(\gamma, e^{b\gamma})$ gives the best search strategy in this setting. A strategy S represented by its radius vector $X(\gamma)$ is called periodic and monotone, if γ is always increasing and X also satisfies $X(\gamma + 2\pi) \geq X(\gamma)$.

The factor of the best monotone and periodic strategy is given by $\min_b e^{2\pi} b\sqrt{1+\frac{1}{b^2}} = 17.289\ldots$ and achieves its minimum for $b = 0.15540\ldots$, see Alpern and Gal [1]. Note, that the task does not include that the origin of the ray has to be visited.

Unfortunately, it was not shown that a periodic and monotone strategy is the best strategy for this problem. Alpern and Gal state, that it *seems to be a complicated task* to show that the spiral optimizes the competitive factor. Thus, the given factor cannot be adapted to be a lower bound to our problem.

Therefore we consider a discrete bundle of n rays that emanate from the the start and which are seperated by an angle $\alpha = \frac{2\pi}{n}$, see Figure 4. We are searching for a goal on one of the n rays. Again the goal is detected if it is swept by the radius vector of the trajectory. Note, that if n goes to infinity we are back to the original problem. But we can neither assume that we have to visit the rays in a periodic order nor that the depth of the visit increases in every step.

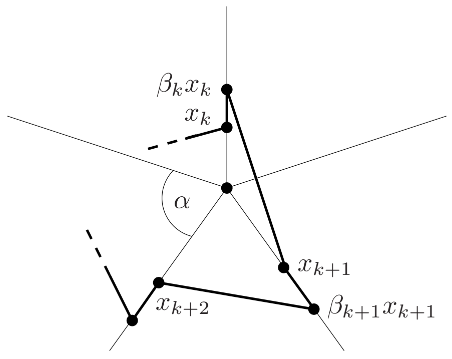


Figure 4: A bundle of n rays and the representation of a strategy.

Therefore we would like to make an approximation and represent a strategy as follows. At the k-th step, we hit a ray, say ray i, at distance x_k and leave the ray at distance $\beta_k x_k$ with $\beta_k \geq 1$. Therefore we move a distance $\beta_k x_k - x_k$ along the ray i and then we move to the next ray within a distance $\sqrt{(\beta_k x_k)^2 - 2\cos(\alpha)\beta_k x_k x_{k+1} + (x_{k+1})^2}$, see Figure 4. Let us assume that the ray i for x_k and $\beta_k x_k$ is visited the next time at index J_k. The worst-case occurs if we did not see the goal at the ray i up to distance x_k and find the goal at step x_{J_k} on i arbitrarily close behind $\beta_k x_k$. The competitive factor is bigger than

$$\frac{1}{\beta_k x_k}\Bigg(x_{J_k} - \beta_k x_k + \sum_{i=1}^{J_k-1} \beta_i x_i - x_i + \sqrt{(\beta_i x_i)^2 - 2\cos(\alpha)\beta_i x_i x_{i+1} + (x_{i+1})^2}\Bigg),$$

where $x_{J_k} - \beta_k x_k$ denotes the movement to goal.

By simple trigonometry the shortest distance from $\beta_k x_k$ to a neighboring ray is given by $\beta_k x_k \sin\left(\frac{2\pi}{n}\right)$. Fortunately, this distance is smaller than the distance $\sqrt{(\beta_k x_k)^2 - 2\cos(\alpha)\beta_k x_k x_{k+1} + (x_{k+1})^2}$ to any other ray. Therefore a lower bound on the above worst-case factor is given by

$$-1 + \frac{\sum_{i=1}^{J_k-1} \beta_k x_k \sin\left(\frac{2\pi}{n}\right)}{\beta_k x_k}$$

Altogether, we have to find a lower bound for $\frac{\sum_{i=1}^{J_k-1} \beta_k x_k}{\beta_k x_k}$ where J_k always denotes the next visit of the ray of x_k. Fortunately, this problem also represents the competitive analysis for the m ray problem where we can only move along the rays. It was shown by [7] and [2] that for this problem there is an optimal strategy that visits the rays with increasing depth and in a periodic order, that is $J_k = k+n$ and $i = k$. The best strategy is given by $f_i = (n/(n-1))^i$. Altogether, this results in a function

$$(n-1)\sin\left(\frac{2\pi}{n}\right)\left(\frac{n}{n-1}\right)^n$$

for n rays. We can make n arbitrarily big because our construction is valid for every n. Note, that we also have a lower bound for the problem of searching a point in the plane, the lower bound is close to the factor of the spiral.

Theorem 4 *For the ray search problem there is no strategy that achieves a better factor than*

$$-1 + \lim_{n\to\infty}(n-1)\sin\left(\frac{2\pi}{n}\right)\left(\frac{n}{n-1}\right)^n = -1 + 17.079\ldots$$

Additionally, every strategy for searching a point in the plane achieves a competitive factor bigger then $17.079\ldots$ *and the optimal spiral achieves a factor of* $17.289\ldots$

5 Conclusion

We considered the problem of searching a ray and its origin under the competitive framework.

If the ray starts on a known ray r' and is also perpendicular to r' we will find the origin within a path length of $1.059\ldots$ times the shortest path to the origin. This factor is optimal.

In general a logarithmic spiral solves the task within a competitive factor of $22.51\ldots$ whereas a lower bound of $16.079\ldots$ is given.

The lower bound construction can also be used if it is not necessary to visit the origin and if the corresponding line of every ray goes through the starting point. For this sub-problem a competitive strategy with factor $17.289\ldots$ was already known. We can proof that there is no strategy with a factor better than $17.079\ldots$ in this setting.

There are still some gaps between the lower and upper bounds of the factors which have to be closed.

References

[1] S. Alpern and S. Gal. *The Theory of Search Games and Rendezvous.* Kluwer Academic Publications, 2003.

[2] R. Baeza-Yates, J. Culberson, and G. Rawlins. Searching in the plane. *Inform. Comput.*, 106:234–252, 1993.

[3] P. Berman. On-line searching and navigation. In A. Fiat and G. Woeginger, editors, *Competitive Analysis of Algorithms.* Springer-Verlag, 1998.

[4] E. D. Demaine, S. P. Fekete, and S. Gal. Online searching with turn cost. *Theor. Comput. Sci.*, Submitted.

[5] A. Fiat and G. Woeginger, editors. *On-line Algorithms: The State of the Art*, volume 1442 of *Lecture Notes Comput. Sci.* Springer-Verlag, 1998.

[6] R. Fleischer, T. Kamphans, R. Klein, E. Langetepe, and G. Trippen. Competitive online approximation of the optimal search ratio. ESA 2004.

[7] S. Gal. *Search Games*, volume 149 of *Mathematics in Science and Engeneering.* Academic Press, New York, 1980.

[8] M. Hammar, B. J. Nilsson, and S. Schuierer. Parallel searching on m rays. *Comput. Geom. Theory Appl.*, 18:125–139, 2001.

[9] C. Hipke, C. Icking, R. Klein, and E. Langetepe. How to find a point on a line within a fixed distance. *Discrete Appl. Math.*, 93:67–73, 1999.

[10] C. Icking, T. Kamphans, R. Klein, and E. Langetepe. On the competitive complexity of navigation tasks. In H. Bunke, H. I. Christensen, G. D. Hager, and R. Klein, editors, *Sensor Based Intelligent Robots*, volume 2238 of *LNCS*, pages 245–258, Berlin, 2002. Springer.

[11] M.-Y. Kao, J. H. Reif, and S. R. Tate. Searching in an unknown environment: An optimal randomized algorithm for the cow-path problem. *Inform. Comput.*, 133(1):63–79, 1996.

[12] E. Koutsoupias, C. H. Papadimitriou, and M. Yannakakis. Searching a fixed graph. In *Proc. 23th Internat. Colloq. Automata Lang. Program.*, volume 1099 of *LNCS*, pages 280–289. Springer, 1996.

[13] E. Langetepe. *Design and Analysis of Strategies for Autonomous Systems in Motion Planning.* PhD thesis, Department of Computer Science, FernUniversität Hagen, 2000.

[14] A. López-Ortiz and S. Schuierer. The ultimate strategy to search on m rays? *Theor. Comput. Sci.*, 261(2):267–295, 2001.

[15] J. S. B. Mitchell. Geometric shortest paths and network optimization. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 633–701. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000.

[16] N. S. V. Rao, S. Kareti, W. Shi, and S. S. Iyengar. Robot navigation in unknown terrains: introductory survey of non-heuristic algorithms. Technical Report ORNL/TM-12410, Oak Ridge National Laboratory, 1993.

[17] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Commun. ACM*, 28:202–208, 1985.

On Optimizing Multi-Sequence Functionals for Competitive Analysis

Tom Kamphans[*] Elmar Langetepe[*]

Abstract

The efficiency of an online motion planning algorithm often is measured by a constant competitive factor C. Competitivity means, that the cost of an C-competitive online strategy with incomplete information is only C times worse than the optimal offline solution with full information. If a strategy is represented by an infinite sequence $X = f_1, f_2, \ldots$, the problem of finding a strategy with minimal C often results in minimizing functionals F_k in X. There are two main paradigm for finding an optimal sequence $f_1, f_2, \ldots$ that minimizes F_k for all k. Namely, optimality of the exponential function and equality approach. If the strategy has to be defined by more than one interacting sequence both approaches may fail. We show a simple motion planning example with two interacting sequences and present its solution.

1 Introduction

Search games, i. e. games where two players, a searcher and a hider, compete with each other, are studied in many variations in the last 60 years since the first work by Koopman in 1946. For example, Bellman [3] introduced the search for an immobile hider located on the real line with a known probability distribution, Gal [5] and independently Baeza-Yates et. al. [2] solve this problem for a uniformly distributed location of the hider. The book by Gal [5] and the reissue by Alpern and Gal [1] gives a comprehensive overview on results on search games.

The length of the searcher's trajectory is often used as payoff of a search game. To get a finite value for the game, we use the competitive framework, that is, we compare the length of the searcher's trajectory to the shortest distance to the hider. Gal [5] calls this a normalized cost function. More precisely, we call a search strategy *competitive* with a factor C, if $|\pi| \leq C \cdot |\pi_{\text{opt}}|$ holds for every location of the hider, where $|\pi|$ denotes the length of the searcher's path and $|\pi_{\text{opt}}|$ the shortest path. The competitive framework was introduced by Sleator and Tarjan [15], and used in many settings since then, see for example the survey by Fiat and Woeginger [4] or, for the field of online robot motion planning, see the surveys [12, 8].

In most settings, a search strategy, X, can be given by a sequence of values $f_1, f_2, f_3, \ldots$ denoting e. g. the exploration depth in the i-th iteration step, and the competitive factor can be given by a functional $F_k(X)$, where k denotes the number of iteration steps. Since we want to minimize the costs, we have to find a strategy that minimizes F_k. There are two commonly used methods to find such a strategy. The first is, to show that the functionals F_k fulfill certain conditions—see below. Gal [5] showed, that a strategy with $f_i = a^i$ minimizes the F_k's, and we have just to find an appropriate a using simple analysis. Another method is, to show that there is a strategy X' that achieves the optimal competitive factor, C, not only asymptotically, but *exactely in every step.* With this we establish a closed form for X'. Both methods, however, work well for strategies that can be given by *one* sequence f_i. If we have search games that rely on more than one parameter, the theorem by Gal may not be applicable and the equality approach may fail.

We introduce a variation of the search for a goal on an infinite line: the goal is located on two rays emanating from the searcher's origin, with an angle γ between the rays. The searcher can move in the free space between the two rays, and finds the goal as soon as it reaches the goals position or a position behind the goal, seen from the origin. A reasonable strategy to solve this problem can be described by *two* sequences, α_i and β_i, see Figure 1.

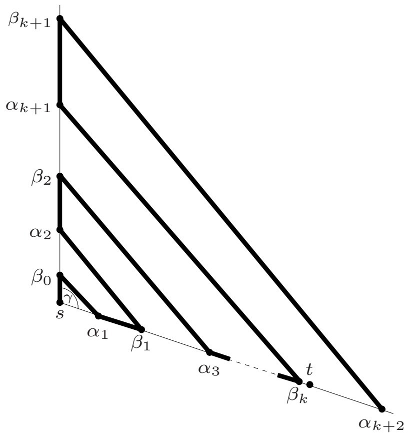


Figure 1: Representation of a strategy with two sequences.

[*]University of Bonn, Institute of Computer Science I, D-53117 Bonn, Germany.

2 Exponential function and equality approach

A strategy for searching an unknown goal on an infinite ray can be represented by an infinite sequence $F = f_1, f_2, \ldots$ of positive values. The agent moves f_1 steps to the left of the start, returns to the goal, moves f_2 steps to the right of the start point, returns to the goal and so on. The worst-case for the competitive factor occurs, if we miss the goal at step k by an ϵ, return to the start, move f_{k+1} to the other direction, return to the start and find the goal in the $(k+2)$-th step at distance $f_k + \epsilon$. Thus, the worst-case factor is given by $\frac{2\sum_{i=1}^{k+1} f_i + f_k}{f_k} = 1 + 2\frac{\sum_{i=1}^{k+1} f_i}{f_k}$, and it suffices to find a sequence $X = f_1, f_2, \ldots$ that minimizes $F_k(f_1, f_2, \ldots) := \frac{\sum_{i=1}^{k+1} f_i}{f_k}$ for all k. Note, that we additionally have to take care for the first movement. We assume that the goal is at least one step away from the start which gives the additional inequality $2f_1 \leq C \cdot 1$.

Let $F(X) := \lim_{k\to\infty} F_k(X)$ more precisely, we are searching for a strategy X with

$$\inf_Y \sup_k F_k(Y) = C \text{ and } \sup_k F_k(X) = C\,.$$

Two main approaches for solving the given problem are discussed in the literature. We briefly repeat the main ideas.

Optimality of the exponential function: The functional F_k is continous and unimodal. Unimodality is defined by $F_k(A \cdot X) = F_k(X)$ and $F_k(X+Y) \leq \max\{F_k(X), F_k(Y)\}$ for every constant A and two sequences X and Y. Unimodality means that scalar multiplication and the addition of two sequences does not increase the value of the functional.

If F_k additionally fulfills some other reasonable properties, it was shown by [1, 5, 14] that an exponential function minimizes F_k, or more precisely

$$\sup_k F_k(X) \geq \inf_a \sup_k F_k(A_a)$$

where $A_a = a^0, a^1, a^2, \ldots$ and $a > 0$. Altogether, the problem of searching a point on a line is solved by

$$\inf_a \frac{\sum_{i=1}^{k+1} a^{i-1}}{a^{k-1}} = \frac{2^2}{2-1} = 4\,.$$

Equality approach: On the other hand some authors [7, 10, 11, 9, 13] suggest to adjust an optimal strategy $X = f_1, f_2, \ldots$ with $F_k(X) \leq C$ to an optimal strategy $X' = f_1', f_2', \ldots$ with $F_k(X') = C$ where C is the (probably unknown) best achievable factor. It can be shown that for the 2-ray search problem such a strategy X' exists. The main reason is that $F_k = \frac{\sum_{i=1}^{k} f_i}{f_k}$ increases in f_k and F_l decrease in f_k for all $l \neq k$. Therefore we can inductively adjust a given optimal strategy X. How will we find the optimal strategy? One will try to retrieve a recurrence for the values of X' from the equation $F_k(X') = C = F_{k+1}(X')$

For the 2-ray search problem we assume that $X = f_1, f_2, \ldots$ achieves equality in every step. Thus, we have $\sum_{i=1}^{k+2} f_i = Cf_{k+1}$ and $\sum_{i=1}^{k+1} f_i = Cf_k$. Subtracting both sides gives the recurrence $f_{k+2} = C(f_{k+1} - f_k)$ for $k = 1, 2, \ldots$ Obtaining positive solutions for recurrences can be solved by analytic means, see [6]. It can be shown that for $C < 4$ there is no positive sequence that fulfills the given recurrence $f_{k+2} = C(f_{k+1} - f_k)$. Additionally, for $f_i := (i+1)2^i$ we have $f_{k+2} = (k+3)2^{k+1} = 4(f_{k+1} - f_k) = (3k+4)2^{k+2} - (k+1)2^{k+2}$ and $C = 4$ is optimal.

Altogether, we have two optimal solutions stemming from different paradigm. In the following we will combine both paradigm in order to solve more sophisticated functionals.

3 A simple online problem

We discuss a variant of the 2-ray search problem. We are searching for a target on 2-rays r_1 and r_2, emanating from a single source s and building an angle γ, Figure 1. It is allowed to move in the plane from one ray to the other. A target t at distance $|st|$ on ray r_i can be detected, if we visit a point p on r_i with $|pt| \geq |st|$. Therefore a search strategy need not visit all points on the ray. We denote the problem as the 2γ-ray-scan problem. The distance to the goal and the goal's ray is not known in advance. The best offline strategy moves directly along the corresponding ray to the goal.

[2, 1] dicuss a similar variant without looking back, the goal is detected only if the goal is visited. This variant can be described by a single sequence and was solved in [1] with an exponential function.

A strategy for the 2γ-ray-scan problem can be represented as follows. We start with ray r_1 and move along it for a while up to distance β_0. Then we will move on a straight line to a point at distance α_1 on ray r_2 and move along r_2 for a while until leaving the ray at distance β_1 and so on. Altogeher, we have a sequence of leave points and a sequence of hit points and we denote every hit point by its distance α_i and every leave point by its distance β_i, see 1. For convenience, we set $\alpha_0 = 0$. A reasonable strategy fulfills $\beta_{i-2} \leq \alpha_i \leq \beta_i$.

The worst-case for the competitive factor occurs, if we miss the goal by an ϵ on the first ray, move to the second ray and find the goal after returning back to the first ray. Setting $\alpha_0 = 0$ we have to minimize the following functional

$$G_k([(\alpha_0, \alpha_1, \ldots), (\beta_0, \beta_1, \ldots)] :=$$

$$\frac{\sum_{i=0}^{k+1} \beta_i - \alpha_i + \sqrt{\alpha_{i+1}^2 - 2\alpha_{i+1}\beta_i \cos\gamma + \beta_i^2}}{\beta_k}\,.$$

One can proof the optimality of a exponential function for β_i by showing the prerequisites in [5, 1, 14].

Theorem 1 *If there exists a strategy* $X = [(\alpha_0, \alpha_1, \ldots), (\beta_0, \beta_1, \ldots)]$ *such that*

$$\inf_Y \sup_k G_k(Y) = C \text{ and } \sup_k G_k(X) = C,$$

then there is always a solution $Z = [(\alpha_0, \alpha_1, \ldots), (1, \beta^1, \beta^2, \ldots)]$ *so that*

$$\inf_Y \sup_k G_k(Y) = C \text{ and } \sup_k G_k(Z) = C\,.$$

Proof. We show that the unimodality condition holds, the other prerequisites are easily fulfilled. Scalar multiplication is simply satisfied.

It suffices to show that $\sqrt{a_{i+1}^2 - 2a_{i+1}b_i \cos\gamma + b_i^2} + \sqrt{c_{i+1}^2 - 2c_{i+1}d_i \cos\gamma + d_i^2}$ is bigger than $\sqrt{(a_{i+1}+c_{i+1})^2 - 2(a_{i+1}+c_{i+1})(b_i+d_i)\cos\gamma + + (b_i+d_i)^2}$ which is the triangle inequality for the vectors $((a_{i+1}-b_i)\cos(\gamma/2), (a_{i+1}+b_i)\sin(\gamma/2))$ and $((c_{i+1}-d_i)\cos(\gamma/2), (c_{i+1}+d_i)\sin(\gamma/2))$.

Now, let $G_k([(a_0,\ldots),(b_0,\ldots)]) \leq D$ and $G_k([(c_0,\ldots),(d_0,\ldots)]) \leq D$ then we have $\sum_{i=0}^{k+1} b_i - a_i + \sqrt{a_{i+1}^2 - 2a_{i+1}b_i\cos\gamma + b_i^2} + + \sum_{i=0}^{k+1} c_i - d_i + \sqrt{c_{i+1}^2 - 2c_{i+1}d_i\cos\gamma + d_i^2} \leq D(b_k + d_k)$ and the left-hand side of the inequality is greater than or equal to $\sum_{i=0}^{k+1}(b_i+d_i) - (a_i + c_i)\sqrt{(a_{i+1}+c_{i+1})^2 - 2(a_{i+1}+c_{i+1})(b_i+d_i)\cos\gamma + + (b_i+d_i)^2}$ which completes the proof. □

Unfortunately, this result will not give us a strategy because there is a second sequence. We still have to optimize

$G_k([(\alpha_0, \alpha_1, \ldots), \beta]) :=$

$$\frac{\sum_{i=0}^{k+1} \beta^i - \alpha_i + \sqrt{\alpha_{i+1}^2 - 2\alpha_{i+1}\beta^i\cos\gamma + \beta_i^2}}{\beta^k}$$

On the other hand if we can show that there is a strategy with $G_k([(\alpha_0, \alpha_1, \ldots), (\beta_0, \beta_1, \ldots)]) = C$ for all $k > l$, the subtraction of two equations

$$\sum_{i=0}^{k} \beta_i - \alpha_i + \sqrt{\alpha_{i+1}^2 - 2\alpha_{i+1}\beta_i\cos\gamma + \beta_i^2} = C\beta_{k-1}$$

$$\text{and} \sum_{i=0}^{k+1} \beta_i - \alpha_i + \sqrt{\alpha_{i+1}^2 - 2\alpha_{i+1}\beta_i\cos\gamma + \beta_i^2} = C\beta_k$$

results in a recurrence

$$\beta_{k+1} - \alpha_{k+1} + \sqrt{\alpha_{k+2}^2 - 2\alpha_{k+2}\beta_{k+1}\cos\gamma + \beta_{k+1}^2}$$
$$= C(\beta_k - \beta_{k-1})\,.$$

Unfortunately, this is a non-linear recurrence and cannot be solved easily.

4 Combining two paradigms

We suggest to combine both approaches. First, we show that at least for $\gamma \leq \pi/2$ there is indeed a strategy $[(\alpha_0, \alpha_1, \ldots), (\beta_0, \beta_1, \ldots)]$ so that $\sum_{i=0}^{k+1} \beta_i - \alpha_i + \sqrt{\alpha_{i+1}^2 - 2\alpha_{i+1}\beta_i\cos\gamma + \beta_i^2} = C\beta_k$ for all $k \geq 1$.

Then we make use of the subtraction idea above and obtain

$$C = \left(\frac{1}{\beta_k - \beta_{k-1}}\right)\left(\beta_{k+1} - \alpha_{k+1} + \sqrt{\alpha_{k+2}^2 - \alpha_{k+2}\beta_{k+1}2\cos\gamma + \beta_{k+1}^2}\right) \quad (1)$$

which again gives a functional but without a sum in the denominator. Now, we solve this functionals by showing that the prerequisites of the exponential solution is again fulfilled.

Lemma 2 *For* $\gamma \geq \frac{\pi}{2}$ *there is always an optimal solution* $X = [(\alpha_0, \alpha_1, \ldots), (\beta_0, \beta_1, \ldots)]$ *that achieves* $\sum_{i=0}^{k+1} \beta_i - \alpha_i + \sqrt{\alpha_{i+1}^2 - 2\alpha_{i+1}\beta_i\cos\gamma + \beta_i^2} = C\beta_k$ *for all* $k \geq 0$.

Proof. We show the property by induction. We adjust a given strategy $[(\alpha_0, \alpha_1, \ldots), (\beta_0, \beta_1, \ldots)]$ to a strategy $[(\alpha_0, \alpha_1, \ldots), (\beta'_0, \beta'_1, \ldots)]$ such that all β'_i decrease but are still positive.

For $k = 0$ we may have $\sum_{i=0}^{2} \beta_i - \alpha_i + \sqrt{\alpha_{i+1}^2 - 2\alpha_{i+1}\beta_i\cos\gamma + \beta_i^2} < C\beta_0$ for the optimal C. Thus, we decrease β_0 to $\beta'_0 := \beta_0 - \epsilon$ until we obtain equality. The distance $\sqrt{\alpha_1^2 - 2\alpha_1\beta_0\cos\gamma + \beta_0^2}$ decreases for $\gamma \geq \frac{\pi}{2}$ and $G_k([(\alpha_0, \alpha_1, \ldots), (\beta'_0, \beta_1, \ldots)])$ gets smaller for all $k > 1$. β'_0 is positive since we have to subsume the distance from β_1 to $\alpha_2 > \beta_0$.

Now let us assume that the property holds for all $l \leq k-1$ and let $G_k([(\alpha_0, \alpha_1, \ldots), (\beta'_0, \beta'_1, \ldots)]) < C$. We again can decrease β_k to $\beta'_k := \beta_k - \epsilon$ until we have equality. For $\alpha_k < \beta_k$ we decrease the denominator of $G_l([(\alpha_0, \alpha_1, \ldots), (\beta'_0, \beta'_1, \ldots)])$ for all $l \geq k$ and for $k-1$ but not for $l \leq k-2$. Note, that $\sqrt{\alpha_{k+1}^2 - 2\alpha_{k+1}\beta_k\cos(\gamma) + \beta_k^2}$ decreases for $\gamma \geq \frac{\pi}{2}$. If $\alpha_k = \beta'_k$ is reached, we additionally decrease the denominator for $l = k-2$. Note, that this is true, since for $\gamma \geq \frac{\pi}{2}$ the distances $\sqrt{\alpha_{k+1}^2 - 2\alpha_{k+1}\beta'_k\cos\gamma + \beta'^2_k}$ and $\sqrt{\beta'^2_k - 2\beta'_k\beta_{k-1}\cos\gamma + \beta_{k-1}^2}$ decrease.

Finally we will achieve equality for $G_k([(\alpha_0, \alpha_1, \ldots), (\beta'_0, \beta'_1, \ldots)])$ with β'_k. β'_k is positive, since we have to subsume the distance β_{k+1} to α_{k+2}. Unfortunately, $G_l([(\alpha_0, \alpha_1, \ldots), (\beta'_0, \beta'_1, \ldots)]) < C$ may hold for $l = k-2$ and $l = k-1$. By induction hypothesis, we adjust the strategy again such that $G_l([(\alpha_0, \alpha_1, \ldots), (\beta'_0, \beta'_1, \ldots)]) = C$ for all $l \leq k-1$. Now it again may happen that $G_k([(\alpha_0, \alpha_1, \ldots), (\beta'_0, \beta'_1, \ldots)]) < C$ holds.

We repeat the above process, since the vector $(\beta'_0, \beta'_1, \ldots, \beta'_k)$ always decrease by construction but still remain positive, the vector $(\beta'_0, \beta'_1, \ldots, \beta'_k)$ finally has to run into a positive limit that fulfills equality. □

Note, that for the equality strategy we still have $\beta_{k-2} \leq \alpha_k \leq \beta_k$. The functional (1) for $k \geq 1$ fulfills the criterion for the exponential function which can be proven by the same arguments as in the proof of Theorem 1.

Lemma 3 *Functional (1) can be optimized by an exponential function, that is, there is an optimal strategy with factor* $\left(\frac{1}{\beta^k - \beta^{k-1}}\right)\left(\beta^{k+1} - \alpha_{k+1} + \right.$ $\left. + \sqrt{\alpha_{k+2}^2 - 2\alpha_{k+2}\beta^{k+1}\cos\gamma + \beta_{k+1}^2}\right)$.

Now we can set $\alpha_i = C_i\beta^i$ for $C_i \in [1, \frac{1}{\beta^2}]$ and it remains to optimize

$$\frac{\beta^2\left(1 - C_i + \sqrt{(C_{i+1}\beta)^2 - 2C_{i+1}\beta\cos\gamma + 1}\right)}{\beta - 1}$$

for β and C_i and C_{i+1}. In the next iteration step C_{i+1} takes over the role of C_i. Therefore the best we can do is that we fix C_i by a constant D which means that we have to optimize

$$\frac{\beta^2\left(1 - D + \sqrt{(D\beta)^2 - 2D\beta\cos\gamma + 1}\right)}{\beta - 1}$$

for β and $D \in [1, 1/\beta^2]$.

Theorem 4 *The strategy for the 2γ-ray-scan problem for $\gamma \geq \pi/2$ can be achieved by minimizing*

$$f(\beta, \gamma, D) := \frac{\beta^2\left(1 - D + \sqrt{(D\beta)^2 - 2D\beta\cos\gamma + 1}\right)}{\beta - 1}$$

over β and $D \in [1, 1/\beta^2]$.

It can be shown that there is a reasonable $D(\gamma, \beta)$ that minimizes $f(\beta, \gamma, D)$. If $D(\gamma, \beta)$ is inside $[1, 1/\beta^2]$ then we have to compare the minima over β for $D = 1$, $D = 1/\beta^2$ and $D = D(\gamma, \beta)$. $D = 1/\beta^2$ represents a strategy without any gaps. $D = 1$ represents a strategy that does not slip along the rays and $D = D(\gamma, \beta)$ represents something in between.

For example for $\gamma = \pi/2$ we can show that

$$f(\gamma, \beta, D(\gamma, \beta)) = \frac{\beta^2\left(1 - \frac{1}{\sqrt{-1+\beta^2}\beta} + \sqrt{\frac{1}{-1+\beta^2} + 1}\right)}{\beta - 1}$$

which is minimal for $\beta = 1.839\ldots$ and $D = 0.352\ldots \in [1, 0.295\ldots]$. The optimal competitive factor is given by $7.413\ldots$ wheras the optimal factor for $D = 1/\beta^2$ is $7.472\ldots$

For $\gamma = \pi$ we obtain the *normal* doubling strategy, which is represented by $\beta = 2$ and $D = 1/4$.

5 Conclusion

We have shown that for solving a double sequence functional it is useful to combine the methods of equality approach and the optimality of the exponential function. With the help of multi-sequence functionals more sophisticated strategies can be represented. We think that the presented method can be extended to functionals with more than two sequences and that it should be possible to iterate the combination process more than once.

References

[1] S. Alpern and S. Gal. *The Theory of Search Games and Rendezvous*. Kluwer Academic Publications, 2003.

[2] R. Baeza-Yates, J. Culberson, and G. Rawlins. Searching in the plane. *Inform. Comput.*, 106:234–252, 1993.

[3] R. Bellman. An optimal search problem. *SIAM Rev.*, 274(5), 1963.

[4] A. Fiat and G. Woeginger, editors. *On-line Algorithms: The State of the Art*, volume 1442 of *Lecture Notes Comput. Sci.* Springer-Verlag, 1998.

[5] S. Gal. *Search Games*, volume 149 of *Mathematics in Science and Engeneering*. Academic Press, New York, 1980.

[6] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics*. Addison-Wesley, Reading, MA, second edition, 1994.

[7] C. Hipke, C. Icking, R. Klein, and E. Langetepe. How to find a point on a line within a fixed distance. Technical Report 220, Department of Computer Science, FernUniversität Hagen, Germany, 1997.

[8] C. Icking, T. Kamphans, R. Klein, and E. Langetepe. On the competitive complexity of navigation tasks. In H. Bunke, H. I. Christensen, G. D. Hager, and R. Klein, editors, *Sensor Based Intelligent Robots*, volume 2238 of *LNCS*, pages 245–258, Berlin, 2002. Springer.

[9] T. Kamphans and E. Langetepe. Optimal competitive online ray search with an error afflicted robot. ESA 2004.

[10] E. Langetepe. *Design and Analysis of Strategies for Autonomous Systems in Motion Planning.* PhD thesis, Department of Computer Science, FernUniversität Hagen, 2000.

[11] A. López-Ortiz and S. Schuierer. The ultimate strategy to search on m rays? *Theor. Comput. Sci.*, 261(2):267–295, 2001.

[12] N. S. V. Rao, S. Kareti, W. Shi, and S. S. Iyengar. Robot navigation in unknown terrains: introductory survey of non-heuristic algorithms. Technical Report ORNL/TM-12410, Oak Ridge National Laboratory, 1993.

[13] S. Schuierer. Searching on m bounded rays optimally. Technical Report 112, Institut für Informatik, Universität Freiburg, Germany, 1998.

[14] S. Schuierer. Lower bounds in on-line geometric searching. *Comput. Geom. Theory Appl.*, 18:37–53, 2001.

[15] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Commun. ACM*, 28:202–208, 1985.

Minimizing Local Minima in Terrains with Higher-Order Delaunay Triangulations

Thierry de Kok* Marc van Kreveld† Maarten Löffler‡

Abstract

We show that triangulating a set of points with elevations such that the number of local minima of the resulting terrain is minimized is NP-hard for degenerate point sets. The same result applies when there are no degeneracies for higher-order Delaunay triangulations. Two heuristics are presented to minimize the number of local minima for higher-order Delaunay triangulations, and they are compared experimentally.

1 Introduction

A fundamental geometric structure in computational geometry is the triangulation. It is a partitioning of a point set or region of the plane into triangles. A triangulation of a point set P partitions the convex hull of P into triangles whose vertices are exactly the points of P. There are many different ways in which one can define the quality of a triangulation of a set of points. A criterion that is always important for triangulations is the nice shape of the triangles. This can be formalized in several ways [2, 3]. In this paper, nice shape is formalized by *higher-order Delaunay triangulations* [4]. They provide a class of triangulations that are all reasonably well-shaped, depending on a parameter k.

Definition 1 *A triangle in a point set P is* order-k *if its circumcircle contains at most k points of P. A triangulation of a set P of points is an* order-k Delaunay triangulation *if any triangle of the triangulation is order-k.*

So a Delaunay triangulation is an order-0 Delaunay triangulation. For any positive integer k, there can be many different order-k Delaunay triangulations. We also define the *useful order* of an edge as the lowest order of a triangulation that includes this edge.

When using triangulations for terrain modeling, one should realize that terrains are formed by natural processes. This implies that there are linear depressions (valleys) formed by water flow, and very few local minima occur [7]. Local minima can be caused by erroneous triangulation: an edge may stretch from one side of a valley to the opposite side. Such an edge is an artificial dam, and upstream from the dam in the valley, a local minimum appears. It is generally an artifact of the triangulation. Therefore, minimizing local minima is an optimization criterion for terrain modeling.

This paper discusses triangulations of a point set P of which elevations are given. The objective is to triangulate P with an order-k Delaunay triangulation, such that there are as few local minima as possible. In Section 2 we show that over all possible triangulations, minimizing local minima is NP-hard. This result relies heavily on degenerate point sets. For order-k Delaunay triangulations, NP-hardness can also be shown for non-degenerate point sets, for $k = \Omega(n^\epsilon)$ and $k = O(n^{1/4})$. (For $k = 1$, an $O(n \log n)$ time algorithm that minimizes local minima was given in [4].) Then we discuss two heuristics for minimizing local minima. In Sections 3 and 4 we present the flip and hull heuristics and their efficiency. The latter was introduced before in [4]; here we give a more efficient algorithm. In Section 5 we apply the two heuristics on various terrains to examine the possibilities of higher-order Delaunay triangulations to reduce local minima, and to test which of the two heuristics is better.

2 NP-hardness

For a set P of n points in the plane, it is easy to compute a triangulation that minimizes the number of local minima if there are no degeneracies. Assume p is the lowest point. Connect every $q \in P\backslash\{p\}$ with p to create a star network with p as the center. Complete this set of edges to a triangulation in any way. Since every point but p has a lower neighbor, no point but p can be a local minimum. Hence, this triangulation is one that minimizes the number of local minima. When many degeneracies are present, minimizing the number of local minima is NP-hard.

Theorem 1 *Let P be a set of n points in the plane, and assume that the points have elevations. It is NP-hard to triangulate P with the objective to minimize the number of local minima of the polyhedral terrain.*

*Institute of Information and Computing Sciences, Utrecht University, takok@cs.uu.nl

†Institute of Information and Computing Sciences, Utrecht University, marc@cs.uu.nl

‡Institute of Information and Computing Sciences, Utrecht University, mloffler@cs.uu.nl

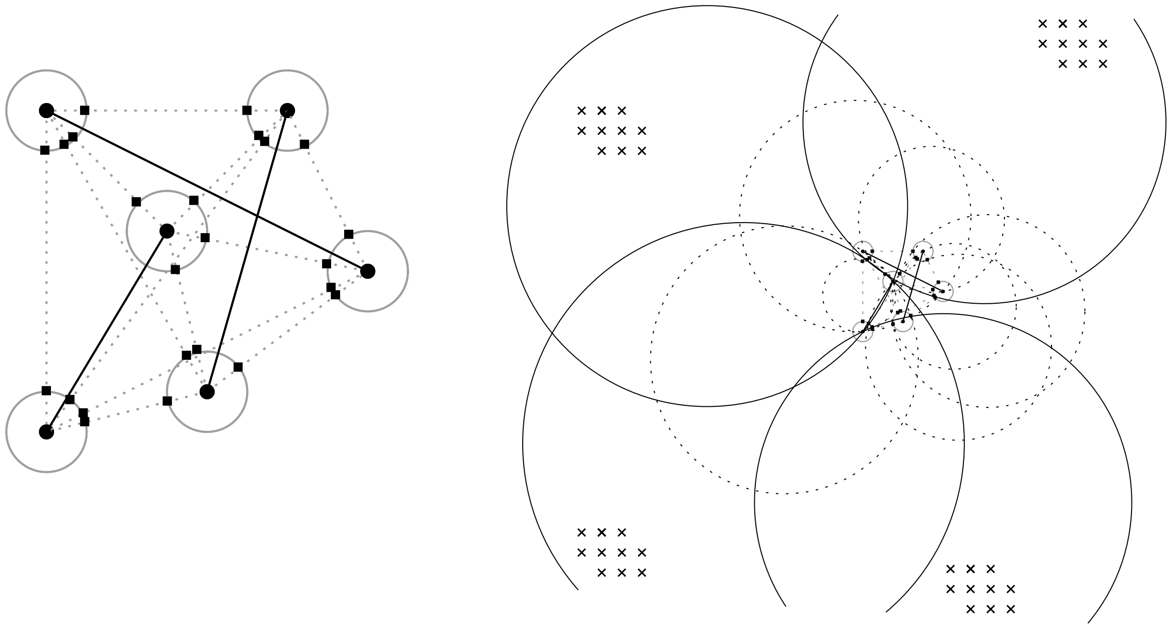


Figure 1: Left, construction for the NP-hardness proof. Square points are in H, cirular points are in P. Right, NP-hardness for higher-order Delaunay triangulations. Crosses are points at least $4r$ from the points in $P \cup H$.

Proof. By reduction from maximum size non-intersecting subset in a set of line segments [1]. Let S be any set of n line segments in the plane, and assume all $2n$ endpoints are disjoint (this can easily be enforced by extending segments slightly). Let P be the set of the $2n$ endpoints. Let ϵ be the smallest distance between two points in P. For every point $p \in P$, let $C(p)$ be a circle centered at p with radius $\epsilon/3$. If p' is the point in P such that pp' is a segment of S, then for every $q \in P \backslash \{p, p'\}$, place a point at the intersection of pq and $C(p)$. We call these points *shields*, because they prevent p and q from being connected by a line segment in any triangulation. Let H be the set of shields.

We assign elevations as follows. For every segment in S, one endpoint isassigned elevation 1 and the other isassigned elevation 2. Every shield in H isassigned elevation 3. By the choice of shields, every point with elevation 1 is a local minimum, and no point with elevation 3 can be a local minimum. A point with elevation 2 is a local minimum if and only if the segment from S that connects p' to the other endpoint p is in the triangulation. Hence, the maximum non-intersecting subset of S corresponds one-to-one with the points with elevation 2 that are local minimum. Since the number of shields is quadratic, NP-hardness follows directly. □

Based on the construction in the proof above, we can show NP-hardness of minimizing the number of local minima for higher-order Delaunay triangulations even when no degeneracies are present.

Corollary 2 *Let P be a set of n points in the plane such that no three points of P lie on a line, and assume that the points have elevations. For any $0 < \epsilon < 1$ and some $c > 0$, it is NP-hard to compute a k-th order Delaunay triangulation that minimizes the number of local minima of the polyhedral terrain for $n^\epsilon \leq k \leq c \cdot n^{1/4}$.*

Proof. Start out with the proof of the theorem above; $|P| = 2n$ and $|H| = 2n(2n-1)$. Let r be the radius of the largest (finite) circle that passes through three points of $P \cup H$ of the construction. Note that r is at least half of the diameter of $P \cup H$. Let $\delta > 0$ be a value chosen such that if three non-colinear points from $P \cup H$ move over a distance δ, then their circle has radius at most $2r$. Both r and δ can be computed in cubic time. For every shield $h \in H$, displace it over a distance d where $0 < d < \delta$, and such that the radius of the circle through h and the two points of P for which h is a shield has radius at least $4r$. Place $|P| + |H| + 1 = 4n^2 + 1$ points inside this circle and at distance at least $4r$ from all points in $P \cup H$. Since the diameter of $P \cup H$ is at most $2r$, this is possible. These points make sure that the triangulation edge for which h was a shield, is still not possible in a $4n^2$-th order Delaunay triangulation: the useful order of the triangulation edge is too high. By construction, other edges between points in P and H are possible. The extra points get elevation 3 as well, and the problem of minimizing the number of local minima in $4n^2$-th order Delaunay triangulations is again the same as maximizing the size of a non-intersecting subset of S.

The number of points in the construction is $O(n^8)$. This gives the proof for $k = c \cdot n^{1/4}$ for some $c > 0$. For smaller values of k we simply place more points at distance at least $4r$. As long as $k = \Omega(n^\epsilon)$ the construction is polynomial. $\square$

3 The flip heuristic

Given a value of k, the flip heuristic repeatedly tests whether the diagonal of a convex quadrilateral in the triangulation can be flipped. It will be flipped if two conditions hold simultaneously: (i) The two new triangles are order-k Delaunay triangles. (ii) The new edge connects the lowest point of the four to the opposite point. A flip does not necessarily remove a local minimum, but cannot create one, and it can make possible that a later flip removes a local minimum.

Our algorithm to perform the flips starts with the Delaunay triangulation and $k' = 1$, then does all flips possible to obtain an order-k' Delaunay triangulation, then increments k' and repeats. This continues until $k' = k$.

We first deal with the maximum number of flips needed, and then we discuss the efficiency of the heuristic.

Lemma 3 *The flip heuristic terminates after at most $O(n^2)$ flips.*

Proof. Normalize the heights of the vertices to be integers in the range $1, \ldots, n$. Observe that this does not influence the flipping criterion. Consider the function $F(T)$ for a triangulation T:

$$F(T) = \sum_{\overline{uv} \in T} \min(u, v)\,.$$

Any flip decreases $F(.)$ with at least one, and $F(.)$ is at most $O(n^2)$ to begin with. $\square$

Lemma 4 *If an edge $\overline{ab}$ is in the triangulation, then the flip heuristic will never have an edge $\overline{cd}$ later with* $\min(c, d) \geq \min(a, b)$ *that intersects $\overline{ab}$.*

Proof. Assume without loss of generality that $a < b$ and $c < d$. Assume further that $a \leq c$, edge $\overline{ab}$ is in the triangulation T, and that $\overline{cd}$ is the first edge flipped into T that violates the property of the lemma. Before the flip, c and d must be in a convex quadrilateral where c is the lowest point of the four. The other two points, f and g, cannot be a because a is lower by assumption. Possibly, f or g is the same as b. The quadrilateral has edges $\overline{cf}, \overline{cg}, \overline{df}, \overline{dg}$, and two of them intersect $\overline{ab}$. But this contradicts the assumption that $\overline{cd}$ is the first edge violating the property. $\square$

An immediate consequence of the lemma above is that an edge that is flipped out of the triangulation cannot reappear. There are at most $O(nk)$ pairs of points in a point set of n points that give order-k Delaunay edges [4]. Therefore, we conclude:

Lemma 5 *The flip heuristic to reduce the number of local minima performs at most $O(nk)$ flips.*

To implement the flip heuristic efficiently, we maintain the set of all convex quadrilaterals in the current triangulation, with the order of the two triangles that would be created if the diagonal were flipped. The order of a triangle is the number of points in the circumcircle of the vertices of the triangle. Whenever a flip is done, we update the set of convex quadrilaterals. At most four are deleted and at most four new ones are created by the flip. We can find the order of the incident triangles by circular range counting queries. Since we are only interested in the count if the number of points in the circle is at most k, we implement circular range counting queries by point location in the order-k Voronoi diagram [6], taking $O(\log n + k)$ time per query. We conclude:

Theorem 6 *The flip heuristic to minimize the number of local minima in k-th order Delaunay triangulations on n points takes $O(nk^2 + nk \log n)$ time.*

4 The hull heuristic

The second heuristic for reducing the number of local minima is the hull heuristic. It was described in Gudmundsson et al. [4], and has an approximation factor of $\Theta(k^2)$ of the optimum. The hull heuristic adds a useful order-k Delaunay edge if it reduces the number of local minima. This edge may intersect several Delaunay edges, which are removed; the two holes in the triangulation that appear are retriangulated with the constrained Delaunay triangulation. No other higher-order Delaunay edges will be used that intersect the two holes. This guarantees that the final triangulation is order-k. It is known that two useful order-k Delaunay edges in general can give an order-$(2k-2)$ Delaunay triangulation [5], which is higher than allowed.

Here we give a slightly different implementation than in [4]. It is more efficient for larger values of k. Also, we include the adaptation that useful lower-order Delaunay edges are inserted first.

Assume that a point set P and an order k are given. We first compute the Delaunay triangulation $\mathcal{T}$ of P, and then compute the set E of all useful k-th order Delaunay edges, as in [4], in $O(nk \log n + nk^2)$ time. There are $O(nk)$ edges in E, and for each we have the lowest order $k' \leq k$ for which it is a useful order-k' Delaunay edge.

Next we determine the subset $P' \subseteq P$ of points that are a local minimum in the Delaunay triangulation.

	0	1	2	3	4	5	6	7	8	9	10
Calif. Hot Springs	47/47	43/43	33/31	29/26	25/20	24/19	23/18	21/18	18/16	18/16	17/15
Wren Peak	45/45	37/37	31/31	27/27	24/22	23/21	21/20	19/20	19/20	19/19	19/17
Quinn Peak	53/53	44/44	36/36	31/29	26/25	24/23	23/21	21/20	20/19	20/19	19/17
Sphinx Lakes	33/33	27/27	22/22	20/19	19/18	17/16	15/12	12/9	11/9	9/8	9/8
Split Mountain	24/24	17/17	14/14	9/9	9/9	9/9	8/8	7/8	6/7	6/6	5/4

Table 1: Results of the flip/hull heuristic for orders 0–10.

Then we determine the subset $E' \subseteq E$ of edges that connect a point of P' to a lower point. These steps trivially take $O(nk)$ time.

Sort the edges of E' by non-decreasing order. For every edge $e \in E'$, traverse $\mathcal{T}$ to determine the edges of $\mathcal{T}$ that intersect e. If any one of them is not a Delaunay edge or is a marked Delaunay edge, then we stop and continue with the next edge of E'. Otherwise, we remove all intersected Delaunay edges and mark all Delaunay edges of the polygonal hole that appears. Then we insert e and retriangulate the two polygons to the two sides of e using the Delaunay triangulation constrained to the polygons. We also mark these edges. Finally, we remove some edges from E'. If the inserted edge e made that a point $p \in P$ is no longer a local minimum, then we remove all other edges from E' where p is the highest endpoint.

Due to the marking of edges, no edge $e \in E'$ will be inserted if it intersects the hull of a previously inserted edge of E'. Also note that every edge of E' is treated in $O(k)$ time. We conclude:

Theorem 7 *The hull heuristic to minimize the number of local minima in k-th order Delaunay triangulations on n points takes $O(nk^2 + nk \log n)$ time.*

5 Experiments

Table 1 shows the number of local minima obtained after applying the flip and hull heuristics to five different terrains. The terrains roughly have 1800 vertices. The vertices were chosen by random sampling from elevation grids with about 100 times more points than the chosen sets.

The values in the table show that higher-order Delaunay triangulations indeed can give significantly fewer local minima than the standard Delaunay triangulation (0-th order). This effect is already clear at low orders, indicating that indeed, many local minima of Delaunay triangulations are caused by having chosen the wrong edges for the terrain (interpolation).

The difference in local minima between the flip and hull heuristics shows that the hull heuristic usually is a bit better, but there are some exceptions.

To test how good the results are, we also tested how many local minima of each terrain cannot be removed simply because there is no useful order-k Delaunay edge possible to a lower point. It turned out that the hull heuristic found the optimal order-k Delaunay triangulation in nearly all cases. Only in six cases we cannot be sure: there was one local minimum left that could potentially be removed.

6 Further research

It remains to be discovered whether minimizing the number of local minima in order-k Delaunay triangulations is already NP-hard for smaller values of k. This is unknown for $k \geq 2$ but significantly less than n^{ϵ}, for any small constant $\epsilon > 0$.

Another topic for further investigation is the removal of other artifacts from terrains. For example, for drainage applications, it is important to have the drainage network coincide with the triangulation edges, and not go over the middle of triangles.

References

[1] P. Agarwal and N. Mustafa. Independent set of intersection graphs of convex objects in 2D. In *Proc. SWAT 2004*, number 3111 in LNCS, pages 127–137, Berlin, 2004. Springer.

[2] M. Bern, H. Edelsbrunner, D. Eppstein, S. Mitchell, and T. S. Tan. Edge insertion for optimal triangulations. *Discrete Comput. Geom.*, 10(1):47–65, 1993.

[3] M. Bern and D. Eppstein. Mesh generation and optimal triangulation. In D.-Z. Du and F. K. Hwang, editors, *Computing in Euclidean Geometry*, volume 4 of *Lecture Notes Series on Computing*, pages 47–123. World Scientific, Singapore, 2nd edition, 1995.

[4] J. Gudmundsson, M. Hammar, and M. van Kreveld. Higher order Delaunay triangulations. *Comput. Geom. Theory Appl.*, 23:85–98, 2002.

[5] J. Gudmundsson, H. Haverkort, and M. van Kreveld. Constrained higher order Delaunay triangulations. *Comput. Geom. Theory Appl.*, to appear, 2005.

[6] E. Ramos. On range reporting, ray shooting and k-level construction. In *Proc. 15th Annu. ACM Symp. on Computational Geometry*, pages 390–399, 1999.

[7] B. Schneider. Geomorphologically sound reconstruction of digital terrain surfaces from contours. In T. Poiker and N. Chrisman, editors, *Proc. 8th Int. Symp. on Spatial Data Handling*, pages 657–667, 1998.

On Properties of Higher-Order Delaunay Graphs with Applications[*]

Manuel Abellanas[†] Prosenjit Bose[‡] Jesús García[§] Ferran Hurtado[¶] Mariano Nicolás[||]
Pedro A. Ramos[**]

Abstract

In this work we study the order-k Delaunay graph, which is formed by edges pq having a circle through p and q and containing no more than k sites. We study the combinatorial structure of the set of triangulations that can be constructed with edges of this graph and show that it is connected under the flip operation if $k \leq 1$ and for every k if points are in convex position. We also study the hamiltonicity of the order-k Delaunay graph and give an application to a coloring problem.

1 Introduction

The Delaunay graph is an ubiquitous structure in the field of Computational Geometry. It is well known that this graph is a triangulation when the points are in general position and that it can be easily completed to a triangulation in the presence of degenerate configurations. An encyclopedic treatment of this structure can be found in the book by Okabe et al. [7].

The edges of a Delaunay triangulation of a planar point set P have a simple geometric definition (i.e. its proximity measure). Two points $p, q \in P$ form a Delaunay edge provided that there exists a circle with p and q on its boundary with no points of $P \setminus \{p, q\}$ in its interior.

This condition can be generalized in a natural way by relaxing the requirement that the circle needs to be empty. In this way, we say that $p, q \in P$ form an edge of the *order-k Delaunay graph* provided that there exists a circle with p and q on its boundary with at most k points of the set $P \setminus \{p, q\}$ inside the circle. Note that the order-0 Delaunay graph is the standard one.

In [3] the authors don't focus on the order-k Delaunay graph yet its edges are defined and called order-k Delaunay edges; then they deal with the problem of computing the set of order-k Delaunay edges which can be completed to a triangulation such that all the triangles have order at most k, where the order of a triangle is defined as the number of points contained inside its circumscribing circle. For the constrained Delaunay triangulation, related problems are considered in [4].

It may be surprising that similar questions have been considered some years ago for graphs related to the Delaunay graph: In [8], properties of the order-k Gabriel Graph (GG) are investigated and an algorithm for its construction is proposed, while in [1] it is shown that the order-20 Relative Neighborhood Graph (RNG) is Hamiltonian.

In this paper, we concentrate mainly on the study of some graph theoretic properties of the order-k Delaunay graph as well on some applications arising from these properties.

2 Order-k Delaunay graph

Throughout this paper, unless explicitly stated otherwise, P will be a set of points in the plane in general position – no three points are collinear and no four are on a circle.

Definition 1 *Given two points $p, q \in P$, the* order *of pq is the smallest integer k such that there exists a circle through p and q containing in its interior k points of P. The order-k Delaunay graph of P, denoted $k - DG(P)$, is formed by the edges with order at most k.*

We start by giving an upper bound on the number of edges of the order-k Delaunay graph which can be derived taking into account its relation with higher order Voronoi diagrams [7].

Theorem 1 *Let P be a set of points in general position and let $|k - DG(P)|$ be the number of edges of the order-k Delaunay graph. Then*

$$|k - DG(P)| \leq 3(k+1)n - 3(k+1)(k+2)$$

If P is in convex position, then

$$|k - DG(P)| \leq 2(k+1)n - \frac{3}{2}(k+1)(k+2)$$

[*]Partially supported by grants NSERC, DURSI 2001SGR00224 and MCYT BFM2003-0368, TIC2003-08933-C02-01 and BMF2002-04402-C02-01

[†]Dep. Matemática Aplicada, Facultad de Informática, Universidad Politécnica de Madrid, Spain. mabellanas@fi.upm.es

[‡]School of Computer Science, Carleton University, Ottawa, Canada. jit@scs.carleton.ca

[§]Dep. Matemática Aplicada, Escuela Univ. de Informática, Universidad Politécnica de Madrid. jglopez@eui.upm.es

[¶]Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya. Ferran.Hurtado@upc.edu

[||]Department of Mathematics, University of Kentucky. cnicolas@ms.uky.edu

[**]Departamento de Matemáticas, Universidad de Alcalá. pedro.ramos@uah.es

Proof. Let b_{pq} be the bisector of points p and q and let $V_k(P)$ be the order-k Voronoi diagram of P. Clearly, if the order of pq is k, an edge of b_{pq} appears for the first time in $V_{k+1}(P)$. In [2], it is shown that the total number of connected components that appear in the set of lines $\{b_{pq} \mid p, q \in P\}$ when all Voronoi diagrams up to order k are put together is

$$\lambda_k = 3kn - \frac{3}{2}k(k+1) - \sum_{j=1}^{k} e_j(P),$$

where $e_j(P)$ is the number of j-sets of P. If P is in convex position, then $\sum_{j=1}^{k} e_j(S) = kn$, while for arbitrary P is known that

$$\sum_{j=1}^{k} e_j(S) \geq 3\binom{k+1}{2}$$

(see [2],[6]). Therefore, the result follows from the fact that $|k - DG(P)| \leq \lambda_{k+1}$. □

3 Flip-graph of order-k triangulations

In this section we study the structure of the set of triangulations that can be constructed using edges of the order-k Delaunay graph. We say that a triangulation T has order k if all its edges have order at most k and there is some edge with order exactly k. We recall that if a triangulation T_1 has two triangles pqr and pqs in convex position, we can get another triangulation T_2 by deleting the edge pq and adding the edge rs. This operation is called a *flip*. In this situation, we say that the edge pq is locally Delaunay if the circle passing through p, q and r does not contain point s.

Definition 2 *The flip-graph of triangulations with order at most k, denoted by $TG_k(P)$, is defined in the following way:*

1. *the vertices are the triangulations of P with order at most k,*
2. *two triangulations T_1 and T_2 are connected with an edge in $TG_k(P)$ if they differ in a flip.*

If $k = 0$, $TG_0(P)$ is a single vertex (the Delaunay triangulation) and thus connected. In the following theorem we answer the question of the connectedness of these graphs.

Theorem 2

a) *$TG_1(P)$ is connected.*

b) *$TG_k(P)$ can be disconnected if $k \geq 2$.*

c) *If P is in convex position, then $TG_k(P)$ is connected for every $k \geq 0$.*

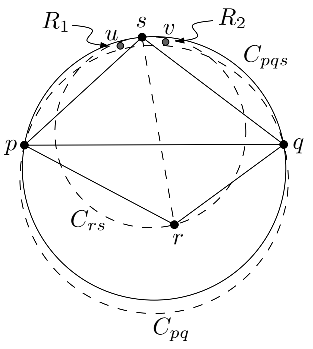


Figure 1: Illustration for the proof of Theorem 2

Proof. Let T be a triangulation with order one and let DT be the Delaunay triangulation of P. We are going to show that if $T \neq DT$ there exists an edge of T which is not locally Delaunay and can be flipped to an edge with order at most one.

Let pq be an edge which is not locally Delaunay (then, it has order one) and let rs be the edge that we get when pq is flipped (see Figure 1). If rs has order at most one then we have done, so assume that rs has order at least two. Because pq is not locally Delaunay and has order one, there exists a circle C_{pq} passing through p and q and containing a single point, which is necessarily either r or s. In the following, we assume that C_{pq} contains r and, therefore, the edges pr and qr are Delaunay edges.

Let C_{pqs} be the circle passing through p, q and s and C_{rs} the circle tangent to C_{pqs} at s and passing through r. Let R_1 and R_2 be the regions inside C_{rs} and outside both of the circle C_{pq} and the quadrilateral $prqs$. It is easy to see that each of the regions contains exactly one point, as illustrated in Figure 1).

Let us denote by u and v, respectively, the points inside the regions R_1 and R_2, and by C_{prs} the circle through p, r and s. The circle C_{prs} contains at least two points and no point different from u and v can be inside it. This shows that the edge pv has order at most one. In an analogous way, it can be seen that the edge qu has order at most one. If the edge ps is not locally Delaunay then we have finished because the triangle psu is in T and we can flip the edge ps to qu so we can assume that ps is locally Delaunay.

If triangle psu is not in T, then we can consider the set of triangles C intersected by segment qu and show that if $p'q's'$ and $p'us'$ are adjacent triangles in C then the edge $p's'$ is not locally Delaunay while the edge $q'u$ has order zero and this concludes the proof of part a).

In Figure 2 we show an example of a triangulation with order two such that every possible flip increases its order to three. Therefore, $TG_2(P)$ is not connected.

The proof of part c) is omitted in this extended abstract due to space limitations. □

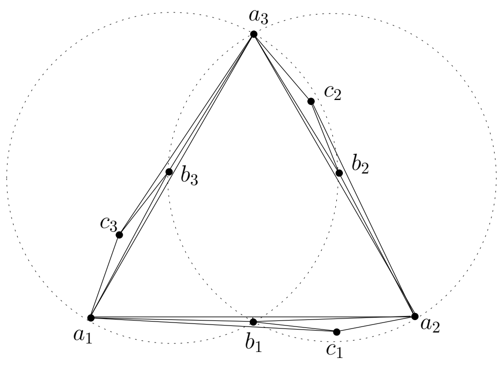


Figure 2: An isolated triangulation in $TG_2(P)$

4 Hamiltonicity of Order-k Delaunay Graph

In this section, we show that the order-15 Gabriel Graph (GG) contains a Hamiltonian cycle. Note that 15-GG is a subgraph of the 15-DG. The key idea behind the proof is the following. Given a particular Hamiltonian cycle h through a set of n points, define the distance sequence, $ds(h) = \delta_1, \ldots, \delta_n$ to be the sequence of edge lengths in the cycle sorted from longest to shortest edge. Given any two Hamiltonian cycles x and y, we can compare lexicographically their edge length sequences. In the following theorem we prove that a cycle associated with an edge length sequence which is minimum with that order has the property that every edge belongs to 15-GG.

Theorem 3 *Given a set P of n points in the plane in general position, the graph 15-GG contains a Hamiltonian cycle (and hence 15-DG too).*

Proof. Let H be the set of all Hamiltonian cycles through the points of P. Let $m = a_0, a_1, \ldots, a_{n-1}$ be a cycle in H with minimal distance sequence. We will show that all of the edges of m are in 15-GG. We proceed by contradiction.

Suppose that there are some edges in m that are not in 15-GG. Let $e = [a_i a_{i+1}]$ be the longest edge that is not in 15-GG (all index manipulation is modulo n). Let B be the circle with a_i and a_{i+1} as diameter.

Claim 1: No edge of m can be completely inside B. Suppose there was an edge $f = [a_j, a_{j+1}]$ inside B. By deleting e and f from m and adding either $[a_i, a_j], [a_{i+1}, a_{j+1}]$ or $[a_i, a_{j+1}], [a_{i+1}, a_j]$, we construct a new cycle m' whose distance sequence is strictly smaller than that of m since $d(a_i, a_{i+1}) > max\{d(a_i, a_j), d(a_{i+1}, a_{j+1}), d(a_i, a_{j+1}), d(a_{i+1}, a_j)\}$. But this is a contradiction since m is a minimal distance sequence.

Therefore, we may assume that no edge of m lies completely inside B. Since e is not 15-GG there must be at least $w \geq 16$ points of P in B. Let $U = u_1, u_2, \ldots, u_w$ represent these points indexed in the order we would encounter them on the cycle starting from a_i. Let $S = s_1, s_2, \ldots, s_w$ and $T = t_1, t_2, \ldots, t_w$ represent the vertices where s_i is the vertex preceding u_i on the cycle and t_i is the vertex succeeding u_i on the cycle.

Let D be the circle centered at a_{i+1} with radius $2r$.

Claim 2: No point of T can be inside D. Suppose $t_j \in T$ is in D, then $d(t_j, a_{i+1}) < 2r$. Construct a new cycle m' by removing the edges $[u_j, t_j], [a_i, a_{i+1}]$ and adding the edges $[a_{i+1}, t_j], [a_i, u_j]$. Since the two edges added have length strictly less than $2r$, $ds(m') < ds(m)$ which is a contradiction.

Let c be the midpoint of the edge $[a_i, a_{i+1}]$. Let C be the circle centered at c with radius $2r$ and

Claim 3: There are at most 4 points of T in C. Suppose that there are 5 points of T in C. Note that the 5 points are in $C \cap \overline{D}$ by the previous claim. However, this means that there must be two points t_j, t_k such that $\angle(t_j, c, t_k) < \pi/3$. But this implies that $|\overline{t_j t_k}| < 2r$.

Since $|T| \geq 15$, there are at least 11 points of T outside C. Decompose the plane into 10 cones of angle $\pi/5$ centered at c. By the pigeon-hole principle, there must be one cone with at least 2 points, t_j and t_k. We note that $d(t_j, t_k)$ is either less than $2r$ or less than $\max d(c, t_j) - r, d(c, t_k) - r$ (a proof of this fact can be found in the technical report). Construct a new cycle m' from m by first deleting $[t_j, u_j], [t_k, u_k], [a_i, a_{i+1}]$. This results in three paths. One of the paths must contain both a_i and either t_j or t_k. WLOG, suppose that a_i and t_j are on the same path. Add the edges $[a_i, u_k], [a_{i+1}, u_j], [t_j, t_k]$. The resulting cycle m' has a strictly smaller distance sequence since $\max [t_j, u_j], [t_k, u_k], [a_i, a_{i+1}] > \max [a_i, u_k], [a_{i+1}, u_j], [t_j, t_k]$.

□

5 Coloring with Applications

Given a set of n points in the plane, Har-Peled and Smorodinsky [5] showed how to assign one of m colors to each of the n points such that every circle C containing more than one point has at least one point in C with a unique color. Such a coloring is called a *conflict-free* coloring (CF-coloring for short). The Delaunay graph is used both in the coloring algorithm and to show that m is $O(\log n)$. This type of coloring finds application in the assignment of frequencies in a cellular network.

In this section, we generalize the result in [5]. We show that with $O(\log n/\log(8ck/(8ck-1))$ colors, a set of n points in the plane can be colored so that every circle containing at least k points contains at least k points with unique color (where the maximum number of edges in $(k-1)$-DG is ckn for some constant c). We call such a coloring a k-conflict-free coloring. In the context of cellular networks, this can be viewed as ensuring that for every client in range of k or more towers, there always exists at least k different towers with which the client can communicate without interference.

As noted in Theorem 1, the number of edges in $(k-1)$-DG is at most ckn where $c = 3$ when the points are in general position and $c = 2$ when points are in convex position. This implies that the average degree of a vertex in $(k-1)$-DG is at most $2ck$ and, by using a standard argument which is omitted in this extended abstract, it can be seen that there are always *big* independent sets with bounded degree:

Lemma 4 *Every $(k-1)$-DG has an independent set of size at least $n/8ck$ where each vertex in the set has degree at most $4ck$.*

The coloring algorithm is simple and repeated applies the above lemma. Find a large independent set in the $(k-1)$-DG of the given point set P. Assign a unique color to the points in the independent set. Remove these points from P and repeat as long as $|P| > 0$. In the next lemma, we show that this algorithm provides a k-conflict free coloring and the total number of colors used is $\log n/\log(8ck/(8ck-1))$

Lemma 5 *With $\log n/\log(8ck/(8ck-1))$ colors, a set of n points can be colored so that every circle containing at least k points contains k points whose color is unique.*

Proof. First, at each iteration, we remove an independent set of size at least $n/8ck$. Let $C(n)$ represent the number of colors used for a $(k-1)$-DG graph with n vertices. We can bound $C(n)$ with the following recurrence: $C(n) \leq C((8ck-1)n/8ck) + 1$. This recurrence resolves to $C(n) \leq \log n/\log(8ck/(8ck-1))$ as required.

Next, we show that the coloring is k-conflict free. Let C be any circle containing a set P of at least k points. Consider the k points in C whose colors have highest value (recall that the first independent set was given color 0 and an independent set removed at step i was given color i). If all these k points have unique colors, the lemma is proved. For sake of a contradiction, assume that at least 2 of these k points have the same color. Let i be the largest color whose value is not unique. Note that there are fewer than k points in P whose color value is strictly greater than i. Also note that at iteration i of the algorithm, all points with color less than i have been removed from P. Let P_i be the set of points in P receiving color i. Since C contains P_i, there is a circle C' contained in C that has two points x, y of P_i on its boundary and no points of P_i in its interior. However, since there are fewer than k points whose color is larger than i, this means that C' contains fewer than k points in its interior at iteration i of the algorithm. However, this contradicts the fact that x and y are in an independent set selected at iteration i. $\square$

Corollary 6 *A set of n points in general position can be colored with $\log n/\log(24k/(24k-1))$ colors so that every circle containing at least k points contains k points whose color is unique. If the set of n points is in convex position, then $\log n/(\log(16k/(16k-1))$ colors are sufficient*

Note that we only used the fact that there are large numbers of vertices of bounded degree in $(k-1)$-DG in order to show that there is a sufficiently large independent set. If one can find a larger independent set that is guaranteed to exist in all $(k-1)$-DG graphs, then the above bounds can be improved.

6 Conclusion

In this work we have investigated some properties of higher order Delaunay graphs. There are several questions that remain open, and we emphasize the following:

- give some lower bound on the size of k-DG and a tight upper bound,
- show that k-DG is Hamiltonian for small k.

References

[1] M. S. Chang, C. Y. Tang, and R. C. T. Lee. 20-relative neighborhood graphs are Hamiltonian. In *Proc Internat. Sympos. on Algorithms*, LNCS-450, pp 53–65. Springer, 1990.

[2] H. Edelsbrunner, N. Hasan, R. Seidel, and X. J. Shen. Circles through two points that always enclose many points. *Geom. Dedicata*, 32:1–12, 1989.

[3] J. Gudmundsson, M. Hammar, and M. van Kreveld. Higher order Delaunay triangulations. *Comput. Geom. Theory Appl.*, 23:85–98, 2002.

[4] J. Gudmundsson, H. Haverkort, and M. van Kreveld. Constrained higher order Delaunay triangulations. In *Abstracts 19th European Workshop Comput. Geom.*, pages 105–108, 2003.

[5] S. Har-Peled and S. Smorodinsky. Combinatorial geometry: On conflict-free of points and simple regions in the plane. In *Proc. 19th Annu. ACM Sympos. Comput. Geom.*, pages 114–123, 2003.

[6] L. Lovász, K. Vesztergombi, U. Wagner, and E. Welzl. Convex quadrilaterals and k-sets. *Contemporary Mathematics*, 342:139–148, 2004.

[7] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams, 2nd Ed.*. J. Wiley and Sons, 2000.

[8] T. H. Su and R. C. Chang. The k-Gabriel graphs and their applications. In *Proc. Internat. Sympos. on Algorithms*, LNCS-450, pp 66–75. Springer, 1990.

Constructing Higher-Order Voronoi Diagrams in Parallel

Henning Meyerhenke*

Abstract

We use Lee's sequential algorithm [17] to create the first parallel algorithm which constructs the order-k Voronoi diagram of a planar point set. The algorithm is developed and analyzed within two parallel models, the fine-grained PRAM and the coarse-grained CGM. Its applications include k-nearest neighbor searches in a parallel context, which are important for many applications in computational geometry. The fine-grained algorithm requires $O(k\, log^2\, n)$ time and $O(k^2 n\, log\, n)$ work on a CREW-PRAM, whereas the coarse-grained version requires the ordinary Voronoi diagram as input and then takes $O(\frac{k^2(n-k)\, log\, n}{p})$ running time and $O(k)$ communication rounds on a CGM with $O(\frac{k^2(n-k)}{p})$ local memory per processor.

1 Introduction

The Voronoi diagram is one of the most popular geometric structures studied in the field of computational geometry due to its rich algorithmic application in placement and motion planning, triangulations, connectivity graphs and closest-site problems [5]. A generalization of ordinary Voronoi diagrams are those of order k, whose regions are the locus of points with the same k nearest neighbors. Given an order-k Voronoi diagram, one can find the k-nearest point site of a query point efficiently, which is very important for generalized closest-site problems and for the closely related higher order Delaunay triangulations [15].

Since these applications from geometry and geographic information systems often benefit from parallel computing, we develop a simple parallel algorithm to construct all higher order Voronoi diagrams of some given order k and below. After the description of the sequential algorithm the parallel algorithm is developed and analyzed both in the fine-grained PRAM model (section 2) and in the coarse-grained CGM model (section 3), which is more relevant in practice.

In an ordinary Voronoi diagram of a planar point set S of size n each Voronoi region is induced by exactly one point site of S, since it is the locus of points with the same nearest neighbor in S. The concept of order k extends this perception by making each Voronoi region the locus of points which have the same k nearest neighbors (thus, an ordinary Voronoi diagram has order one). An example of the order-2 diagram of a planar point set is depicted in figure 1.

Previous works on sequential algorithms for constructing higher order Voronoi diagrams include the books of Preparata and Shamos [18] and Edelsbrunner [13]. The latter points out the duality between k-levels in arrangements and higher order Voronoi diagrams, an idea which is used in some of the sequential algorithms developed for the posed problem by Agarwal et al. [1], Aurenhammer [4], Aurenhammer and Schwarzkopf [6], Boissonnat et al. [7], Chan [8], Chazelle and Edelsbrunner [9], Ramos [19], and Lee [17].

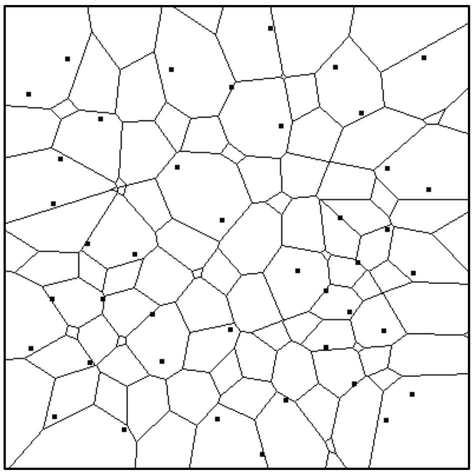


Figure 1: Voronoi diagram of order 2 of a planar point set, generated with an applet written by B. Schaudt [20].

D. T. Lee [17] describes a sequential algorithm for computing the family of all these diagrams of order $\leq k$ in $O(k^2 n\, log\, n)$ time. His method is iterative as it modifies $V_i(S)$, the Voronoi diagram of order i, in order to obtain the diagram of order $i+1$. Its space complexity for each iteration is $O(i^2(n-i))$, since the order-i diagram has $O(i(n-i))$ regions and for each of which i inducing points are stored. Meanwhile, the algorithm's time complexity has been improved to $O(n\, log\, n + nk^2)$ by Aggarwal et al. [2], whose algorithm can delete one point site from a Voronoi diagram and recompute the resulting one in linear time.

The iterative approach of Lee's algorithm (transforming the diagram of order i into the diagram of

*Department of Computer Science, University of Paderborn, henningm@upb.de order $i+1$) is mainly based on the following idea: Each Voronoi region r of order i, which is induced by some $H \subset S$ with $|H| = i$, is partitioned into subregions which are the respective locus of points with the same $i+1$ neighbors in S. Since all points of r share the same i nearest neighbors (namely the points in H), what remains to be done is to partition r into subregions according to their nearest neighbor in $S \backslash H$. This partitioning procedure works by merging $V_1(S \backslash H)$, the ordinary Voronoi diagram of the remaining $n-i+1$ points, appropriately with r (cf. Lee [17] for details).

2 The PRAM algorithm

As Lee has shown [17, p. 482], two adjacent regions of a Voronoi diagram of order i share $i-1$ of their i nearest neighbors. When partitioning such a region r according to $V_1(S \backslash H)$, it is split into regions w.r.t. to the $(i+1)$-nearest neighbors of r (i.e. the (possibly) different $(i+1)$-nearest neighbors of different points in r). The key observation is that only few additional information is required to do this.

Lemma 1 *The $i+1$-nearest neighbors of a region r of $V_i(S)$ all induce adjacent regions of r.*

Proof. Points on Voronoi edges of order i share the same $i+1$ nearest neighbors because they have not only one, but two i-nearest neighbors. If r is induced by the point set H, then its edges are induced by H and one additional point each. This point induces the adjacent region whose boundary the respective edge is part of. $\square$

Corollary 2 *In order to partition a Voronoi region r of order i into subregions of order $i+1$, it is sufficient to know r's neighboring regions of order i (and their inducing points).*

This locality property is obviously useful for a parallel version of Lee's algorithm. Following its scheme, the problem mainly consists of partitioning each order-i region into subregions of order $i+1$ and merging equivalent subregions. To do this for some region r, one only needs to know the points which induce the regions adjacent to r.

For the PRAM algorithm we therefore compute for each order-i region r that part of $V_1(S \backslash H)$ (the ordinary Voronoi diagram of S without the points that induce r) which contributes to r. This can be done using the algorithm developed by Amato et al. [3], which requires $O(log^2 n)$ time and $O(n \, log \, n)$ work on a EREW-PRAM to compute the Voronoi diagram of n sites in the plane. After that, only some minor update operations have to be performed to complete one iteration and obtain the order-$(i+1)$ diagram.

More formally we state the algorithm as follows:

```
Input: Planar point set $S^n$, $p \le n$ processors, k
Output: Order-$k$ Voronoi diagram of $S$

1. Use Amato et al.'s algorithm [3] to compute the
   Voronoi diagram of $S$.
2. FOR $i := 1$ TO $k-1$ DO
   /* Transform the diagram of order $i$ into one of
   order $i+1$ */
   (a) FOR EACH region $r_j$ PARDO
       i. Partition $r_j$ induced by $H_j \subset S$
          into subregions according to $V_1(S \backslash H_j)$,
          which is computed by Amato et al.'s al-
          gorithm.
   (b) FOR EACH new region $r_m$ not updated yet
       PARDO
       i. Update $r_m$ by merging it with equiva-
          lent subregions and deleting old edges
          within the merged region.
```

Theorem 3 *The family of all order-($\le k$) Voronoi diagrams can be computed on a CREW-PRAM using $O(k \, log^2 n)$ time and $O(k^2 n \, log \, n)$ work.*

Proof. Recall that an order-i Voronoi diagram has $O(i(n-i))$ regions. If an r_j of them has s_j edges, we need to compute the ordinary Voronoi diagram of s_j points. The total amount of work per outer iteration is therefore $\sum_{j=1}^{k(n-k)} s_j \, log \, s_j = kn \, log \, n$. (The parallel merging/deleting loop requires only constant time and a linear amount of work.) Since one region can have at most $O(k(n-k))$ edges, the algorithm takes $O(log^2(k(n-k)) = O(log^2 n)$ time for each of the $k-1$ iterations. For this method, no concurrent write operations are necessary, but concurrent read operations are. $\square$

Unfortunately, the work complexity does not match the improved sequential running time. For this, a parallel algorithm comparable to Aggarwal et al.'s sequential one [2] that deletes one site from an existing Voronoi diagram and can recompute the new one using a linear amount of work, would be necessary.

3 The CGM algorithm

We use now the locality property explained in the previous section to develop a coarse-grained parallel algorithm for the problem within the (slightly modified) CGM model [11]. The original CGM model consists of p processors connected by some arbitrary network, with each processor having a local memory of size $O(\frac{n}{p})$. Interprocessor communication is realized by message passing in communication rounds, during which no processor can send or receive more data items than fit into its local memory.

It was shown by Dehne et al. [11] that many collective communication primitives can be simulated on a CGM by a constant number of global sort operations. Moreover, Goodrich proved [14] that coarse-grained sorting can be performed optimally within a constant number of communication rounds if $\frac{n}{p} \geq p$.

For our algorithm we slightly modify the model by increasing the local memory bound to $O(\frac{k^2(n-k)}{p})$, which is the space complexity of the sequential algorithm divided by the number of processors.

The ordinary Voronoi diagram of the point set S is part of the input of our algorithm.[1] This diagram of order 1 must have the property to be *x-disjoint*, i.e. it is stored in such a way that S is distributed on the processors within disjoint intervals w.r.t. the x-coordinate. This enables a processor to determine efficiently which other processor stores a particular point site. (If the diagram is not x-disjoint, it can be made so via a constant number of global sort operations.)

The algorithm works as follows:

Input: $CGM(n,p)$ with $\frac{n}{p} \geq p$ and $O(\frac{k^2(n-k)}{p})$ local memory per processor; planar point set S of size n and its Voronoi diagram, which is distributed on the p processors in such a way that each processor stores $O(\frac{n}{p})$ Voronoi regions and edges; order k of the desired output.

Output: $V_k(S)$ distributed on the p processors such that every processor stores $O(\frac{k(n-k)}{p})$ Voronoi edges and regions.

1. If $V_1(S)$ is not x-disjoint, then it is rearranged by a constant number of global sorting steps to make it x-disjoint.

2. Every processor sends the locally stored point with the largest x-coordinate to all the other processors.

   With this information every processor determines by binary search for each *boundary region* (i.e. a local Voronoi region with at least one non-local neighbor region), which other processor stores its *non-local neighbor regions* (i.e. regions stored on other processors that share an edge with a locally stored region).

3. FOR $i := 1$ TO $k-1$ DO

   (a) Each processor sends necessary information about its boundary regions of order i to the processors which store their non-local neighbor regions (if present).

   (b) Using Lee's algorithm [17, p. 482] with the enhancement by Aggarwal et al. [2], each processor transforms its local part of $V_i(S)$ into the corresponding order-i Voronoi diagram.

   (c) Since some of the new Voronoi regions are computed on two processors but are to be used on only one of them, we proceed as follows: If i is even, a newly created duplicate region is only retained on the processor with the lower number, otherwise only on the processor with the higher number.

[1]The best deterministic CGM algorithm for computing the Voronoi diagram of arbitrary planar point sets is due to Diallo et al. [12] and requires $O(\frac{n\,log\,n\,log\,p}{p})$ local computation steps and $O(log\,p)$ communication rounds. Moreover, two rather complicated randomized CGM algorithms exist, which both compute the planar Voronoi diagram with high probability in optimal time of $O(\frac{n\,log\,n}{p})$ and with $O(1)$ communication rounds [10, 16].

Lemma 4 *A processor creates at most $O(\frac{i(n-i)}{p})$ Voronoi regions of order $i+1$ during the transition from $V_i(S)$ to $V_{i+1}(S)$.*

Proof. An ordinary Voronoi diagram can be distributed on p processors such that each processor stores $O(\frac{n}{p})$ Voronoi regions and edges so that the claim is true for $i = 1$. Let us assume now inductively that it holds for arbitrary $i_0 \leq i$.

According to our assumption, a processor stores $O(\frac{i(n-i)}{p})$ Voronoi regions and edges of an order-i Voronoi diagram. These local regions can be seen as an independent sub-diagram, which can be influenced by at most $O(\frac{i(n-i)}{p})$ non-local Voronoi regions, since the local regions cannot have more neighbor regions than edges. The total number of all these regions is still $O(\frac{i(n-i)}{p})$, so that the order-$(i+1)$ sub-diagram which is created during the transition, has at most $O(\frac{(i+1)(n-(i+1))}{p}) = O(\frac{i(n-i)}{p})$ regions. □

Theorem 5 *Let a coarse-grained multicomputer $CGM(n,p)$ with $\frac{n}{p} \geq p$ and local memory size $O(k^2(n-k))$ be given and let the Voronoi diagram of a planar point set S be distributed on $CGM(n,p)$ such that each processor stores $O(\frac{n}{p})$ Voronoi regions and edges. Then one can compute the order-k Voronoi diagram of S with a time complexity of $O(\frac{k^2(n-k)\,log\,n}{p})$ for local computation using $O(k)$ supersteps and communication rounds, during which every processor sends and receives at most $O(k^2(n-k))$ data items.*

Proof. Using corollary 2, we conclude that each processor can transform its locally stored Voronoi sub-diagram of order i into the sub-diagram of order $i+1$ if it knows all neighbors of its locally stored regions. This is ensured by the communication in step 3a. Since all duplicate regions are eliminated in step 3c, the union of all sub-diagrams forms the Voronoi diagram of order $i+1$ after iteration i.

The communication complexity of the algorithm is clearly dominated by step 3a, during which at most

$O(\frac{i(n-i)}{p})$ Voronoi regions are communicated. (Note that for each region the number of inducing points sent/received is not greater than the region's number of edges.) This can be accomplished by a total exchange operation. Hence, no more than $O(\frac{i(n-i)}{p})$ points are sent and received during one communication round.

Furthermore, the local time complexity is dominated by this communication step, too. It requires in total $\sum_{i=1}^{k-1} O(\frac{i(n-i)\,log\,n}{p}) = O(\frac{k^2(n-k)\,log\,n}{p})$ local computation. In contrast to this, the other steps require only $O(\frac{n\,log\,n}{p})$ (step 1), $O(\frac{n\,log\,p}{p})$ (step 2), and $\sum_{i=1}^{k-1} O(\frac{i(n-i)}{p}) = O(\frac{k^2(n-k)}{p})$ (both step 3b and step 3c), respectively.

Clearly, the local memory space bound of $O(\frac{k^2(n-k)}{p})$ is necessary, but never exceeded. $\square$

4 Conclusion

In this paper we present and analyze a parallel algorithm for constructing higher order Voronoi diagrams for two parallel models, PRAM and CGM. For both models we use Lee's iterative idea [17] to transform the diagram of order i into one of order $i+1$. Although the algorithm is quite simple, it appears to be the first parallel one for the construction of higher order Voronoi diagrams.

Two possible improvements could be the object of further research: First, a parallel algorithm which deletes a point site from a Voronoi diagram and can recompute the new one using a linear amount of work. This would make our PRAM algorithm work-optimal when compared to Lee's sequential algorithm.

Secondly, the CGM algorithm is only efficient for small k. Although small k seem to be more relevant in practice, it would nevertheless be of interest to obtain a CGM algorithm which requires significantly less than $O(k)$ communication rounds.

References

[1] P. K. Agarwal, M. de Berg, J. Matoušek, O. Schwarzkopf. Constructing Levels in Arrangements and Higher Order Voronoi Diagrams. In *Proc. 10th Symp. on Comp. Geometry* (1994), pp. 67–75.

[2] A. Aggarwal, L. J. Guibas, J. Saxe, P. W. Shor. A Linear-Time Algorithm for Computing the Voronoi Diagram of a Convex Polygon. In *Discrete & Comput. Geometry* **4** (1989), pp. 591–604.

[3] N. M. Amato, M. T. Goodrich, E. A. Ramos. Parallel algorithms for higher-dimensional convex hulls. In *Proc. 35th Annual IEEE Symp. on Foundations of Comp. Science* (1994), pp. 683–694.

[4] F. Aurenhammer. A New Duality Result Concerning Voronoi Diagrams. In *Discrete & Comput. Geometry* **5** (1990), pp. 243–254.

[5] F. Aurenhammer. Voronoi Diagrams - A Survey of a Fundamental Geometric Data Structure. In *ACM Comput. Surv.* **23** (1991), no. 3, pp. 345–405.

[6] F. Aurenhammer, O. Schwarzkopf. A Simple On-Line Randomized Incremental Algorithm for Computing Higher Order Voronoi Diagrams. In *Proc. 7th Symp. on Computational Geometry* (1991), pp. 142–151.

[7] J.-D. Boissonnat, O. Devillers, M. Teillaud. A Semi-Dynamic Construction of Higher Order Voronoi Diagrams and its Randomized Analysis. In *Algorithmica* **9** (1993), pp. 329–356.

[8] T. M. Chan. Random Sampling, Halfspace Range Reporting, and Construction of ($\leq k$)-Levels in Three Dimensions. In *SIAM J. Comput.* **30** (2000), no. 2, pp. 561–575.

[9] B. Chazelle, H. Edelsbrunner. An Improved Algorithm for Constructing kth-Order Voronoi Diagrams. In *IEEE Transactions on Computers* **C-36** (1987), no. 11, pp. 1349–1354.

[10] F. Dehne, X. Deng, P. Dymond, A. Fabri, A. A. Khokhar. A Randomized Parallel Three-Dimensional Convex Hull Algorithm for Coarse-Grained Multicomputers. In *Theory of Computing Systems* **30** (1997), no. 6, pp. 547–558.

[11] F. Dehne, A. Fabri, A. Rau-Chaplin. Scalable parallel computational geometry for coarse grained multicomputers. In *Int. J. Comput. Geometry* **6** (1996), no. 3, pp. 379–400.

[12] M. Diallo, A. Ferreira, A. Rau-Chaplin. A Note On Communication-Efficient Deterministic Parallel Algorithms for Planar Point Location and 2d Voronoi Diagram. In *Parallel Processing Letters* **11** (2001), no. 2-3, pp. 327–340.

[13] H. Edelsbrunner. Algorithms in Combinatorial Geometry. Springer-Verlag, 1987.

[14] M. T. Goodrich. Communication-Efficient Parallel Sorting. In *SIAM J. on Computing* **29** (1999), no. 2, pp. 416–432.

[15] J. Gudmundsson, M. Hammar, M. J. van Kreveld. Higher Order Delaunay Triangulations. In *Comp. Geometry - Theory and Applications* **23** (2002), no. 1, pp. 85–98.

[16] U. Kühn. Lokale Eigenschaften in der algorithmischen Geometrie mit Anwendungen in der Parallelverarbeitung. Dissertation Westfälische Wilhelms-Universität Münster, 1998.

[17] D. T. Lee. On k-nearest neighbor Voronoi diagrams in the plane. In *IEEE Transactions on Computers* **31** (1982), no. 6, pp. 478–487.

[18] F. P. Preparata, M. I. Shamos. Computational Geometry - An Introduction. Springer-Verlag New York, 1985.

[19] E. A. Ramos. On range reporting, ray shooting and k-level construction. In *Proc. 15th Symp. on Computational Geometry* (1999), pp. 390–399.

[20] B. Schaudt. Higher Order Voronoi Diagrams: A Java Applet. Website available at www.msi.umn.edu/~schaudt/voronoi/voronoi.html.

Delineating Boundaries for Imprecise Regions*

Iris Reinbacher† Marc Benkert‡ Marc van Kreveld† Alexander Wolff‡

Abstract

In geographic information retrieval users use names of geographic regions that do not have a well-defined boundary, like Southern France. We present two approaches to compute reasonable boundaries of such regions, based on evidence of points that are likely to lie inside or outside this region.

1 Introduction

Geographic information retrieval is concerned with information retrieval for spatially related data, including Web searching. Certain specialized search engines allow queries that ask for things (hotels, museums) in the (geographic) neighborhood of some named location. Another typical query could specify locations such as Central Mexico or the British Midlands. The latter is an example of a named region for which no exact boundaries exist. The extent of such a region is in a sense in the minds of the people.

Since geographic queries may ask for Web pages on castles in the British Midlands, it is still useful to have a reasonable boundary for this imprecise region. We would then be able to find Web pages for locations in the British Midlands that mention castles, even if they do not contain the words British Midlands. We would need to store a reasonable boundary for the British Midlands in the ontology (that stores all geographic information, including coordinates and geographic concepts) during preprocessing, and use query time for searching in the spatial part of the combined spatial and term index.

To determine a reasonable boundary for an imprecise region we can use the Web once again. The enormous amount of text on all Web pages can be used as a source of data; the idea of using the Web as a geo-spatial database appeared before in [9, 11]. A possible approach is using so-called *trigger phrases*. For any reasonable size city in the British Midlands, like Nottingham, it is quite likely that some Web page contains a sentence fragment like "…Nottingham, a city in the British Midlands, …", or "Nottingham is located in the British Midlands…". Such fragments give a location that is most likely in the British Midlands, while other cities like London or Cardiff, that do not appear in similar sentence fragments, give locations that are not in the British Midlands. Details of using trigger phrases to determine locations inside or outside a region to be delineated can be found in [2]. Obviously, the process is not too reliable and false positives and false negatives are likely to occur. In the reliable case, Alani et al. [1] give a Voronoi Diagram based method to delineate regions.

We have arrived at the following computational problem: given a set of "inside" points (red) and a set of "outside" points (blue), determine a reasonable polygon that separates the two sets well. Possible criteria for a reasonable polygon are that it is simply-connected and has small perimeter (but as it must still contain most red points, it is fat and almost convex).

In computational geometry, red-blue separation algorithms exist of various sorts. Red-blue separation by a line is simply linear programming and takes $O(n)$ time for n points. Red-blue separation by a line with the minimum number of misclassified points takes $O((n + k^2) \log k)$ expected time, where k is the number of misclassified points [4]. Other fixed separation shapes like strips, wedges, and sectors can also be considered [3]. When polygons are the separator, then the most natural problem is the minimum perimeter polygon that separates the bichromatic point set. Euclidean travelling salesperson can be reduced to this problem, which makes it intractable [5]. Gudmundsson and Levcopoulos [7] give an $O(n \log n)$-time algorithm that finds a polygon whose perimeter is at most $O(\log n)$ times as long as the minimum perimeter one. Separation by minimum link shapes received attention as well (e.g., [10]).

In this paper we present two approaches to determine a reasonable polygon for a set of red and blue points. The first approach, described in Section 2, starts with a red polygon with blue points inside, and we try to change the shape of the polygon to get more blue points outside. The second approach, which changes the color of points to obtain a better shape of the polygon is presented in Section 3.

2 Adaptation Method

Let R be the set of red points, B the set of blue points, and let n be the total number of points. In the adap-

*This research is supported by the EU-IST Project No. IST-2001-35047 (SPIRIT) and by the grant WO 758/4-1 of the German Science Foundation (DFG).

†Institute of Information and Computing Sciences, Utrecht University, { iris, marc }@cs.uu.nl

‡Fakultät für Informatik, Universität Karlsruhe, http://i11www.ira.uka.de/people.

tation method, we start with some simply-connected polygon P and adapt it until all blue points inside P are no longer inside, or the shape has to be changed too dramatically.

For an appropriate value of α, we choose our initial polygon P to be the largest simply-connected component of the α-hull of the red points. This way, we can determine a suitable initial shape and possibly remove red outliers (the red points outside P) in the same step. Once we have computed P, the problem that remains is changing P so that the blue points are no longer inside. The resulting polygon should be contained in P and its perimeter should be as small as possible. In this section we discuss the problem of making sure that no blue point remains in the interior, although in practice it may be better for the final shape to allow some blue points to stay inside. They would be considered misclassified.

2.1 One blue point inside P

First, we assume that there is only one blue point b inside the polygon P. We want to determine a polygon P' so that b is not inside, P' is contained in P, all vertices of P are not outside P', and the perimeter of P' is minimal. It is clear that P' cannot have edges that intersect the exterior of P. We consider two cases: the special case where only point b is inside P, and the more general case where P contains b and a number of red points.

Lemma 1 *The optimal polygon P' is a possibly degenerate simple polygon (i.e. vertices and edges may be repeated) (i) with b on the boundary, and (ii) which includes all edges of P, with the exception of one edge.*

2.1.1 One blue and no red points inside P

Let P be a red polygon with only one blue point b inside. Let $e = \overline{p_1p_2}$ be the edge of P that does not appear in P'. The endpoints p_1 and p_2 are connected by a path π via b in P'. We denote the path that leads to the smallest perimeter of P' by $\pi_{\min}$; it consists of a shortest geodesic path between b and p_1, and between b and p_2. The optimal polygon P' has the same boundary as P, except that edge e is replaced by $\pi_{\min}$. In the optimal solution P', the edge e and the shortest path $\pi_{\min}$ have the following properties:

Lemma 2 1. *The path $\pi_{\min}$ is a simple path.*

2. *A funnel π with root b and base e can only be minimal if e is partially visible from b.*

For every two adjacent vertices p_i and p_{i+1} of the polygon, we compute the shortest paths connecting them to b. The algorithm of Guibas et al. [8] can find them in $O(n)$ time. For each possible base $e = \overline{p_ip_{i+1}}$ and corresponding funnel, we add the length of the two paths and subtract the length of the edge e to get the additional length of this choice.

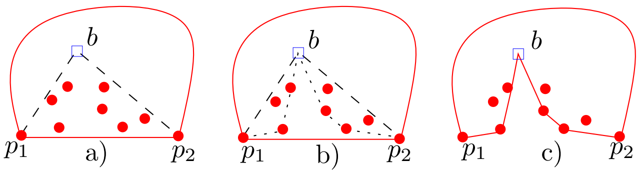


Figure 1: Removing one blue point from the red α-shape. a) The old polygon b) Computing the shortest path c) The new polygon

Theorem 3 *For a simple polygon P with n vertices and with a single point b inside, we can compute the shortest perimeter polygon P' that is contained in P, that contains all vertices of P, and that does not have b in its interior, in $O(n)$ time.*

2.1.2 One blue and several red points inside P

In the general case we may also have red points inside P. Let R be the set of these red points, and assume that its size is $O(n)$. We need to adapt the algorithm given before to take these red points into account. We at first ignore all red points. We again compute all funnels from b to every edge e of P. We get a partitioning of P into $O(n)$ funnels with disjoint interiors. In every funnel F we do the following: If there are no red points inside F, we store the length of the funnel without its base edge e. Otherwise, we need to find a shortest path $\pi_{\min}$ from one endpoint of e to b and back to the other endpoint of e, such that all red points in R still lie inside the resulting polygon P'.

The shortest path $\pi_{\min}$ inside some funnel F with respect to a set $R \cap F$ of red points consists of two chains which, together with the base edge e, again forms a funnel F'. This funnel is not allowed to contain points of $R \cap F$. We need to consider all possible ways of making such funnels. Using dynamic convex hulls, we can obtain:

Theorem 4 *For a simple polygon P with n vertices, a single point b inside and set R of $O(n)$ red points inside, we can compute the shortest perimeter polygon P' that is contained in P, that contains all vertices of P and all red points of R, and that does not have b in its interior, in $O(n \log n)$ time.*

2.2 Several blue points inside P

2.2.1 Several blue and no red points inside P

For now, we assume that P is convex and has n vertices. With $m \ll n$ blue points inside the red polygon

P, we could try to adapt the first algorithm to compute a polygon that has all blue points no longer inside. However, as an iterative application of this algorithm may not lead to an optimal, smallest perimeter solution, we need to find a different approach. First we will prove some properties of an optimal solution.

Lemma 5 *Let P be a convex polygon with a set B with m blue points inside. In an optimal solution, let B be partitioned into k subsets. Then we have:*

1. *Each optimal path π of a subset B' consists of a part of the convex hull of B', and the two outer tangents of B' and some edge e of P, such that e and π form a convex polygon.*
2. *No two optimal paths intersect.*

It follows directly from this lemma that we need only consider partitionings of the blue points into subsets with disjoint convex hulls.

Lemma 6 *For a set of m points in the plane, there are $O(C^m)$ different partitionings into subsets with disjoint convex hulls, for some constant C.*

It is easy to see that every subset B_i chooses its optimal edge independently of all other sets. We give the following algorithm.

Let $\mathcal{P} = \{B_1, \ldots, B_k\}$ be a partitioning of the m blue points inside the polygon P into k subsets with disjoint convex hulls. For each B_i and every edge e of P we determine their outer tangents and compute the length of the shortest path. We store the optimal configuration with the shortest added length for $\mathcal{P}$. We do this for every possible partitioning $\mathcal{P}$ of the blue points inside P into subsets with disjoint convex hulls. Finally, we generate the polygon P' by replacing the appropriate edges of P with the optimal paths for the groups in the optimal partitioning.

Theorem 7 *For a convex polygon P with n vertices and m blue points inside, we can compute a minimum perimeter polygon P' that has no blue points in the interior and no vertices of P in the exterior, in $O(C^m \cdot n)$ time, for some constant C.*

In the case of a simple polygon P, we generate all possible $O(C^{m \log m})$ partitionings of the m blue points inside P. However, the properties of an optimal solution as well as the algorithm to find it, remain essentially the same. We can state:

Theorem 8 *For a simple polygon P with n vertices and m blue points inside, we can compute a minimum perimeter polygon P' that is contained in P, has no blue points in the interior and no vertices of P in the exterior, in $O(C^{m \log m} \cdot n)$ time, for some constant C.*

3 Recoloring methods

In this section we present the recoloring approach. We are given a set P of n points, each of which is either red or blue. We first compute the Delaunay Triangulation DT(P) of P. In DT(P), we color edges red if they connect two red points, blue if they connect two blue points, and green otherwise. A red point is incident only to red and green edges, and a blue point is incident only to blue and green edges. We will recolor a point if it is surrounded by points of the other color. We define for each point its green angle:

Definition 1 *Let $p \in P$ and let the edges of $DT(P)$ be colored as above. Then the* green angle *ϕ of p is*

- $360°$, *if p is only incident to green edges,*
- *the maximum turning angle between two or more radially consecutive incident green edges,*
- $0°$, *if p has at most one radially consecutive incident green edge (or no incident green edges).*

We recolor points only if their green angle ϕ is at least some threshold value $\Phi \geq 180°$; a suitable value can be found empirically. After the algorithm has terminated, we define the regions as follows. Let M be the set of midpoints of the green edges. Then, each Delaunay triangle contains either no or two points of M. For each triangle that contains two points of M, we connect them by a straight line segment. They define the boundary between the red and the blue region. Before we present specific recoloring schemes, we make the following basic observation.

Observation 1 *If we can recolor a blue point to be red, then we do not destroy this option if we first recolor other blue points to be red. If we can recolor a red point to be blue, then we do not destroy this option if we first recolor other red points to be blue.*

In the preferential recoloring scheme, we first recolor all blue points with green angle $\phi \geq \Phi$ red, and then all red points with green angle $\phi \geq \Phi$ blue. This scheme has linear running time, however, it is not fair and therefore not satisfactory.

3.1 The Angle-and-Perimeter Scheme

In the angle-and-perimeter scheme, we require that every recoloring decreases the perimeter of the separating polygon(s). Also, only points with green angle $\geq \Phi$ ($\Phi \geq 180°$) will be recolored. When there are several choices of recoloring a point, we select the one that has the largest green angle.

Theorem 9 *The number of recolorings in the angle-and-perimeter recoloring algorithm is at least $\Omega(n^2)$ and at most $2^n - 1$ in the worst case.*

The condition that recoloring is only allowed if it decreases the separation perimeter is needed, otherwise the separation perimeter may increase. Every recoloring implies the recoloring of all edges incident to the recolored point and updating its green angle as well as the green angle of all its neighbors and the perimeter of the polygons. We summarize:

Theorem 10 *The running time for the angle-and-perimeter recoloring algorithm is $O(n \cdot Z)$, where $Z \leq 2^n - 1$ denotes the maximum number of recolorings.*

3.2 The Angle-and-Degree Scheme

In the angle-and-degree scheme we require that a point p may only be recolored if the number of green edges incident to it goes down. This requirement is used in conjuction to the green angle $\geq \Phi$ condition for p. The degree condition gives a higher importance to the number of witnesses for the recoloring of a point. We need the following definition.

Definition 2 *A point p is a* good *neighbor of p' if p and p' are connected and have the same color, otherwise p is a* bad *neighbor of p'. Let $\delta(p)$, the δ-*value *of p, be the difference between the numbers of bad and good neighbors of p.*

We recolor a point p if its green angle ϕ is at least some threshold Φ and its δ-value is larger than some threshold $\delta_0 \geq 1$. This implies the recoloring of all edges incident to p and updating its green angle as well as the δ-value of p and all its neighbors. We can state the following two theorems:

Theorem 11 *The number of recolorings done in the angle-and-degree recoloring algorithm is $\Theta(n)$.*

Theorem 12 *The running time for the angle-and-degree recoloring algorithm is $O(\Delta \cdot Z)$, where $Z = \Theta(n)$ denotes the number of recolorings and Δ is the maximum degree in the Delaunay triangulation.*

4 Conclusions

This paper discussed algorithmic problems related to determining a reasonable boundary of a polygon, based on a set of points assumed to be inside (red), and a set of points assumed to be outside (blue). We presented two basic approaches. The first was to formulate the problem as a minimum perimeter polygon computation, based on an initial red polygon and one or more blue points that should not be inside. For the case of one point in the polygon we presented a linear time algorithm, and also $O(n \log n)$ time algorithm if there are also points that must stay inside. For the case of m points inside the polygon, we presented fixed-parameter tractable algorithms running in $O(C^m \cdot n)$ and $O(C^{m \log m} \cdot n)$ time, for convex and simple polygons, respectively.

The second approach involved changing the color, or inside-outside classification of points if they are surrounded by points of the other color. We proved a few lower and upper bounds on the number of recolorings for different criteria of recoloring. A remaining open problem is whether the angle-only version of this recoloring method terminates or not.

Another open problem is computing a minimum perimeter polygon for m blue points inside and n red points that must stay inside. Can a fixed-parameter tractable algorithm be given in this case as well?

Acknowledgements: Iris Reinbacher was also supported by a travel grant of the Netherlands Organization for Scientific Research (NWO). We thank Subodh Vaid, Hui Ma, and Markus Völker for implementing the algorithms.

References

[1] H. Alani, C. Jones, and D. Tudhope. Voronoi-based region approximation for geographical information retrieval with gazetteers. *Int. J. Geographical Information Science*, 15(4):287–306, 2001.

[2] A. Arampatzis, M. van Kreveld, I. Reinbacher, C. Jones, S. Vaid, P. Clough, H. Joho, and M. Sanderson. Web-based delineation of imprecise regions. manuscript, 2004.

[3] E. M. Arkin, F. Hurtado, J. S. B. Mitchell, C. Seara, and S. S. Skiena. Some separability problems in the plane. In *Abstracts EWCG 2000*, pages 51–54.

[4] T. M. Chan. Low-dimensional linear programming with violations. In *Proc. FOCS 2002*, pages 570–579.

[5] P. Eades and D. Rappaport. The complexity of computing minimum separating polygons. *Pattern Recogn. Lett.*, 14:715–718, 1993.

[6] H. Edelsbrunner, D. G. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *IEEE Trans. Inform. Theory*, IT-29:551–559, 1983.

[7] J. Gudmundsson and C. Levcopoulos. A fast approximation algorithm for TSP with neighborhoods and red-blue separation. *LNCS*, 1627:473–482, 1999.

[8] 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.

[9] A. Markowetz, T. Brinkhoff, and B. Seeger. Exploiting the internet as a geospatial database. In *Workshop on Next Generation Geospatial Information*, 2003.

[10] C. S. Mata and J. S. B. Mitchell. Approximation algorithms for geometric tour and network design problems (extended abstract). In *Proc. SoCG 1995*, pages 360–369.

[11] Y. Morimoto, M. Aono, M. Houle, and K. McCurley. Extracting spatial knowledge from the web. In *Proc. SAINT'03*. 2003.

An Efficient Algorithm for Label Updating in 2PM Model to Avoid a Moving Object

Farshad Rostamabadi* Mohammad Ghodsi †

Abstract

In this paper, we present a simple and fast algorithm for updating labels of a set of points in presence of a moving point-shaped object. The labels are assumed to be axis parallel, unit length, square shaped, each attached exclusively to a point on one of its horizontal (vertical) edges, denoted by 2PM model. The updated labeling should include all labels, avoid the moving point with largest possible label length. We allow flip and resize operations on labels for updating a labeling. The known algorithm for this problem, where labels may be attached to their corresponding points on the middle of any edges, the 4PM model, uses $O(n^2)$ preprocessing time and $O(n)$ space to update the labeling in $O(\lg n + k)$ time, where k is the number of update operations (Rostamabadi and Ghodsi, CCCG'04). In this paper, we present a simpler and more efficient algorithm that uses $O(n \lg n)$ time and $O(n)$ space for preprocessing with simplified data structures and updates the labeling with the same time bound.

1 Introduction

Automated label placement is an important problem in map generation, geographical information systems, and computer graphics. This problem, in its simple form, is to attach a label (regularly a text) to each point, line, curve, or a region in the map. Point-label placement has received good attention. In a valid labeling, labels should be pairwise disjoint, and each label should be attached to its feature point [2]. There are different variations of point-labeling that are discussed in [1, 3, 5, 6, 7].

In this paper, we are interested in a simple and efficient algorithm to update labels in a point-labeling map. We assume that our map is composed of a number of points each labeled by a unit-length axis-parallel square label. It is assumed that the point of each label appears in the middle of only one of its horizontal (vertical) edges. Hereafter, we denote this labeling model by 2PM[1]. Besides, a point-shaped object moves on the labeling and triggers an event every time it changes its location. Labels are allowed to be flipped or resized to avoid the moving object. The goal is to efficiently generate the updated labeling with maximum label length in response to each event triggered by the moving object.

A more general version of this problem, where each point can appear at the middle of any of four edges of its label, is considered in [4]; we refer to this labeling model as 4PM. Although, the proposed algorithm also works for our problem, but we are interested in a simpler algorithm with more efficient data structures.

Authors of [4] present a weighted and directed graph, *the conflict graph*, to convey the effect of all possible flip and resize operations. Since the conflict graph may have complex structure, the preprocessing phase requires $O(n^2)$ time in 4PM model.

We use the same definition of flip and resize operations as in [4], but we simplify the conflict graph definition by using some properties of the 2PM model, hence the preprocessing time reduces to $O(n \lg n)$. Besides, the result of the preprocessing phase can be stored in a more simpler data structures. We also show that the given labeling can be updated in $O(\lg n + k)$ time in response to each event triggered by the moving object, where k is the number of update operations.

2 Definitions

The label updating problem is precisely defined as follows. We are given a valid labeling L composed of points $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$ and unit-length axis-parallel square labels $\mathcal{L} = \{\ell_1, \ell_2, \ldots, \ell_n\}$, where ℓ_i is attached to p_i on the mid-point of one of its horizontal (vertical) edges. The moving point generates an event whenever it changes its current position to a new location q. The problem is to update L and obtain a new q-*avoiding* labeling L_q for all points in $\mathcal{P}$, such that the moving point q does not intersect with any label in L_q, while the new labels are as close as possible to their original length. A label ℓ_i can be flipped over the edge containing its corresponding point p_i and the flipped position of ℓ_i is denoted by $f(\ell_i)$. ℓ_i can also be resized to any length, $\alpha \leq 1$ as

*School of Computer Science, Institute for Studies in Fundamental Sciences, Tehran, Iran. rostamabadi@ipm.ir

†Computer Engineering Department, Sharif University of Technology, Tehran, Iran. ghodsi@sharif.edu

[1]2P is a known two-position label modeling, and we add an M to it to denote that the point should appear at the middle of one of its horizontal (or vertical) edges of the label.

long as p_i remains on the mid-point of its edge. The new location of ℓ_i is denoted by $r(\ell_i, \alpha)$. Such a final re-labeling is denoted by *q-avoiding optimum labeling*.

We name the above labeling model as 2PM. To be more precise, we formally define the 2PM and 4PM square-labeling models as follows:

Definition 1 *In 2PM model, every label is attached exclusively to a point on the middle of one of its horizontal (vertical) edges.*

Definition 2 *In 4PM model, every label is attached exclusively to a point on the middle of one of its edges.*

In next section, we formally define the conflict graph.

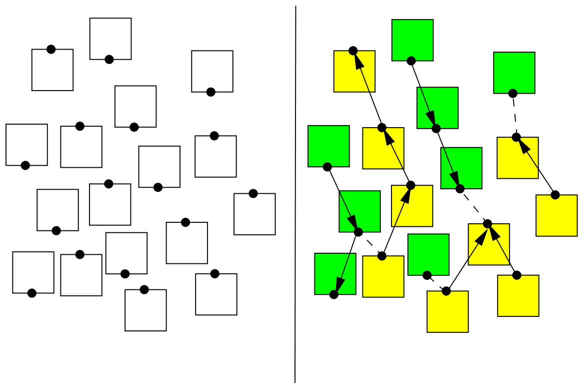

(a) (b)

Figure 1: (a) Initial labeled map. (b) The conflict graph: D^+ and D^- (solid) and B edges (dashed).

2.1 The Conflict Graph

The conflict graph $\mathcal{G} = (V, E)$, where $\mathcal{V} = V^+ \cup V^-$, is a directed weighted graph encoding all possible flip and resize operations. The conflict graph is both vertex and edge weighted. Let v_i be the representing vertex for label ℓ_i attached to the point p_i. The set V^+ (V^-) contains all vertices v_i where p_i is on the top (bottom) edge of ℓ_i.

The edge set of the conflict graph, $\mathcal{E}$, is composed of three sets D^+, D^-, and B as follows. The edge set D^+ (D^-) contains all directed edges (v_i, v_j) where both v_i and v_j belongs to V^+ (V^-) and $f(\ell_i)$ intersects ℓ_j. If $f(\ell_i)$ intersects ℓ_j but v_i and v_j are in different vertex sets, then the undirected edge (v_i, v_j) belongs to B. More formally:

$\mathcal{E} = D^+ \cup D^- \cup B$, where
$D^+ = \{(v_i, v_j) | v_i, v_j \in V^+\}$, and
$D^- = \{(v_i, v_j) | v_i, v_j \in V^-\}$, and
$B = \{(v_i, v_j) | v_i$ and v_j are not in the same vertex set$\}$.

A sample input labeling and the generated conflict graph are shown in Figure 1. The D^+ and D^- edges are solid while the B edges are dashed. Besides, the vertices of V^+ have yellow (light gray) labels and the vertices of V^- have green (dark gray) labels. We define two induced subgraphs over the V^+ and V^- vertices in the following and will prove some important properties for them in the next section.

$$G^+ = (V^+, D^+),$$
$$G^- = (V^-, D^-).$$

In $\mathcal{G}$, an edge weight represents the optimal label length of a single flip (and possibly one resize) operation on a single label, and a vertex weight represents the optimal label length if multiple flip and resize operations are allowed over all labels in the map. To define the weight of an edge, we first introduce the g function below.

The resize operation is needed when two (original or flipped) labels, say ℓ_a and ℓ_b, overlap. There are many resize values of γ_a and γ_b such that the resized labels $r(\ell_a, \gamma_a)$ and $r(\ell_b, \gamma_b)$ do not overlap. From the problem definition, we are interested in the values where $\min\{\gamma_a, \gamma_b\}$ is maximized. We define this value as the $g(\ell_a, \ell_b)$.

Performing a flip operation on ℓ_i may cause an intersection between $f(\ell_i)$ and at most two other labels. Let ℓ_j be one of the intersecting labels with $f(\ell_i)$. We define the weight of the edge (v_i, v_j), for both directed and undirected edges equal to $g(f(\ell_i), \ell_j)$.

The weight assigned to v_i, which denoted by $w(v_i)$, represents the label length of a labeling when ℓ_i is assumed to be flipped and other labels may be flipped or resized to generate the maximum possible label length. Obviously, the weight of vertex v_i has a very close relation to the optimal labeling when the moving point is inside ℓ_i. So, we first prove some basic properties of the optimal labeling and then complete the definition of vertex weight.

3 Properties of The Optimal Solution

Let the moving point be at position q inside label ℓ_i, and L_i^α be the optimal q-avoiding labeling with minimum number of flip and resize operations. Besides, assume that all labels in L_i^α have length larger than $\alpha \leq 1$. We define $G_i^\alpha = (V_i^\alpha, E_i^\alpha)$ as a subgraph of $\mathcal{G}$ where V_i^α contains all vertices that their corresponding labels are flipped. Let E_i^α be the induced edges over V_i^α (all edges of $\mathcal{E}$ with both ends in V_i^α). The following lemmas show some basic properties of G_i^α.

Lemma 1 *If $v_i \in V^+$ then all vertices of V_i^α are in V^+.*

Proof. Assume that $f(\ell_i)$ intersects a label ℓ_j where $v_j \in V^-$. Obviously, flipping ℓ_j produces smaller labels of length $g(f(\ell_i), f(\ell_j))$ than $g(f(\ell_i), \ell_j)$. So, in the optimal solution, the vertex $v_j \in V^-$ can not belong to L_i^α. □

From the above lemma, we can conclude that if $v_i \in V^+$ then $G_i^\alpha \subseteq G^+$. Besides, by the symmetry of G^+ and G^-, we can also conclude that if $v_i \in V^-$ then $G_i^\alpha \subseteq G^-$

Lemma 2 *G_i^α is a DAG, rooted at v_i.*

Proof. Since all vertices of G_i^α belong to either V^+ or V^-, then all edges are upward or downward. So, G_i^α can not contain a loop, and is a DAG. Besides, it is easy to see that there is a directed path (a sequence of flips) from v_i to any other vertex in G_i^α hence v_i is the root vertex of G_i^α. □

Lemma 3 *Let v_j be a leaf vertex in G_i^α and (v_j, v_k) be an edge attached to v_j. Then one of the followings is true:*

1. *If (v_j, v_k) exists then $w(v_j, v_k) \geq \alpha$,*
2. *v_j has zero out-degree and (v_j, v_k) does not exist.*

Proof. Will appear in final version. □

4 Vertex weight definition

In this section, we will provide a bottom-up recursive definition and algorithm for vertex weights using lemmas from previous section. The weight of v_i is the minimum label length in the optimal updated labeling when ℓ_i is flipped (equivalently, the query point is inside ℓ_i) and also corresponds to a subgraph of G^+ or G^- generating the optimal updated labeling. For simplicity, we focus on G^+, but all the definitions and the algorithm can also be applied to G^-.

First, we define the weight of zero out-degree vertices of G^+. Consider $v_i \in G^+$ where the out-degree of vertex v_i in G^+, denoted by $d^+_{\text{out}}(v_i)$, is zero. If v_i is also a zero out-degree vertex in $\mathcal{G}$, then $f(\ell_i)$ has no intersection with other labels hence the optimal updated labeling after flipping ℓ_i has no label of length less than one. Otherwise, the weight of undirected edges starting from v_i will define the vertex weight. More precisely, where $d^+_{\text{out}}(v_i) = 0$, we have:

$$w(v_i) = \min(\{1\} \cup \{w(v_i, v_j) | (v_i, v_j) \in B\}).$$

Second, for non-zero out-degree vertices, consider an edge $(v_i, v_j) \in D^+$ and assume that ℓ_i is flipped. There are two alternatives to remove the intersection between $f(\ell_i)$ and ℓ_j: (1) Resize both intersecting labels to some smaller length, and (2) Flip ℓ_j and solve the problem recursively (if applicable). The minimum generated label length is $g(f(\ell_i), \ell_j)$ in the former case (according to the definition of edge weight) and $w(v_j)$

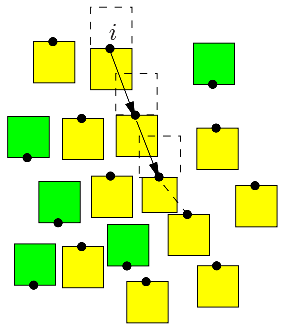


Figure 2: G_i^α and the optimal generated L_i^α.

in the latter case. We summarise the above description into a function, denoted by h, in the following:

$$h(v_i, v_j) = \begin{cases} \max(w(v_i, v_j), w(v_j)) & , v_j \in V^+ \\ w(v_i, v_j) & , v_j \notin V^+ \end{cases}$$

Using above function, the vertex weights can precisely be defined as:

$$w(v_i) = \begin{cases} \min\{\{1\} \cup \{w(v_i, v_j) | (v_i, v_j) \in B\}\} \\ \qquad\qquad , d^+_{\text{out}}(v_i) = 0 \\ \min\{h(v_i, v_j) | (v_i, v_j) \in \mathcal{E}\} \\ \qquad\qquad , d^+_{\text{out}}(v_i) \neq 0 \end{cases}$$

Obviously, using a simple bottom-up algorithm, all vertex weights can be calculated in $O(n)$ time.

The value of $w(v_i)$ defines a DAG subgraph, denoted by G_i, rooted at v_i and corresponds to the optimal labeling obtained by flipping ℓ_i. G_i is generated implicitly in the bottom-up calculation of $w(v_i)$ in time $O(n)$ for all vertices. But, if the value of $w(v_i)$ is available, it can simply be obtained using a top-down algorithm in time $O(|G_i|)$.

In figure 2 an optimal labeling L_i^α along with the corresponding G_i is shown for an arbitrary vertex.

Assume that the query point is inside ℓ_i and we have to flip ℓ_i. Having $w(v_i)$ been calculated in the preprocessing phase, we build G_i and produce the required update operations, in time $k = O(|G_i|)$, using the following transformation:

1. Flip label ℓ_m iff v_m is not a leaf vertex in G_i,
2. Flip and resize ℓ_j to $\min\{w(v_j, v_k) | (v_j, v_k) \notin G_i\}$ for all leaf vertices v_j.
3. Resize ℓ_k to length $\min\{w(v_j, v_k) | \ell_k \cap \ell_j \neq \emptyset \wedge v_j \in G_i \wedge (v_j, v_k) \notin G_i\}$ for all $v_k \notin G_i$.

According to lemma 3, it is easy to see the following theorem.

Theorem 4 *The vertex subgraph G_i of vertex v_i corresponds to the optimal labeling with labels of length at least $w(v_i)$, when the query point is inside ℓ_i and ℓ_i has to be flipped.*

5 The Optimal Algorithm

In this section, we provide the optimal algorithm for label updating in 2PM model. The algorithm has a preprocessing and an online phase. In the former phase, the optimal labeling lengths are computed and stored for each label (vertex) in overall time $O(n \lg n)$. In the later phase, we compute the operations required to update the labeling using the optimal labeling lengths in time $O(\lg n + k)$ where k is the number of flip and resize operations.

In the preprocessing phase the following steps should be taken:

```
Preprocessing Phase
1. Build the conflict graph and compute edge and
   vertices weights.
2. Build a point location data structure on all labels.
```

In the online phase, for a given point q, the optimal labeling can be generated with the following steps:

```
Online Phase
1. Locate $\ell_i$ containing $q$.
2. if no such label was found, then the original la-
   beling is optimal and exit.
3. Maximize $\gamma$ where $r(\ell_i, \gamma)$ does not contain $q$.
4. if $\gamma \geq w(v_i)$ then the resize operation $r(\ell_i, \gamma)$
   generate the optimal labeling and exit.
5. Calculate $G_i$.
6. Transform $G_i$ to the optimal labeling.
```

It is easy to see the following theorem.

Theorem 5 *Given a moving point q on a labeling L, the time required to generate an updated q-avoiding labeling is $O(\lg n + k)$ where k is the number of operation required to update L.*

6 Conclusion

In this paper, we introduced the problem of updating a squared axis-parallel labeled map to avoid a moving point. We develop a simple and fast algorithm and improve the previous results from $O(n^2)$ preprocessing time to $O(n \lg n)$ with the same space and query time. We modeled the initial labeling and update operations with a weighted multi-graph with at most $O(n)$ edges and vertices called conflict graph. We also showed that given a point q, the optimal q-avoiding labeling corresponds to a subgraph of the conflict graph that can be found in time $O(\lg n + k)$ where k is the number of update operations.

References

[1] Rob Duncan, Jianbo Qian, Antoine Vigneron, and Binhai Zhu. Polynomial time algorithms for three-label point labeling. *Theoretical Computer Science*, 296(1):75–87, 2003.

[2] Joe Marks and Stuart Shieber. The computational complexity of cartographic label placement. Technical Report TR-05-91, Harvard CS, 1991.

[3] Zhongping Qin, Alexander Wolff, Yinfeng Xu, and Binhai Zhu. New algorithms for two-label point labeling. In Mike Paterson, editor, *Proc. 8th Annu. Europ. Symp. on Algorithms (ESA'00)*, volume 1879 of *Lecture Notes in Computer Science*, pages 368–379, Saarbrücken, 5–8 September 2000. Springer-Verlag.

[4] Farshad Rostamabadi and Mohammad Ghodsi. A fast algorithm for updating a labeling to avoid a moving point. In *Proceedings of the 16th Canadian Conference on Computational Geometry (CCCG'04)*, pages 204–208, 2004.

[5] Michael J. Spriggs and J. Mark Keil. A new bound for map labeling with uniform circle pairs. *Information Processing Letters*, 81(1):47–53, 2002.

[6] Tycho Strijk and Alexander Wolff. Labeling points with circles. *International Journal of Computational Geometry and Applications*, 11(2):181–195, April 2001.

[7] Alexander Wolff, Michael Thon, and Yinfeng Xu. A simple factor-2/3 approximation algorithm for two-circle point labeling. *International Journal of Computational Geometry and Applications*, 12(4):269–281, 2002.

Exact Analysis of Optimal Configurations in Radii Computations

(Extended abstract)

René Brandenberg[*] Thorsten Theobald[†]

Abstract

We propose a novel characterization of (radii-) minimal projections of polytopes onto j-dimensional subspaces. Applied on simplices this characterization allows to reduce the computation of an outer radius to a computation in the circumscribing case or to the computation of an outer radius of a lower-dimensional simplex. This allows to close a gap in the knowledge on optimal configurations in radii computations, such as determining the radii of smallest enclosing cylinders of regular simplices in general dimension.

1 Introduction

Radii computations of the following form occur in many applications in computer vision, robotics, computational biology, and massive data set analysis (see [7] and the references therein). Let $\mathcal{L}_{j,n}$ be the set of all j-dimensional linear subspaces (hereafter *j-spaces*) in n-dimensional Euclidean space $\mathbb{E}^n$. The *outer j-radius* $R_j(C)$ of a convex body $C \subset \mathbb{E}^n$ is the radius of the smallest enclosing j-ball in an optimal orthogonal projection of C onto a j-space $J \in \mathcal{L}_{j,n}$, where the optimization is performed over $\mathcal{L}_{j,n}$. The optimal projections are called *R_j-minimal projections.* See [1, 5, 10] for exact algebraic algorithms, [8, 11, 14] for approximation algorithms, and [3, 7] for the computational complexity. In this paper we show the following new characterization of optimal projections:

Theorem 1 *Let $1 \le j \le n < m$ and $P = \operatorname{conv}\{v^{(1)}, \ldots, v^{(m)}\} \subset \mathbb{E}^n$ be an n-polytope. Then one of the following is true.*

a) In every R_j-minimal projection of P there exist $n+1$ affinely independent vertices of P which are projected onto the minimal enclosing j-sphere.

b) $j \ge 2$ and $R_j(P) = R_{j-1}(P \cap H)$ for some hyperplane $H = \operatorname{aff}\{v^{(i)} : i \in I\}$ with $I \subset \{1, \ldots, m\}$.

If $j = 1$ or if P is a regular simplex then always case a) holds. Moreover, the number ν of affinely independent vertices projected onto the minimal enclosing j-sphere is at least $n-j+2$ and there exists a $(\nu-1)$-flat F such that $R_j(P) = R_{j+\nu-n-1}(P \cap F)$. The bound $n-j+2$ is best possible.

Theorem 1 allows to reduce the computation of an outer radius of a simplex to the computation in the circumscribing case or to the computation of an outer radius of a facet of the simplex. Reductions of smallest enclosing cylinders to circumscribing cylinders are used in exact algorithms as well as for complexity proofs (see, e.g., [1] and [7]), and have previously been given only for $j \in \{1, n\}$ as well as for dimension 3. Theorem 1 generalizes and unifies these results.

The characterization provides effective means for the analysis of optimal configurations in radii computations (for general dimension a known difficult task). As an example, we reduce the computation of the outer $(n-1)$-radius of a regular simplex to the following optimization problem of symmetric polynomials in n variables:

$$\begin{array}{llll} \min & \sum_{i=1}^{n+1} s_i^4 & \text{s.t.} & \sum_{i=1}^{n+1} s_i^3 = 0\,, \\ & \sum_{i=1}^{n+1} s_i^2 = 1\,, & \text{and} & \sum_{i=1}^{n+1} s_i = 0\,. \end{array} \qquad (1)$$

The system is solved by reducing it to an optimization problem in six variables with additional integer constraints, leading to the following result.

Theorem 2 *Let $n \ge 2$ and T_1^n be a regular simplex in $\mathbb{E}^n$ with edge length 1. Then*

$$R_{n-1}(T_1^n) = \begin{cases} \sqrt{\frac{n-1}{2(n+1)}} & \text{if } n \text{ is odd,} \\ \frac{2n-1}{2\sqrt{2n(n+1)}} & \text{if } n \text{ is even.} \end{cases}$$

The case n odd has already been settled independently by Pukhov [9] and Weißbach [12] who both left open the even case. There also exists a later paper on $R_{n-1}(T_1^n)$ for even n [13], but as pointed out in [1] the proof contained a crucial error. Thus Theorem 2 (re-)completes the determination of the sequence of outer j-radii of regular simplices [9]. [1]

[*]Zentrum Mathematik, Technische Universität München, Boltzmannstr. 3, D–85747 Garching bei München, brandenb@ma.tum.de

[†]Dept. of Computer Science, Yale University, P.O. Box 208285, New Haven, CT 06520–8285, theobald@cs.yale.edu

[1]All omitted proofs as well as further analysis of the problems can be found in the full paper [2].

2 Preliminaries

Throughout the paper we work in Euclidean space $\mathbb{E}^n$, i.e., $\mathbb{R}^n$ with the usual scalar product $x \cdot y$ and norm $||x|| = (x \cdot x)^{1/2}$. $\mathbb{B}^n$ and $\mathbb{S}^{n-1}$ denote the (closed) unit ball and unit sphere, respectively. For a set $A \subset \mathbb{E}^n$, the linear, affine, and convex hull of A are denoted by $\mathrm{lin}(A)$, $\mathrm{aff}(A)$, and $\mathrm{conv}(A)$, respectively.

A set $C \subset \mathbb{E}^n$ is called a *body* if it is compact, convex and contains interior points. Accordingly, we always assume that a polytope $P \subset \mathbb{E}^n$ is full-dimensional (unless otherwise stated). Let $1 \le j \le n$. A *j-flat* F (an affine subspace of dimension j) is *perpendicular* to a hyperplane H with normal vector h if h and F are parallel. For $p, p' \in \mathbb{E}^n$ and subspaces $E \in \mathcal{L}_{j,n}$, $E' \in \mathcal{L}_{j',n}$, a j-flat $F = p + E$ and a j'-flat $F' = p' + E'$ are *parallel* if $E \cup E' = \mathrm{lin}(E \cup E')$. A *$j$-cylinder* is a set of the form $J + \rho\mathbb{B}^n$ with an $(n-j)$-flat J and $\rho > 0$. Let $1 \le j \le k \le n$. If $C' \subset \mathbb{E}^n$ is a compact, convex set whose affine hull F is a k-flat then $R_j(C')$ denotes the radius of a smallest enclosing j-cylinder $\mathcal{C}'$ relative to F, i.e., $\mathcal{C}' = J' + R_j(C')(\mathbb{B}^n \cap F)$ with a $(k-j)$-flat $J' \subset F$.

A *simplex* $\mathrm{conv}\{v^{(1)}, \dots, v^{(n+1)}\}$ (with affinely independent $v^{(1)}, \dots, v^{(n+1)} \in \mathbb{E}^n$) is *regular* if all its vertices are equidistant. Whenever a statement is invariant under orthogonal transformations and translations we denote by T^n *the* regular simplex in $\mathbb{E}^n$ with edge length $\sqrt{2}$. Let $\mathcal{H}^n_\alpha = \{x \in \mathbb{E}^{n+1} : \sum_{i=1}^{n+1} x_i = \alpha\}$. Then the *standard embedding* $\mathbf{T}^n$ of T^n is defined by $\mathbf{T}^n = \mathrm{conv}\left\{e^{(i)} \in \mathbb{E}^{n+1} : 1 \le i \le n+1\right\} \subset \mathcal{H}^n_1$, where $e^{(i)}$ denotes the i-th unit vector in $\mathbb{E}^{n+1}$. By $\mathcal{S}^{n-1} := \mathbb{S}^n \cap \mathcal{H}^n_0$ we denote the set of unit vectors parallel to $\mathcal{H}^n_1$. A j-cylinder $\mathcal{C}$ containing some simplex S is called a *circumscribing* j-cylinder of S if all the vertices of S are contained in the boundary of $\mathcal{C}$.

3 Minimal and circumscribing j-cylinders

The minimal enclosing ball B of a polytope $P \subset \mathbb{E}^n$ may contain only few vertices of P on its boundary, but in cases where less than $n+1$ vertices of P are contained in the boundary of B, there exists a hyperplane H such that $P \cap \mathrm{bd}(B) \subset H$ and the center of B is contained in H. Then the smallest enclosing ball of P and the smallest enclosing ball of $P \cap H$ relative to H have the same radius. In [6] the following characterization for the minimal enclosing 1-cylinder (two parallel hyperplanes defining the width of the polytope) is given:

Proposition 3 *Any minimal enclosing 1-cylinder of a polytope $P \subset \mathbb{E}^n$ contains at least $n+1$ affinely independent vertices of P on its boundary.*

We provide a characterization of the possible configurations of minimal enclosing j-cylinders of polytopes, unifying and generalizing the above statements.

Lemma 4 *Let $P = \mathrm{conv}\{v^{(1)}, \dots, v^{(m)}\}$ be a polytope in $\mathbb{E}^n$, $1 \le j \le n-1$, and J be an $(n-j)$-flat such that $\mathcal{C} = J + R_j(P)\mathbb{B}^n$ is a minimal enclosing j-cylinder of P. Then for every $I \subset \{1, \dots, m\}$ such that $\{i : v^{(i)} \in \mathrm{bd}(\mathcal{C})\} \subset I$ and $H_I := \mathrm{aff}\{v^{(i)} : i \in I\}$ is of affine dimension $n-1$, J is parallel to H_I.*

Proof. Suppose that there exists a hyperplane $H := H_I$ of this type with J not parallel to H. Let $\bar{n} := |\{v^{(i)} \in H : 1 \le i \le m\}|$. Without loss of generality $H = \{x \in \mathbb{E}^n : x_n = 0\}$ and $I = \{v^{(1)}, \dots, v^{(\bar{n})}\}$. Hence, $v^{(\bar{n}+1)}, \dots, v^{(m)} \notin H \cup \mathrm{bd}(C)$.

It suffices to consider the case that J is not perpendicular to H. Let $p, s^{(1)}, \dots, s^{(n-j)} \in \mathbb{E}^n$ such that $J = p + \mathrm{lin}\{s^{(1)}, \dots, s^{(n-j)}\}$. Since J is not parallel to H, we can assume $p = 0 \in J \cap H$, $s^{(1)}_n = \dots = s^{(n-j-1)}_n = 0$ and $s^{(n-j)}_n > 0$. For every $s'_n \in (0, s^{(n-j)}_n)$ and $s' := (s^{(n-j)}_1, \dots, s^{(n-j)}_{n-1}, s'_n) \in \mathbb{E}^n$ let $J' = p + \mathrm{lin}\{s^{(1)}, \dots, s^{(n-j-1)}, s'\}$. Since J and H are not perpendicular we obtain $J \neq J'$, and because $v^{(1)}, \dots, v^{(\bar{n})} \in H$ that

$$\mathrm{dist}(v^{(i)}, J') \le \mathrm{dist}(v^{(i)}, J), \quad 1 \le i \le \bar{n}, \qquad (2)$$

where $\mathrm{dist}(\cdot,\cdot)$ denotes the Euclidean distance. In (2), "$<$" holds whenever $v^{(i)} \notin K := J^\perp \cap H$. Obviously, $\dim(K) = j-1$. If none of the $v^{(i)}$ lies in $K \cap \mathrm{bd}(\mathcal{C})$ then, by choosing s'_n sufficiently close to $s^{(n-j)}_n$, all vertices of P lie in the interior of $\mathcal{C}' = J' + R_j(P)\mathbb{B}^n$, a contradiction to the minimality of $\mathcal{C}$. Hence, there must be some vertex of P in $K \cap \mathrm{bd}(\mathcal{C})$. Let $\bar{k} := |\{v^{(i)} \in K \cap \mathrm{bd}(\mathcal{C}) : 1 \le i \le m\}|$. We can assume that $v^{(1)}, \dots, v^{(\bar{k})} \in K \cap \mathrm{bd}(\mathcal{C})$. Let $F := \mathrm{conv}\{v^{(1)}, \dots, v^{(\bar{k})}\}$ and $k := \dim F$. Suppose $F \cap J = \emptyset$. We have shown above that for sufficiently small s'_n the rotation from J to J' keeps all vertices within the j-cylinder $\mathcal{C}'$ and $v^{(1)}, \dots, v^{(\bar{k})}$ are the only vertices on $\mathrm{bd}(\mathcal{C}')$. Let J'' be a translate of J' with $\mathrm{dist}(J'', F) < \mathrm{dist}(J', F)$, and J'' sufficiently close to J' to keep $v^{(\bar{k}+1)}, \dots, v^{(m)}$ within the interior of $\mathcal{C}'' = J'' + R_j(P)\mathbb{B}^n$. Then all vertices of P lie in the interior of $\mathcal{C}''$, again a contradiction.

It follows that $F \cap J \neq \emptyset$, and since $F \subset K = J^\perp \cap H$ that $F \cap J = p = 0$. Since $\mathrm{dist}(p, v^{(i)}) = R_j(P)$ for all $i \in \{1, \dots, \bar{k}\}$ and since $p \in F$, it follows that p is the unique center of the smallest enclosing k-ball of F. Let J''' result from J' by rotating J' around the origin towards a direction in $\mathbb{R}^n \setminus (\bigcup_{i=1}^{\bar{k}} (v^{(i)})^\perp)$. For $i \in \{1, \dots, \bar{k}\}$ the property $\mathrm{dist}(v^{(i)}, J) = \mathrm{dist}(v^{(i)}, J') = \mathrm{dist}(v^{(i)}, p)$ implies $\mathrm{dist}(v^{(i)}, J''') < \mathrm{dist}(v^{(i)}, J')$. By keeping the rotation sufficiently small, $v^{(\bar{k}+1)}, \dots, v^{(m)}$ remain in the interior of $\mathcal{C}''' = J''' + R_j(P)\mathbb{B}^n$. Now, all vertices lie in the interior of $\mathcal{C}'''$, once more a contradiction. □

Lemma 5 *Let $P = \mathrm{conv}\{v^{(1)}, \dots, v^{(m)}\}$ be a polytope in $\mathbb{E}^n$, $1 \le j \le n$, and J be an $(n-j)$-flat* *such that $\mathcal{C} = J + R_j(P)\mathbb{B}^n$ is a minimal enclosing j-cylinder of P. If there exists a hyperplane $H_I = \operatorname{aff}\{v^{(i)} : i \in I\}$ which is parallel to J, then one of the following holds:*

a) There exists a vertex $v^{(i)} \notin H_I$ that lies on the boundary of $\mathcal{C}$; or

b) $j \geq 2$, $J \subset H_I$, and $R_j(P) = R_{j-1}(P \cap H_I)$.

Proof. By Proposition 3, for $j = 1$ always a) holds; so let $j \geq 2$, and suppose neither a) nor b) holds. Since b) does not hold there exist $(n-j)$-flats parallel to J and closer to H_I, and since a) does not hold, for any such $(n-j)$-flat J', such that all vertices $v^{(i)} \notin H_I$ stay within $\mathcal{C}$, the distances from the vertices $v^{(i)}$, $i \in I$, to J' are strictly smaller than their distances to J. Hence $\mathcal{C}$ cannot be a minimal enclosing cylinder. $\square$

In the case that P is a simplex, the proof can be carried out more explicitly: Let $P^{(n+1)}$ be the facet of P not including the vertex $v^{(n+1)}$. Suppose that J is parallel to $P^{(n+1)}$, that $P^{(n+1)} \subset H := \{x \in \mathbb{E}^n : x_n = 0\}$, and that $v_n^{(n+1)} > 0$. Let $p \in J$. Since $v_n^{(n+1)} > 0$ it follows $p_n \geq 0$ and obviously

$$R_j(P) \geq v_n^{(n+1)} - p_n. \tag{3}$$

On the other hand, since J is parallel to $P^{(n+1)}$,

$$R_j(P)^2 = R_{j-1}^2(P^{(n+1)}) + p_n^2. \tag{4}$$

Let $p_n^* = ((v_n^{(n+1)})^2 - R_{j-1}^2(P^{(n+1)}))/2v_n^{(n+1)}$ be the unique minimal solution for p_n to (3) and (4). Due to $p_n \geq 0$, we obtain $p_n = \max\{0, p_n^*\}$. Now, we see that case a) holds if $p_n = p_n^*$ and case b) if $p_n = 0$.

If the number ν of affinely independent vertices of P lying on the boundary of $\mathcal{C}$ is at most n, it follows from Lemma 4 and 5 that case b) of Theorem 1 must hold. Moreover, if $\nu \leq n-1$ we can apply these lemmas on the lower-dimensional polytope $P \cap H_I$ with H_I as in Lemma 5. This argument can be iterated. If during this iteration the outer 1-radius of a polytope P' has to be computed, then by Proposition 3 the minimal enclosing 1-cylinder touches at least $\dim(P')+1$ affinely independent vertices. From the same iterative argument it follows that $R_j(P) = R_{j+\nu-n-1}(P \cap F)$ for some $(\nu-1)$-flat F.

Suppose $S = \operatorname{conv}\{v^{(1)}, \dots, v^{(n+1)}\}$ is a simplex in $\mathbb{E}^n$, and $\bar{J}$ an $(n-j)$-flat, such that

$$\begin{aligned} \operatorname{dist}(v^{(1)}, J) &= \cdots = \operatorname{dist}(v^{(n-j+2)}, J) \\ &= R_1(\operatorname{conv}\{v^{(1)}, \dots, v^{(n-j+2)}\}) \\ &> \operatorname{dist}(v^{(n-j+3)}, J) \\ &\geq \cdots \geq \operatorname{dist}(v^{(n+1)}, J). \end{aligned}$$

Then $R_j(S) = R_1(\operatorname{conv}\{v^{(1)}, \dots, v^{(n-j+2)}\})$ and $n-j+2$ vertices are situated on the boundary of the minimal enclosing j-cylinder.

The last point which remains to proof Theorem 1 is that every minimal enclosing j-cylinder of the regular simplex T^n is circumscribing. Due to Proposition 4 it suffices to show that p_n^* is positive for all $1 \leq j \leq n-1$, showing that b) in Lemma 5 never holds for T^n. We omit the details and refer to the full paper [2].

4 Reduction to an algebraic optimization problem

In this section, we provide an algebraic formulation for a minimal circumscribing j-cylinder $J + \rho(\mathbb{B}^{n+1} \cap \mathcal{H}_0^n)$ of the regular simplex $\mathbf{T}^n$. Let $J = p + \operatorname{lin}\{s^{(1)}, \dots, s^{(n-j)}\}$ with pairwise orthogonal (p.o.) $s^{(1)}, \dots, s^{(n-j)} \in \mathcal{S}^{n-1}$, and p be contained in the orthogonal complement of $\operatorname{lin}\{s^{(1)}, \dots, s^{(n-j)}\}$. The projection P of a vector $z \in \mathcal{H}_1^n$ onto the orthogonal complement of $\operatorname{lin}\{s^{(1)}, \dots, s^{(n-j)}\}$ (relative to $\mathcal{H}_1^n$) can be written as $P(z) = (I - \sum_{k=1}^{n-j} s^{(k)}(s^{(k)})^T)z$, where I denotes the identity matrix. Using the convention $x^2 := x \cdot x$, the computation of the square of R_j for a polytope with vertices $v^{(1)}, \dots, v^{(m)}$ (embedded in $\mathcal{H}_1^n$) can be expressed as

$$\begin{array}{llrcl} & & \min \rho^2 & & \\ \text{(i)} & \text{s.t.} & (p - Pv^{(i)})^2 & \leq & \rho^2\,, \\ \text{(ii)} & & p \cdot s^{(k)} & = & 0\,, \\ \text{(iii)} & & s^{(1)}, \dots, s^{(n-j)} & \in & \mathcal{S}^{n-1}, \quad \text{p.o.}, \\ \text{(iv)} & & p & \in & \mathcal{H}_1^n\,, \end{array}$$

where $i = 1, \dots, m$ and $k = 1, \dots, n-j$. In the case of $\mathbf{T}^n$, (i) can be replaced by

$$\text{(i')} \qquad \left(p - e^{(i)} + \sum_{k=1}^{n-j} s_i^{(k)} s^{(k)}\right)^2 = \rho^2,$$

where the equality sign follows from Theorem 1. By (ii) and $s^{(k)} \in \mathcal{S}^{n-1}$, (i') can be simplified to

$$\text{(i'')} \qquad p^2 - \rho^2 = \sum_{k=1}^{n-j} (s_i^{(k)})^2 + 2p_i - 1\,.$$

Summing over all i gives $(n+1)(p^2 - \rho^2) = (n-j) + 2 - (n+1)$, i.e., $p^2 - \rho^2 = \frac{1-j}{n+1}$. We substitute this value into (i'') and obtain $p_i = \frac{1}{2}\left(\frac{n-j+2}{n+1} - \sum_{k=1}^{n-j}(s_i^{(k)})^2\right)$. Hence, all the p_i can be replaced in terms of the $s_i^{(k)}$,

$$\begin{aligned} \rho^2 &= \frac{(2+(n-j))(2-(n-j))}{4(n+1)} \\ &\quad + \frac{1}{4}\sum_{i=1}^{n+1}\left(\sum_{k=1}^{n-j}(s_i^{(k)})^2\right)^2 + \frac{j-1}{n+1}\,, \qquad (5) \\ p \cdot s^{(k)} &= -\frac{1}{2}\sum_{i=1}^{n+1}\sum_{k'=1}^{n-j}(s_i^{(k')})^2 s_i^{(k)}\,. \end{aligned}$$

We arrive at the following characterization of the minimal enclosing j-cylinders:

Theorem 6 *Let $1 \le j \le n$. A set of vectors $s^{(1)}, \dots, s^{(n-j)} \in \mathcal{S}^{n-1}$ spans the underlying $(n-j)$-dimensional subspace of a minimal enclosing j-cylinder of $\mathbf{T}^n \subset \mathcal{H}_1^n$ if and only if it is an optimal solution of the problem*

$$\begin{array}{lrcl} \min & \sum\limits_{i=1}^{n+1} \left(\sum\limits_{k=1}^{n-j} (s_i^{(k)})^2 \right)^2 & & \\ \text{s.t.} & \sum\limits_{i=1}^{n+1} \sum\limits_{k'=1}^{n-j} (s_i^{(k')})^2 s_i^{(k)} & = & 0\,, \\ & s^{(1)}, \dots, s^{(n-j)} & \in & \mathcal{S}^{n-1}, \quad \text{p.o.}, \end{array}$$

where $k = 1, \dots, n-j$.

In case $j = n-1$ the previous program reduces to (1). By (5), in order to prove $R_{n-1}(T^n) = (2n-1)/(2\sqrt{n(n+1)})$ for even n, we have to show that the optimal value of (1) is $1/n$. We apply the following statement from [1].

Proposition 7 *Let $n \ge 2$. The direction vector $(s_1, \dots, s_{n+1})^T$ of any extreme circumscribing $(n-1)$-cylinder of $\mathbf{T}^n$ satisfies $|\{s_1, \dots, s_{n+1}\}| \le 3$.*

Using Proposition 7, (1) can be written as the following polynomial optimization problem in six variables with additional *integer* conditions.

$$\begin{array}{llrcl} & \min & k_1 s_1^4 + k_2 s_2^4 + k_3 s_3^4 & & \\ \text{(i)} & \text{s.t.} & k_1 s_1^3 + k_2 s_2^3 + k_3 s_3^3 & = & 0\,, \\ \text{(ii)} & & k_1 s_1^2 + k_2 s_2^2 + k_3 s_3^2 & = & 1\,, \\ \text{(iii)} & & k_1 s_1 + k_2 s_2 + k_3 s_3 & = & 0\,, \\ \text{(iv)} & & k_1 + k_2 + k_3 & = & n+1\,, \\ & & s_1, s_2, s_3 \in \mathbb{R}, & & k_1, k_2, k_3 \in \mathbb{N}_0\,. \end{array} \tag{6}$$

Since the odd case of Theorem 2 is well-known [9, 12], we assume from now on that n is even.

For $k_3 = 0$ the equality constraints in (6) immediately yield $k_1 = k_2 = (n+1)/2 \notin \mathbb{N}$, and similarly, for $s_2 = s_3$ we obtain $k_1 = k_2 + k_3 = (n+1)/2 \notin \mathbb{N}$. Hence, we can assume that s_1, s_2, and s_3 are distinct and $k_1, k_2, k_3 \ge 1$. Moreover, for $s_3 = 0$ the resulting optimal value is $1/n$ which will turn out to be the optimal solution. Finally, by (iii), not all of the s_i have the same sign. Hence it suffices to show that for $s_1 < 0$ and $s_3 > s_2 > 0$ every admissible solution to the constraints of (6) has value at least $1/n$.

The linear system in k_1, k_2, k_3 defined by (i), (ii), and (iii) is regular and can be solved for k_1, k_2, k_3:

$$k_1 = \frac{s_2 + s_3}{-s_1(s_2 - s_1)(s_3 - s_1)}\,, \tag{7}$$

$$k_2 = \frac{s_1 + s_3}{s_2(s_2 - s_1)(s_3 - s_2)}\,, \tag{8}$$

$$k_3 = \frac{-(s_1 + s_2)}{s_3(s_3 - s_1)(s_3 - s_2)}\,. \tag{9}$$

Since all factors in the denominators are strictly positive, (8) and (9) imply in particular $s_1 + s_3 > 0$ and $s_1 + s_2 < 0$.

With (iv) in (6) we can express one of the s_i by the others, e.g. $s_2 = -\frac{s_1 + s_3}{(n+1)s_1 s_3 + 1}$, and using this it can be successively shown that $k_1 < (n+1)/2$. Thus by the integer condition $k_1 \le n/2$, and it follows that for any admissible solution to the constraints of (6) the objective value is at least $1/n$ (for details see [2]). By our remark before Proposition 7 this completes the proof of Theorem 2.

References

[1] R. Brandenberg, T. Theobald. Algebraic methods for computing smallest enclosing and circumscribing cylinders of simplices. *Appl. Algebra Eng. Commun. Comput.* 14:439–460, 2004.

[2] R. Brandenberg, T. Theobald. Radii minimal projections of polytopes and constrained optimization of symmetric polynomials. Preprint.

[3] A. Brieden, P. Gritzmann, R. Kannan, V. Klee, L. Lovász, M. Simonovits. Deterministic and randomized polynomial-time approximation of radii. *Mathematika* 48:63–105, 2001.

[4] T.M. Chan. Approximating the diameter, width, smallest enclosing cylinder and min.-width annulus. *Int. J. Comp. Geom. & Appl.* 12:67–85, 2002.

[5] O. Devillers, B. Mourrain, F. Preparata, Ph. Trébuchet. On circular cylinders through 4 or 5 points. *Disc. Comput. Geom.* 29:83–104, 2002.

[6] P. Gritzmann, V. Klee. Inner and outer j-radii of convex bodies in finite-dimensional normed spaces. *Disc. Comput. Geom.* 7:255-280, 1992.

[7] P. Gritzmann, V. Klee. Computational complexity of inner and outer j-radii of polytopes. *Math. Program.* 59A:163-213, 1993.

[8] S. Har-Peled, K. Varadarajan. High-dimensional shape fitting in linear time. *Disc. Comput. Geom.* 32:269-288, 2004.

[9] S.V. Pukhov. Kolmogorov diameters of a regular simplex. *Mosc. Univ. Math. Bull.* 35: 38–41, 1980.

[10] E. Schömer, J. Sellen, M. Teichmann, C. Yap. Smallest enclosing cylinders. *Algorithmica* 27:170–186, 2000.

[11] K.R. Varadarajan, S. Venkatesh, J. Zhang. On approximating the radii of point sets in high dimensions, *FOCS 2002*, Vancouver, pp. 561–569.

[12] B. Weißbach. Über die senkrechten Projektionen regulärer Simplexe, *Beitr. Algebra Geom.* 15:35–41, 1983.

[13] B. Weißbach. Über Umkugeln von Projektionen regulärer Simplexe, *Beitr. Algebra Geom.* 16:127–137, 1983.

[14] Y. Ye, J. Zhang. An improved algorithm for approximating the radii of point sets, *Proc. Approximation Algorithms for Combinatorial Optimization*, LNCS 2764:178–187, Springer, 2003.

Boolean Operations on 3D Selective Nef Complexes: Optimized Implementation and Experiments[*]

Peter Hachenberger Lutz Kettner

Abstract

Nef polyhedra in d-dimensional space are the closure of half-spaces under boolean set operations. In consequence, they can represent non-manifold situations, open and closed sets, mixed-dimensional complexes and they are closed under all boolean and topological operations.

We implemented a boundary representation of three-dimensional Nef polyhedra with efficient algorithms for boolean operations. These algorithms were designed for correctness and can handle all cases, in particular all *degeneracies*. The implementation is released as Open Source in the CGAL release 3.1.

In this paper, we present experiments in order to (i) evaluate the practical runtime complexity, (ii) illustrate the effectiveness of several important optimizations, and (iii) compare our implementation with the ACIS CAD kernel.

1 Introduction

Data structures for solids and algorithms for boolean operations on geometric models are among the fundamental problems in solid modeling, computer aided design, and computational geometry. We restrict ourselves to partitions of three space into cells induced by planes. A set of planes partition space into cells of various dimensions. Each cell may carry a label. We call such a partition together with the labelling of its cells a *Selective Nef Complex (SNC)*. When the labels are boolean($\{in, out\}$) the complex describes a set, a so-called *Nef polyhedron* [3]. Nef polyhedra can be obtained from halfspaces by boolean operations intersection and complement. Figure 1 shows a Nef polyhedron.

We gave a compact and unique representation of SNCs and algorithms realizing set operations based on this representation in [2]. The current implementation supports the construction of Nef polyhedra from manifold solids, boolean operations, topological operations, and rotations by rational rotation matrices. We follow the exact geometric computation paradigm [6] to achieve robustness.

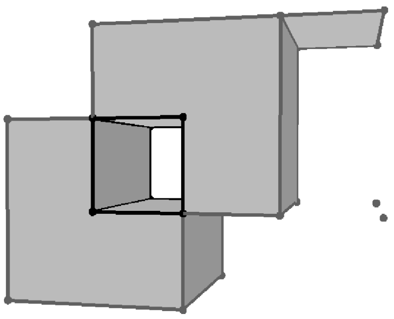


Figure 1: Nef polyhedron with non-manifold edges, a dangling facet, and two isolated vertices.

So far, we focused on the completeness of our algorithms. In this work, we evaluate their performance and present optimizations for important common cases. We evaluate their effectiveness individually and combined in a series of experiments.

Our current optimized implementation has become efficient enough to compare it with other systems. We selected ACIS R13, a common commercial CAD kernel used in many CAD systems [5].

2 Data Structures

Definition 1 (Nef polyhedron [3]) *A* Nef-polyhedron *in dimension d is a point set $P \subseteq \mathbb{R}^d$ generated from a finite number of open halfspaces by set complement and set intersection operations.*

Set union, difference and symmetric difference can be reduced to intersection and complement. Set complement changes between open and closed halfspaces, thus the topological operations *boundary*, *interior*, *exterior*, *closure*, and *regularization* are also in the modeling space of Nef polyhedra. In what follows, we refer to Nef polyhedra whenever we say polyhedra and we restrict ourselves to three dimensions.

In our representation for three-dimensional Nef-complexes, we use two main data structures:

Sphere Maps represent the local neighborhoods at each vertex. We conceptually intersect the local neighborhood of a vertex with a small ε-sphere. We obtain a planar map on the sphere (Figure 2), which

[*]Max-Planck Institut für Informatik, Saarbrücken, Germany, `[hachenberger|kettner]@mpi-sb.mpg.de`.

Partially supported by the IST Programme of the EU as a Shared-cost RTD (FET Open) Project under Contract No IST-2000-26473 (ECG - Effective Computational Geometry for Curves and Surfaces)

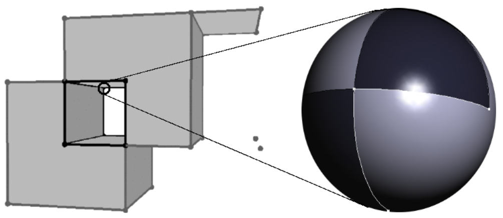


Figure 2: An example of a sphere map. The different colors indicate selected and unselected faces.

forms a two-dimensional Nef polyhedron embedded in the sphere. Sphere maps were introduced in [1].

The *Selective Nef Complex* provides a more easily accessible polyhedron representation. It additionally stores the connections between the local sphere maps. In detail it provides halfedges, facet cycles, halffacets, shells and volumes.

Additionally, we have a kd-tree associated with each Nef polyhedron. It provides fast point location and ray shooting needed in binary operations.

3 Binary Boolean Operations

Based on the SNC data structure, we can implement the binary boolean set operations. We find the sphere maps of all vertices of the resulting polyhedron and synthesize the SNC from there; in more detail:

1. Find possible candidate vertices. We take as candidates the original vertices of both input polyhedra, and we create all intersection points of edge-edge and edge-face intersections.
2. Given a candidate vertex, we find its local sphere map in each input polyhedron. If the candidate vertex is a vertex of one of the input polyhedra, its sphere map is already known. Otherwise a new sphere map is constructed on the fly.
3. Given the two sphere maps for a candidate vertex, we apply the boolean operation on sphere map to obtain the resulting sphere map.

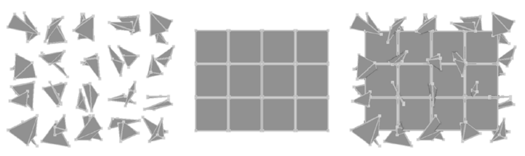
Boolean operations on spheres are an extension of a planar sweep. Instead of a sweep line in the plane, the spheres are cut into two hemispheres and a great arc sweeps around each hemisphere.

4 Experiments

We experimentally evaluate the runtime behavior of our implementation, in particular the binary boolean operations. We have several experiments that support the expected runtime, and we have designed experiments to stress our implementation with worst-case situations. We report on a subset here.

The tests are performed on a 846 MHz Pentium III processor and 256MB RAM. We measure the total runtime and the runtime of the following subroutines:

1. **Point location:** queries kd-tree of an input polyhedron to locate a vertex of the other polyhedron.
2. **Box intersection:** intersection finding on the bounding boxes only, excludes the cost of the callback function, i.e., the intersection test on the actual edge and facet geometry.
3. **Sphere sweeps:** sum of all sphere sweeps performed during boolean operations on sphere maps.
4. **Synthesizing edges:** in the synthesis step, sorts an edge representation based on Plücker coordinates.
5. **Plane sweeps:** in the synthesis step, sorts facet boundary cycles of the result polyhedron.
6. bf Kd-tree construction: in the synthesis step, initializes the kd-tree for the result polyhedron.
7. **Others:** all other parts not listed explicitly in the same graph, i.e. parts which have no critical worst-case or no interesting practical runtime.



Experiment TetGrid

1. Create a regular N^3 grid T of random tetrahedra, i.e., each tetrahedron is randomly generated in half-open fixed-size cubical area. Those areas form a regular N^3 grid.
2. Create a regular $(N-1)^3$ grid C of cubes.
3. Align T and C such that the grid nodes of C are at the centers of the grid cells of T.
4. Unite T and C. Measure time for experiment.

In our first test series, we want to examine the generic runtime behavior if the two input objects and the output object all have similar size. We capture these properties in the TetGrid experiment and measure its runtime for values $N = 3, \ldots, 17$.

In Figure 3 we see the runtime distributed over the main subroutines. The plane sweeps, the kd-tree construction and the point location comprise a major part of the total runtime.

Two further experiments are performed in order to find out about the generic runtime behavior of the binary operations. In the first, we combine two equally sized polyhedra such that a quadratic sized polyhedron results. The runtime of this experiment is mostly determined by the kd-tree construction.

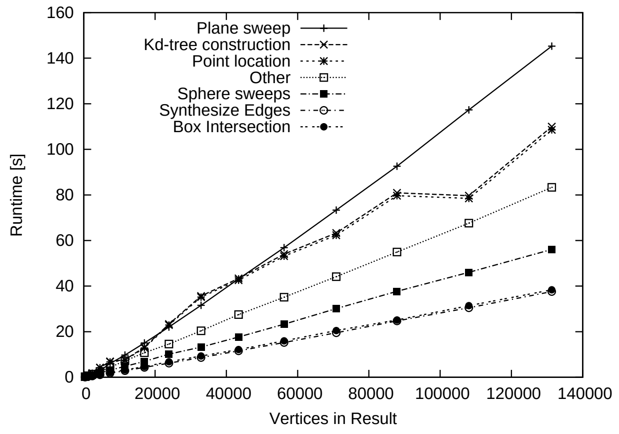


Figure 3: Runtime of the main subroutines in the TetGrid experiment.

In the other experiment, we subtract a simple polyhedron from a complex polyhedron. Here, the point location is essentially responsible for the runtime.

Furthermore, we perform several experiments in order to stress the worst case runtime behavior of the main subroutines. By this we confirm the theoretically evaluated complexity of the sphere sweep, plane sweep, point location and the ray shooting subroutine.

5 Optimizations of the Sphere Overlays

We have seen in the previous section that certain subroutines of the algorithm are very dominant. We implemented several optimizations that prevent the execution of some complex subroutines for many common and easy cases. Here, the optimization of the sphere overlays was most effective.

The sweep-line algorithm is a powerful tool; We use it in the plane for facet boundary cycles and we use it on half-spheres for the sphere maps. However, it is a comparatively costly step, although its asymptotic complexity is close to optimal.

optimizations			number of	runtime	
(i)	(ii)	(iii)	sweeps	sweeps	total
-	-	-	240470	345.12	480.34
+	-	-	14858	36.41	189.56
-	+	-	201227	301.02	437.17
-	-	+	217484	335.84	478.56
+	+	+	12880	27.67	163.65

We evaluate the contribution of the sphere sweeps to the total runtime of a binary boolean operation with a TetGrid experiment for $N = 16$. The table above lists the number of sphere sweeps performed during this operation, together with the runtime of the overlay and the total runtime. The values in the first row refer to a test run without any optimizations. The other rows refer to test runs with one or more of the following optimizations activated:

(i) The sphere sweep algorithm is used only in complex cases. In most cases, the overlay is computed by specialized algorithms.

(ii) Our sphere sweep can process one half-sphere at once. A lot of extra work has to be done to cut each sphere map into two halves and to paste the two resulting half-spheres back together. We try to get by with sweeping only one half-sphere, if possible.

(iii) For some vertices of the input polyhedra it is easy to determine that they will not appear in the resulting polyhedron. For instance, in a union operation every vertex of either polyhedron located in the inside of the other polyhedron is absorbed into this volume. We do not perform an overlay on these vertices.

6 Comparison with ACIS R13

We compare our implementation with Acis R13, a common commercial CAD kernel used in many CAD systems [5]. It should be said that we are comparing apples with oranges here. Acis is handling more general geometries and has some overhead in dispatching function calls to the specialized functions for linear geometry. On the other hand, our implementation handles Nef polyhedra in their full generality with all the potentially occurring degeneracies in the algorithms and it uses exact arithmetic to be reliable and robust.

6.1 Balanced Binary Operations

To get a general impression, we repeat the TetGrid experiment with Acis. Naturally, both algorithms perform on the same data sets.

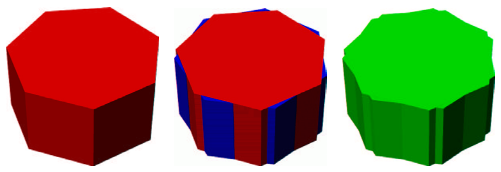
N	result vertices	runtime [s]	
		Acis R13	Nef 3D
3	352	0.29	1.01
4	1135	0.63	3.67
5	2390	1.37	8.43
6	4548	2.79	17.30
7	7383	5.29	29.20
8	11555	10.13	44.29
9	16998	14.27	70.26
10	23883	22.81	102.09
12	43418	35.58	192.12
14	70827	swapping	316.71

The table above shows that Acis is faster by a factor of 4 to 6. No obvious trend is visible.

We get quite a different result, if we subtract a simple object from a complex object. Here, the difference between Acis and our algorithm is quite pronounced with Acis being a factor of about twenty times faster. A notable difference might be in the software interface; Acis modifies the first input object to become the result, while our implementation creates the result from scratch.

6.2 Floating-Point versus Exact Arithmetic

One of the major differences between ACIS and our implementation is our use of exact arithmetic instead of floating point-arithmetic. Floating-point and interval arithmetic are the state-of-the-art in Computer Aided Design, but we are not aware of any system that uses exact arithmetic to solve the remaining cases that floating-point and interval arithmetic cannot solve. An obvious reason is the runtime cost for exact arithmetic, but also the difficulties in realizing exact and efficient solutions for more general curves and surfaces may play a role.



Experiment ROTCYLINDER

1. Create two identical right cylinder C and C', such that the base is a regular polygon with N sides parallel to the x-y plane.
2. Rotate C' around its vertical centerline by α.
3. Unite C and C'. Measure time for experiment.

We use the ROTCYLINDER experiment to demonstrate the effect of exact arithmetic; on one hand, we gain expressiveness in modeling, because we can compute results where ACIS fails very soon, and on the other hand, it shows the runtime cost for exact arithmetic, since the input coordinates grow in this series of experiments.

In this test scenario the endpoints of the intersection edges are extremely close together. Without an adequate precision it is not possible to compute an intersection point that is on both edges and different from the endpoints.

n	α	runtime ACIS R13	runtime Nef 3D
100	10^{-1}	1.08s	4.65s
	10^{-2}	1.05s	4.77s
	10^{-3}	1.08s	4.85s
	10^{-4}	1.07s	4.90s
	10^{-5}	not executable	5.03s
1000	10^{-1}	61s	93s
	10^{-2}	61s	95s
	10^{-3}	61s	97s
	10^{-4}	not executable	98s
2000	10^{-1}	252s	274s
	10^{-2}	253s	280s
	10^{-3}	255s	288s
	10^{-4}	not executable	290s
10000	10^{-7}	not executable	4433 s

The table above shows that ACIS' floating-point operations are insufficient for α smaller than 10^{-3}. On the other side, ACIS is faster, in particular for small instances. For $n = 100$ the factor of our runtime and ACIS' runtime is slightly below 5; for $n = 2000$ it is less than 1.2. Additionally, we performed a run with $n = 10000$, $\alpha = 10^{-7}$ to highlight the robustness of our arithmetic operations.

7 Conclusion

We achieved our goal of a complete, exact, correct and efficient implementation of boolean operations on a very general class of polyhedra in space. Useful extensions with applications in exact motion planning are Minkowski sums and the subdivision of the solid into simpler shapes, e.g., a trapezoidal or convex decomposition in space.

For ease of exposition, we restricted the discussion to boolean flags. Larger label sets can be treated analogously.

Nef complexes are defined by planes. It follows from the work on the *Selective Geometric Complexes* (SGC) of Rossignac and O'Connor [4] that our data structures extend immediately to complexes defined by curved surfaces. However, some of the algorithmic steps become difficult and need further work, such as the synthesis algorithm, where we need unique representations of intersection curves and where we need to sort points on intersection curves.

References

[1] K. Dobrindt, K. Mehlhorn, and M. Yvinec. A complete and efficient algorithm for the intersection of a general and a convex polyhedron. In *Proc. 3rd Work. Alg. Data Struct.*, LNCS 709, pages 314–324, 1993.

[2] M. Granados, P. Hachenberger, S. Hert, L. Kettner, K. Mehlhorn, and M. Seel. Boolean operations on 3D selective Nef complexes: Data structure, algorithms, and implementation. In *Proc. 11th Annu. Europ. Sympos. Algorithms (ESA'03)*, LNCS 2832, pages 654–666. Springer, 2003.

[3] W. Nef. *Beiträge zur Theorie der Polyeder.* Herbert Lang, Bern, 1978.

[4] J. R. Rossignac and M. A. O'Connor. SGC: A dimension-independent model for pointsets with internal structures and incomplete boundaries. In M. Wozny, J. Turner, and K. Preiss, ed., *Geom. Model. for Prod. Eng.* North-Holland, 1989.

[5] A Dassault Systèmes company Spatial Corp. *ACIS R13 Online Help*, 2004.

[6] C. Yap. Towards exact geometric computation. *Comput. Geom. Theory Appl.*, 7(1):3–23, 1997.

Ternary Blending Operations

Galina Pasko[*], Alexander Pasko[†], Tosiyasu L. Kunii[‡]

Abstract

We discuss new analytical formulations for localized and controllable blending operations in the function-based solid modeling. The blending set operations are defined using R-functions and displacement functions with the localized area of influence. The shape and location of the blend are controlled by an additional bounding solid thus turning the operation into a ternary one. We also describe a new approach to solving the problem of shape metamorphosis between k-dimensional shapes by applying space-time bounded blending to the specially constructed (k+1)-dimensional half-cylinders and making cross-sections for getting intermediate shapes under the transformation.

1 Blending in solid modeling

Blending operations in solid modeling generate smooth transitions between two or several surfaces. Blending is also considered a natural property of implicit surfaces, where the basic operation is an algebraic sum (or difference) between skeleton-based scalar fields. Blending operations are typically used in computer-aided design for modeling fillets and chamfers. These operations are usually smooth versions of set-theoretic operations on solids (intersection, union, and difference), which approximate exact results of these operations by rounding sharp edges and vertices.

The major requirements to blending operations [1] are tangency of the blend surface with the initial surfaces, automatic clipping of unwanted parts of the blending surface, C^1 continuity of the blending function everywhere in the domain, support of added and subtracted material blends. Special attention is paid to the intuitive control of the blend shape and position: the construction of the blend and its parameters should have clear geometric interpretation.

[*]IT Institute, Kanazawa Institute of Technology, gip@tokyo.com

[†]IT Institute, Kanazawa Institute of Technology and Hosei University, pasko@k.hosei.ac.jp

[‡]IT Institute, Kanazawa Institute of Technology, tosi@kunii.com

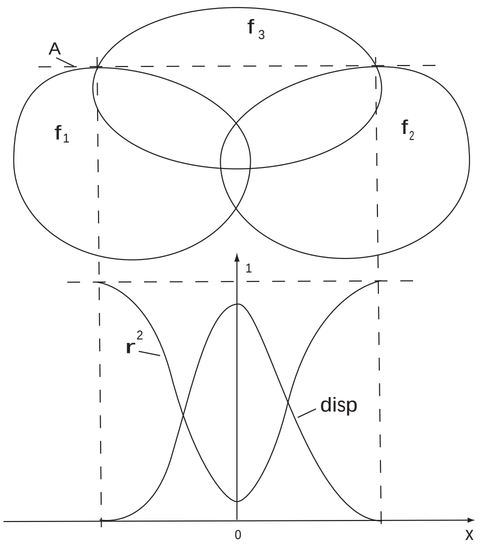


Figure 1: Components of the definition of the ternary bounded blending union operation.

2 Bounded blending

To satisfy most of the above requirements for solids exactly represented by continuous real functions (with implicit surfaces as boundaries), we introduce bounded blending operations defined using R-functions [2] and displacement functions with the localized area of influence. The shape and location of the blend is defined by an additional bounding solid thus making the ternary blending operation (having three solids as arguments).

Let two initial solids be described by the inequalities $f_1(X) \geq 0$ and $f_2(X) \geq 0$, and the bounding solid be described as $f_3(X) \geq 0$, where f_i are continuous real functions, and X is a vector of point coordinates. The bounded blending operation can be defined as

$$F_b(f_1, f_2, f_3) = R(f_1, f_2) + disp_b(f_1, f_2, f_3)$$

where R stands for an R-function defining one of the set-theoretic operations, and $disp_b$ is a displacement function. For example, a union operation can be exactly defined by the following R-function:

$$R(f_1, f_2) = f_1 + f_2 + \sqrt{{f_1}^2 + {f_2}^2}$$

The relatioships between the components of the definition of the ternary bounded blending union operation are shown in Fig. 1. In the upper part of the figure two solids to be blended (f_1 and f_2) and a bounding solid f_3 are shown. The lower part illustrates the behavior of the functions $disp_b$ and r in the cross-section A of the bounding solid.

It is required that the blending surface exists only inside the bounding solid, and only initial surfaces exist outside the bounding solid. Therefore, the displacement function $disp_b(r)$, where $r = r(f_1, f_2, f_3)$ is a generalized distance from the initial surfaces, has to satisfy the following conditions:

1) $disp_b(r) \geq 0$, it takes the maximal value for $r = 0$, and the displacement is symmetric in respect to the initial defining functions;

2) $disp_b(r) = 0, r \geq 1$, is a condition of the blend localization inside the bounding solid;

3) $\partial disp_b/\partial r = 0, r = 1$, means the curve $disp_b(r)$ tangentially approaches the horizontal axis at $r = 1$ and accordingly the blend tangentially approaches initial surfaces.

There are many different functions satisfying the above requirements. Here, we use the polynomial function of lowest order:

$$disp_b(r) = 2r^3 - 3r^2 + 1$$

A different displacement function and details of the generalized distance $r(f_1, f_2, f_3)$ function construction can be found in [3].

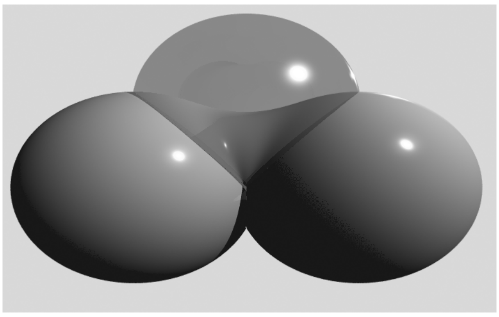


Figure 2: Ternary blending operation: two ellipsoids to be blended and the boundung ellipsoid (transparent).

Let us illustrate such properties of the proposed bounded blending operation as its local character and intuitive control of blend shape and position. In Fig. 2 the pure union of two ellipsoids is changed to the bounded blending union using the third ellipsoid (transparent shape). The resulting blend is located strictly inside the bounding ellipsoid, which produces an unusual blending shape localized at the top part of the initial union of ellipsoids. At the next step (Fig. 3), we increase the size of the bounding ellipsoid and correspondingly change the shape of the blend, which stretches out to the lower part of the initial shape.

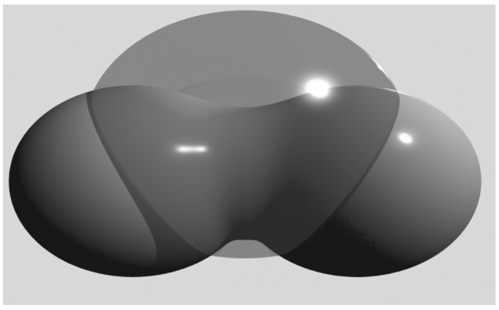


Figure 3: Control of blending by changing the bounding solid.

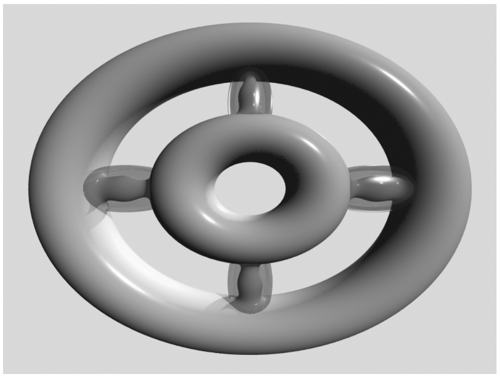


Figure 4: Multiple blending controlled by a bounding solid consisting of four disjoint components.

The definition of the bounding solid by a single function allows for unusual operations such as multiple blending. As it is shown in Fig. 4, the bounding solid can be constructed using arbitrary primitives and operations. In this example, the bounding solid controls the blending union of two non-intersecting tori. The bounding solid is described using R-functions by a single function f_3 as union of four ellipsoids. The result of the bounded blending operation is a single connected solid with multiple blending components located inside the disjoint components of the bounding solid.

The proposed bounded blending operations can replace pure set-theoretic operations in the construction of a solid without rebuilding the entire construction tree data structure. Note that the ternary bounded blending operations have three solids as their arguments and hence require ternary nodes in the construction tree in comparison with binary nodes typical for the traditional Constructive Solid Geometry (CSG).

3 Space-time blending

Shape transformation between given objects (metamorphosis) is one of typical space-time modeling operations. The existing approaches to metamorphosis are based on one or several of the following assumptions: equivalent topology (mainly topological disks or balls are considered), polygonal shape representation, shape alignment (shapes have common coordinate origin and significantly overlap), possibility of shape matching (establishing of shape vertex-vertex, control points or other features correspondence), the resulting transformation should be close to the motion of an articulated figure.

Linear interpolation between functionally defined shapes have proven to solve some of the above problems for computer animation and artistic applications. The problem which remains open is a transformation between non-overlapping shapes, which combines metamorphosis and non-linear motion. We develop a new approach to shape metamorphosis using blending operations in space-time. The key steps of the metamorphosis algorithm are: dimension increase by converting two input kD shapes into half-cylinders in $(k+1)D$ space-time, applying bounded blending union to the half-cylinders, and making cross-sections for getting intermediate shapes [4].

The bounded space-time blending procedure for 2D shapes consists of the following steps:
1) two initial 2D shapes are given on a 2D plane;
2) each shape is considered as a 2D cross-section of a half-cylinder (a semi-infinite cylinder bounded by a plane from one side along the time axis) defined in 3D space-time;
3) two half-cylinders are placed at some distance along time axis to provide a time interval for making the blend;
4) the bounded blending union operation with added material is applied to the 3D half-cylinders with two planar half-spaces orthogonal to the time axis forming a bounding 3D object (a slab between two planes);
5) consecutive cross-sections of the blend along the time axis are combined into a 2D animation.

As no critical assumptions were made in the proposed approach about the dimensionality of the initial shapes, we can apply it to 3D objects. In this case, each shape is considered as a 3D cross-section of a half-cylinder defined in 4D space-time. Note that in both 2D and 3D cases topological changes of objects are handled automatically as shown in Fig. 5.

4 Conclusion

We proposed an original solution for a long-standing problem of the blend localization and control. The main idea is to apply localized displacements to the standard R-functions describing pure set-theoretic operations. This allows for support of several unusual operations such as multiple blending or partial edge blending, which hardly can be supported by other modeling techniques.

We discribed a new approach to shape metamorphosis on the basis of the dimension increase, bounded blending between higher-dimensional space-time objects, and cross-sectioning the blend area for getting frames of the animation. The space-time blending operation with such simple bounds as two planes satisfies the condition of the localization of the shape transformation at some predefined time interval. The proposed approach can handle non-overlapping 2D and 3D shapes with arbitrary topology.

The described bounded blending operations have three objects as their arguments. This brings a new requirement for a functionally based modeling system to support n-ary nodes in the construction tree.

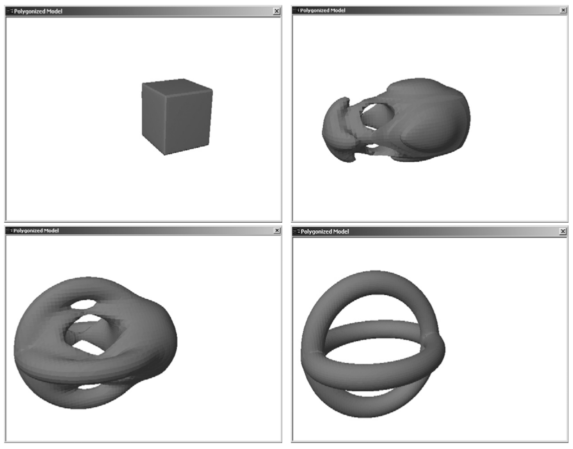


Figure 5: Metamorphosis of a cube into the union of two tori using space-time bounded blending.

References

[1] J. Woodwark. Blends in geometric modeling. In *Proc. The Mathematics of Surfaces II*, pages 255–297., Clarendon Press, 1987.

[2] V. Rvachev. *Theory of R-functions and Some Applications*, Naukova Dumka, Kiev, USSR, 1987 (in Russian).

[3] G. Pasko, A. Pasko, T. L. Kunii. Bounded blending for the function-based shape modeling. *IEEE Computer Graphics and Applications*, 2005 (to appear)

[4] G. Pasko, A. Pasko A., T. L. Kunii. Space-time blending, *Journal of Computer Animation and Virtual Worlds*, vol. 15, No. 2, pages 109–121, John Wiley, 2004.

Algebraic Study of the Apollonius Circle of Three Ellipses

Ioannis Z. Emiris[*] George M. Tzoumas[*]

Abstract

We study the external tritangent Apollonius (or Voronoi) circle to three ellipses. This problem arises when one wishes to compute the Apollonius (or Voronoi) diagram of a set of ellipses, but is also of independent interest in enumerative geometry. This paper is restricted to non-intersecting ellipses, but the extension to arbitrary ellipses is possible.

We propose an efficient representation of the distance between a point and an ellipse by considering a parametric circle tangent to an ellipse. The distance of its center to the ellipse is expressed by requiring that their characteristic polynomial have at least one multiple real root. We study the complexity of the tritangent Apollonius circle problem, using the above representation for the distance, as well as sparse (or toric) elimination. We offer the first nontrivial upper bound on the number of tritangent circles, namely 184.

Keywords: Voronoi diagram, ellipse, mixed volume, Euclidean distance, resultant.

1 Introduction

Voronoi diagrams are interesting constructs with numerous applications and have been studied extensively. However, the bulk of the existing work in the plane concerns point-sites or linear sites such as segments and polygons. More recently, some works have extended Apollonius (or Voronoi) diagrams to the case of circles, e.g. [8]. For the latter problem, the implementation of [3] is now part of the CGAL library. Recent works derive (semi-)algebraic conditions for characterizing the relative position of conics in the plane or certain quadrics in space, e.g. [12], [5].

Our ultimate goal is to compute efficiently and exactly the Apollonius (or Voronoi) diagram of arbitrary sets of ellipses in the plane, under the Euclidean metric. We assume that the ellipses are given algebraically, or implicitly. This is clearly a harder problem than the diagram of circles or the visibility map among ellipses, hence the need for higher degree algebraic operations. As a first step, this paper studies the case of *non-intersecting* ellipses, although our methods readily extend to arbitrary inputs.

The algorithms for the Apollonius diagram of ellipses typically use the following 2 main predicates. Further predicates are examined in [4].

(1) given two ellipses and a point outside of both, decide which is the ellipse closest to the point, under the Euclidean metric

(2) given 4 ellipses, decide the relative position of the fourth one with respect to the external tritangent Apollonius circle of the first three

For predicate (1) we consider a circle, centered at the point, with unknown radius, which corresponds to the distance to be compared. A tangency point between the circle and the ellipse exists iff the discriminant of the corresponding pencil's determinant vanishes. Hence we arrive at a method using algebraic numbers of degree 4, which is optimal. Note that we avoid expressing the coordinates of the tangency point.

Let us focus on predicate (2). In the case of 3 disks, the number of tritangent circles is 8 and the corresponding predicate is of algebraic degree 2 [3]. This problem is also known as *The circle of Apollonius*, because it was first addressed by *Apollonius of Perga*, in about 250 BC. While this has been known since antiquity, the generalization to ellipses is yet to be solved efficiently.

Even the number of tritangent circles to 3 ellipses is not known. The problem involves equations of high degree and obtaining an exact solution is nontrivial. [10] attempts to deal with this problem, but exact computation with the proposed method is not completed and the author reverts to numerical methods. No bounds on the complexity are given, nor on the number of tritangent circles.

We apply the method from predicate (1) and recent advances in sparse (or toric) resultants in order to project all common roots to those of a univariate equation. This leads to the first interesting bound on the number of tritangent circles, namely 184. Mixed volume also gives this bound as does a real algebraic geometry argument [11].

The paper is organized as follows. The next section introduces some of the ellipse's properties. In section 3 we describe an efficient representation for the Euclidean distance between a point and an ellipse and we apply this idea to predicate (1). Finally, section 4 deals directly with the external tritangent Apollonius circle and predicate (2).

[*]Department of Informatics and Telecommunications, National University of Athens, emiris@di.uoa.gr, g.tzoymas@di.uoa.gr

2 The geometry of an ellipse

An *ellipse* is the locus of points in the plane the sum of whose distances from the foci is $2\alpha > 0$. The foci lie at distance 2γ. The length of the major and minor axes are $2\alpha, 2\beta$, resp. where $\beta^2 = \alpha^2 - \gamma^2$. Let (x_c, y_c) be its center and u the angle between the major and the x axes. When $x_c = y_c = u = 0$ the ellipse is in *orthogonal* position, otherwise, it is in *generic* position. The ellipse is a *conic* section with equation:

$$E(x,y) := ax^2 + 2bxy + cy^2 + 2dx + 2ey + f, \quad (1)$$

where $J_2 > 0 \neq ac$ and J_2 is defined below. The ellipse's parameters (a,b,c,d,e,f) are related to its center, rotation, axes and focal distance, but we omit the corresponding equations. The following quantities are *invariants* under rotation and translation:

$$J_1 = a+c = \alpha^2+\beta^2 > 0, \quad J_2 = \begin{vmatrix} a & b \\ b & c \end{vmatrix} = \alpha^2\beta^2 > 0,$$

$$J_3 = \begin{vmatrix} a & b & d \\ b & c & e \\ d & e & f \end{vmatrix} = -J_2^2 < 0.$$

The following quantity is invariant under rotation; its expression uses the lemma below.

$$J_4 = (a+c)f - d^2 - e^2 = J_2(x_c^2 + y_c^2 - J_1).$$

Let L_y, L_x be the lines connecting the leftmost and rightmost points, and the highest with the lowest point, respectively.

Lemma 1 *Consider an ellipse of the form (1). Its center is rational and coincides with the intersection of L_x, L_y, where $x_c = (be-dc)/J_2$, $y_c = (bd-ae)/J_2$.*

Given a point V outside an ellipse, how many normals are there to the ellipse? Let us count normal *segments*, defined as the segment of a line normal to the ellipse at some point Q; the segment's endpoints are Q, V. The boundary of the regions where the number of normals changes is the *evolute*, which is a stretched astroid (see figure 1). For an ellipse in orthogonal position, each point (x,y) on the evolute satisfies: $(\alpha x)^{\frac{2}{3}} + (\beta y)^{\frac{2}{3}} = \gamma^{\frac{4}{3}}$.

Proposition 2 *There are 4, 3 or 2 normals of a point to an ellipse, depending on whether the point lies inside the evolute, lies on the evolute but not at a cusp or, respectively, the point is a cusp or outside the evolute.*

This yields a lower bound on the algebraic complexity of computing the distance of an external point to the ellipse, since any condition on the unknown distance has degree ≥ 4.

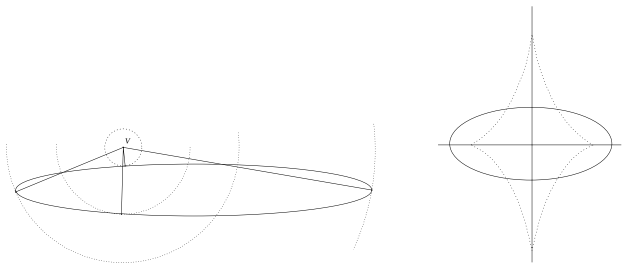


Figure 1: Left: an example of a point with 4 normals. Right: the evolute of an ellipse.

3 Distance between point and ellipse

Consider an ellipse E and a point $V = (v_1, v_2)$ outside E. Let C be a circle centered at V with radius equal to $\sqrt{s}$, for a real $s > 0$. We shall express the *Euclidean distance* $\delta(V,E)$ between V and E by the *smallest* positive value of $\sqrt{s}$ for which C is tangent to E. In comparing distances, it is sufficient to consider squared distance s.

It would be possible to find all tangency points by solving the system:

$$E(x,y) \ = \ \det[\nabla E(x,y), (x,y) - V] \ = \ 0, \qquad (2)$$

and then choosing the appropriate solution, where the 2nd equation constraints the vector $(x,y) - V$ to be normal to E at (x,y). Consider system (2) with an additional equation: $\|(x,y) - V\|^2 = s$. The resultant of the 3 polynomials with respect to x, y is precisely polynomial $\Delta(v_1, v_2, s)$ to be defined below by an alternative manner. It is the algebraic representation of the offset curve to E at distance s.

Our goal is to avoid explicit computation of the tangency points by requiring that system $E = C = 0$ have a multiple root. To arrive at a simple polynomial we apply the theory of characteristic polynomials and pencils [9, 12]. Let us express a conic as $[x,y,1]M[x,y,1]^T$, for an appropriate matrix M. Then E, C correspond to

$$A = \begin{bmatrix} a & b & d \\ b & c & e \\ d & e & f \end{bmatrix}, B = \begin{bmatrix} 1 & 0 & -v_1 \\ 0 & 1 & -v_2 \\ -v_1 & -v_2 & v_1^2 + v_2^2 - s \end{bmatrix}$$

The *pencil* of E and C is $\lambda A + B$, and their characteristic polynomial is

$$\begin{aligned} \phi(\lambda) &= |\lambda A + B| = J_2^2\lambda^3 + c_2\lambda^2 + c_1\lambda + s, \\ c_2(s) &= J_2 s - T(v_1, v_2), \\ c_1(s) &= J_1 s - E(v_1, v_2), \\ T(v_1, v_2) &= J_2[(v_1 - x_c)^2 + (v_2 - y_c)^2 - J_1]. \end{aligned}$$

The discriminant $\Delta(s)$ of $\phi(\lambda)$ is of degree 4:

$$\begin{aligned}\Delta(s) = J_2^2(J_1^2 - 4J_2)\, s^4 + \\ 2J_2(9J_1J_2^2 - J_1^2T + 6J_2T - 2J_1^3J_2 - J_1J_2E)\, s^3 + \\ +(-18J_2^3E + 4J_1J_2ET - 27J_2^4 + J_1^2T^2 - \\ -18J_1J_2^2T + J_2^2E^2 + 12J_1^2J_2^2E - 12J_2T^2)\, s^2 + \\ 2(2T^3 - J_1ET^2 - 6J_1J_2^2E^2 + 9J_2^2ET - J_2E^2T)\, s \\ +E^2(T^2 + 4J_2^2E).\end{aligned}$$

The relative position of a circle and an ellipse falls into one of 9 cases, related to the multiplicity and signs of the real roots of $\phi(\lambda)$ [12, thm.8]. When $\phi(\lambda)$ has at least one multiple root, the ellipse and the circle have at least one tangency point. Note that $\phi(\lambda)$ has at least one negative root because the product of roots equals $-s < 0$.

By picking the smallest positive root of $\Delta(s) = 0$, we assure that $\phi(\lambda)$ has at least one root with multiplicity greater than one. Assume that the circle centered at V grows until it touches E (and then it might continue to grow until it fully contains E). Since V is outside E, the smallest positive root of $\Delta(s)$ corresponds to $\delta(V, E)$.

Proposition 3 *Given an ellipse E and a point V outside E, $\delta(V, E)$ is the square-root of the smallest positive zero of $\Delta(s)$; the latter is a univariate polynomial of degree 4. The degree of the coefficients of $\Delta(s)$ is 6, 8, 10, 12, and 14, in order of decreasing power in s, in v_1, v_2 and the parameters of E. In case (v_1, v_2) is the center of another ellipse E' the degree of the coefficients of $\Delta(s)$ is exactly 22 in the parameters of E, E'.*

Corollary 4 *Given ellipses E_1, E_2 and point V outside both of them, we can decide which ellipse is closest to V by comparing two algebraic numbers of degree 4. The previous section implies that this degree is optimal.*

4 External tritangent circle

Given 3 ellipses in the form of equation (1) we want to find an external tritangent circle, as shown in fig. 2. Eventually, we are interested in deciding on the relative position of a fourth ellipse and the circle. An important open question is: what is the maximum number of real tritangent circles to 3 ellipses? Given the discussion in section 2, we expect that there are at least $4^3 = 64$ such circles.

Let $\sqrt{s}$ be the radius of the tritangent circle and (v_1, v_2) its center. Using the discriminant as above for each of the 3 ellipses, we get

$$\Delta_1(v_1, v_2, s) = \Delta_2(v_1, v_2, s) = \Delta_3(v_1, v_2, s) = 0. \quad (3)$$

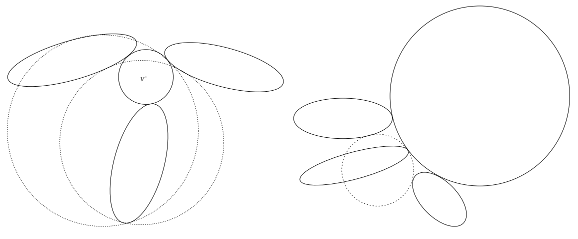


Figure 2: Tritangent circles to 3 ellipses; only one is externally tangent

Among the solutions of this system, the external tritangent circle of interest may or may *not* have the smallest radius; cf. the respective cases in figure 2.

We apply sparse (or toric) elimination theory, using the properties of the resultant and the mixed volume. Given a system of $n+1$ polynomials f_i in n variables, with coefficients c_{ij}, the *resultant* of these polynomials is a new polynomial $R \in \mathbb{Z}[c_{ij}]$ such that when c_{ij} are specialized, $R = 0 \iff \exists a : \forall i\ f_i(a) = 0$. If the roots a lie in a projective (resp. toric) variety, then we refer to the projective (resp. toric) resultant. Given n polynomials in n variables, the *mixed volume* of this polynomial system is a function of the support (Newton polytope) of each polynomial. The mixed volume provides an upper bound on the roots of the system in $(\mathbb{C}^*)^n$. For more information see [2].

Each Δ_i is of total degree 8 in v_1, v_2, s and 4 in s. The mixed volume of system (3) is 256, which is too high. It is known that this bound may not be tight, as it may count complex roots and roots at "infinity".

Indeed, it is possible to reduce the mixed volume of the above system, in order to obtain a better upper bound. We set

$$q := v_1^2 + v_2^2 - s \quad (4)$$

In this case, the matrix B defined in the previous section, contains only linear terms with respect to v_1, v_2, q. The discriminant of the characteristic polynomial is of total degree 6 in v_1, v_2, q and 4 in q; the coefficients of $1, q, q^2, q^3, q^4$ are polynomials in v_1, v_2 of degree 6,5,4,2,1 respectively. The corresponding system has mixed volume 184. Note that solving for v_1, v_2, s requires the use of equation (4). The mixed volume of the system of Δ_i with this additional equation, with respect to v_1, v_2, s, q, is still 184.

Recent advances in matrix formulae for the resultant allow us to compute the resultant of certain systems of 3 bivariate polynomials as a single determinant. One class of such systems are those with identical supports [7]. The corresponding matrix is of hybrid type, i.e., it contains blocks of Sylvester and of Bézout type. The matrix construction has been implemented in Maple by A. Khetan. Our system (3) falls in this class, considered with variables v_1, v_2, after hiding q in the coefficients. Its resultant is a polynomial in q and equals, generically, the determinant of a 58×58 matrix. We denote this matrix by K.

To get an idea of the quantities involved, we have studied a specific example of three ellipses, in random position as in the left-hand side figure 2. The input parameters are signed 10-bit integers. The elements of K are either 0 or polynomials in q of degree 0–10. The computation of the determinant of K is done by interpolation. The determinant of K is a polynomial in q, which we denote by $d(q)$. By substituting different values of q into K we eliminate all indeterminates making the computation of the determinant a trivial task. By making 200 such replacements (in fact, 185 suffice) we obtain 200 pairs of $\langle q, d(q)\rangle$. It turns out that there is a unique interpolating polynomial of degree 184 in q through these values which is exactly the resultant of our example. Hence, in this example the number of complex solutions matches the upper bound given by the mixed volume.

The coefficients of this resultant are, on average, 1385-digit (4603-bit) integers. We 've not yet managed to solve this resultant efficiently and exactly. However, as a preliminary approach we have applied the Aberth method (implemented in [1]) to solve the polynomial numerically. This algorithm yielded 8 real roots in less than a second.

According to [11], there are 184 complex circles in the worst case that are tangent to 3 given conics in the plane. The idea is to consider a manifold (space of complete conics) whose cohomology ring ([6]) has two generators:

$$\begin{aligned} p &:= \text{a conic contains a fixed, but general point} \\ l &:= \text{a conic is tangent to a fixed, but general line} \end{aligned}$$

In this ring, conjunction of conditions is multiplication, and every degree-5 monomial is associated to an integer. For example $p^4 l \mapsto 2$ meaning that there are 2 conics through 4 points and tangent to a line. The condition of tangency to a conic is $2(p+l)$ and a circle contains the two circulars points at infinity $(x_1 : x_2 : x_0) = (i : \pm 1 : 0)$. Thus the expression $p^2[2(p+l)]^3$ maps to 184 which means that there are at most 184 complex circles tangent to 3 conics in the plane. An open question is how many of these circles can be real.

The above computation of the mixed volume, the resultant, and the upper bound give strong indication that the system (3) modified with (4) is optimal. Methods on how to solve system (3) efficiently will be addressed in a future work. In such an approach, one might consider semi-algebraic constraints such as those in [4], in order to prune cases in some subdivision-based algorithm.

Acknowledgements

We acknowledge partial support by PYTHAGORAS, project 70/3/7392 under the EPEAEK program of the Greek Ministry of Educational and Religious Affairs, and IST Program of the EU as a Shared-cost RTD (FET Open) Project under Contract No. IST–2005 (ACS – Algorithms for Convex Shapes).

References

[1] D.A. Bini and G. Fiorentino. Design, analysis, and implementation of a multiprecision polynomial rootfinder. *Numerical Algorithms*, 23:127–173, 2000.

[2] D. Cox, J. Little, and D. O'Shea. *Using Algebraic Geometry.* Number 185 in Graduate Texts in Mathematics. Springer-Verlag, New York, 1998.

[3] I.Z. Emiris and M.I. Karavelas. The predicates of the Apollonius diagram: algorithmic analysis and implementation. *Comp. Geom.: Theory & Appl., Spec. Issue*, 2005. To appear.

[4] I.Z. Emiris and G.M. Tzoumas. Distance predicates among ellipses. Manuscript. www.di.uoa.gr/∼grad0491/msc/DistPredEll.pdf, 2004.

[5] F. Etayo, L. Gonzalez-Vega, and N. del Rio. A new approach for characterising the relative position of two ellipses depending on one or several parameters. 2004. Submitted.

[6] J. Harris. *Algebraic Geometry (A First Course).* Springer-Verlag, Berlin, Germany, 1992.

[7] A. Khetan. The resultant of an unmixed bivariate system. *J. Symb. Comput.*, 36:425–442, 2003.

[8] D.-S. Kim, D. Kim, and K. Sugihara. Voronoi diagram of a circle set from Voronoi diagram of a point set: II. Geometry. *CAGD*, 18:563–585, 2001.

[9] H. Levy. *Projective and Related Geometries.* McMillan Co., 1967.

[10] R. Lewis and S. Bridget. Polynomial equations arising from apollonius problems in biochemistry and pharmacology. In *Proceedings of Applications of Computer Algebra*, 2001.

[11] F. Sottile. Personal communication. 2004.

[12] W. Wang, J. Wang, and M. Kim. An algebraic condition for the separation of two ellipsoids. *Comp. Aided Geom. Design*, 18:531–539, 2001.

The Visibility–Voronoi Complex and its Applications[*]

Ron Wein[†] Jur P. van den Berg[‡] Dan Halperin[§]

Abstract

We introduce a new type of diagram called the VV$^{(c)}$-diagram (the Visibility–Voronoi diagram for clearance c), which is a hybrid between the visibility graph and the Voronoi diagram of polygons in the plane. This diagram can be used for planning natural-looking paths for a robot translating amidst polygonal obstacles in the plane. We also propose an algorithm that is capable of preprocessing a scene of configuration-space polygonal obstacles and constructs a data structure called the VV-complex. The VV-complex can be used to efficiently plan motion paths for any start and goal configuration and *any clearance value* c, without having to explicitly construct the VV$^{(c)}$-diagram for that c-value. We have implemented a CGAL-based software package for computing the VV$^{(c)}$-diagram in an *exact* manner for a given clearance value, and used it to plan natural-looking paths in various applications.

1 Introduction

We study the problem of planning a natural-looking collision-free path for a robot with two degrees of motion freedom moving in the plane among polygonal obstacles. By "natural-looking" we mean that (a) the path should be *short* — that is, it should not contain long detours when significantly shorter routes are possible; (b) it should have a guaranteed amount of *clearance* — that is, the distance of any point on the path to the closest obstacle should not be lower than some prescribed value; and (c) it should be *smooth*, not containing any sharp turns. Requirements (b) and (c) may conflict with requirement (a) in case it is possible to considerably shorten the path by taking shortcuts through narrow passages. In such cases we may prefer a path with less clearance (and perhaps containing sharp turns).

The *visibility graph* [2, Chap. 15] is a well-known data structure for computing the shortest collision-free path between a start and a goal configuration. However, shortest paths are in general tangent to obstacles, so a path computed from a visibility graph usually contains semi-free configurations and therefore does not have any clearance. This not only looks unnatural, it is also unacceptable for many motion-planning applications. On the other hand, planning motion paths using the *Voronoi diagram* of the obstacles [4] yields a path with maximal clearance, but this path may be significantly longer than the shortest path possible, and may also contain sharp turns.

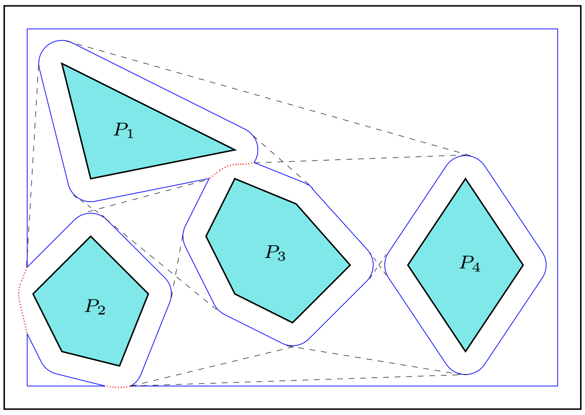


Figure 1: The VV$^{(c)}$-diagram for four convex obstacles located in a rectangular room. The boundary of the union of the dilated obstacles is drawn in a solid line, the relevant portion of the Voronoi diagram is dotted. The visibility edges are drawn using a dashed line.

We suggest a hybrid of these two approaches, called the *VV$^{(c)}$-diagram* (the Visibility–Voronoi diagram for clearance c), yielding natural-looking motion paths, meeting all three criteria mentioned above. It evolves from the visibility graph to the Voronoi diagram as c grows from 0 to ∞, where c is the preferred amount of clearance. Beside the straightforward algorithm for constructing the VV$^{(c)}$-diagram for a given clearance value c, we also propose an algorithm for preprocessing a scene of configuration-space polygonal obstacles and constructing a data structure called the *VV-complex*. The VV-complex can be used to efficiently plan motion paths for any start and goal configuration and any given clearance value c, without having to explicitly construct the VV$^{(c)}$-diagram for that c-value.

2 The VV$^{(c)}$-Diagram

Let $\mathcal{P} = \{P_1, \ldots, P_m\}$ be a set of pairwise interior-disjoint polygons having n vertices in total, representing two-dimensional configuration-space obsta-

[*]This work has been supported in part by the IST Programme of the EU as Shared-cost RTD (FET Open) Project under Contract No IST-2001-39250 (MOVIE — Motion Planning in Virtual Environments), by The Israel Science Foundation founded by the Israel Academy of Sciences and Humanities (Center for Geometric Computing and its Applications), and by the Hermann Minkowski–Minerva Center for Geometry at Tel Aviv University.

[†]Tel-Aviv University, wein@tau.ac.il
[‡]Utrecht University, berg@cs.uu.nl
[§]Tel-Aviv University, danha@tau.ac.il

cles. Given a start configuration, a goal configuration and a preferred clearance value $c > 0$, we wish to find a shortest path between the query configurations, keeping a clearance of at least c from the obstacles where possible, but allowing to get closer to the obstacles in narrow passages when it is possible to make considerable shortcuts.

We begin by dilating each obstacle by c, by computing the Minkowski sum of each polygon with a disc of radius c. The visibility graph of the dilated obstacles contains all shortest paths with a clearance of at least c from the obstacles. Moreover, as each convex polygon vertex becomes a circular arc of radius c, the valid visibility edges are bitangents to two circular arcs (note that the dilated polygon edges are also valid visibility edges). This guarantees that a shortest path extracted from such a visibility graph is $\mathcal{C}_1$-smooth, containing no sharp turns. The only disadvantage in this approach is that narrow, yet collision-free, passages can be blocked when we dilate the obstacles (for example, in Figure 1 there exists such a narrow passage between P_1 and P_3). It is clearly not possible to pass in such passages with a clearance of at least c, but we still wish to allow a path with the maximal clearance possible in this region. To do this, we compute the portions of the free configuration space that are contained in at least two dilated obstacles, and add their intersection with the Voronoi diagram of the original polygons to our diagram. The relevant portions of the Voronoi diagram are connected to the reflex vertices in the union of the dilated boundaries, which we refer to as *chain points*. The resulting structure is called the $\mathrm{VV}^{(c)}$-diagram, and it is easy to show that it can be constructed in $O(n^2 \log n)$ time.

In case our polygons are not convex, we decompose them to obtain a set of convex polygons and compute the boundary of the dilated obstacles for this set. Note that not every reflex vertex of the boundary is now a chain point, as it can also be induced by a reflex vertex of the original polygon. Such reflex vertices are not taken into account in the $\mathrm{VV}^{(c)}$-diagram.

Given a start and a goal configuration we just have to connect them to our $\mathrm{VV}^{(c)}$-diagram and compute the shortest path between them using Dijkstra's algorithm. To this end, we have to associate a weight with each diagram edge. The weight of a visibility edge can simply be equal to its length, while for Voronoi edges we may add some penalty to the edge length, taking into account its clearance value, which is below the preferred c-value. It should be noted that if a path contains a portion of the Voronoi diagram it may not be smooth any more. This is however acceptable, as we consider making sharp turns inside narrow passages to be natural.

We have implemented an extension package of CGAL [1] that robustly computes the $\mathrm{VV}^{(c)}$-diagram of a set of rational polygons, utilizing — among other CGAL packages — the segment Voronoi diagram package by Karavelas [3]. As we wish to obtain an exact representation of the $\mathrm{VV}^{(c)}$-diagram, we may spend some time on the diagram construction, especially if it contains chain points, which are algebraically more difficult to handle. For example, the construction of the $\mathrm{VV}^{(c)}$-diagram depicted in Figure 1 takes about 10 seconds (running a Pentium IV 2 GHz machine with 512 MB of RAM). However, once the $\mathrm{VV}^{(c)}$-diagram is constructed, it is possible to use a floating-point approximation of the edge lengths to speed up the time needed for answering motion-planning queries, so that the average query time is only a few milliseconds.

3 The VV-Complex

The construction of the $\mathrm{VV}^{(c)}$-diagram for a given c-value is straightforward, yet it requires some non-trivial geometric and algebraic operations that should be computed in a robust manner — see the full version of this paper [5] for the details. Moreover, if we wish to plan motion paths for different c-values and select the best one (according to some criterion), we must construct the $\mathrm{VV}^{(c)}$-diagram for each c-value from scratch. In this section we explain how to efficiently preprocess an input set of polygonal obstacles and construct a data structure called the VV-complex, which can be queried to produce a natural-looking path for every start and goal configuration and for *any* preferred clearance value c.

Let us examine what happens to the $\mathrm{VV}^{(c)}$-diagram as c continuously changes from zero to infinity. For simplicity, we consider only convex obstacles in this section. As we mentioned before, $\mathrm{VV}^{(0)}$ is the visibility graph of the original obstacles, while $\mathrm{VV}^{(\infty)}$ is their Voronoi diagram, so as c grows visibility edges disappear from $\mathrm{VV}^{(c)}$ and make way to Voronoi chains. We start with a set of visibility edges containing all pairs of the polygonal obstacle vertices that are mutually visible, regardless whether these edges are bitangents of the obstacles.[1] We also include the original obstacle edges in this set, and treat them as visibility edges between two neighboring polygon vertices. Furthermore, we treat our visibility edges as directed, such that if the vertex u "sees" the vertex v, we will have two directed visibility edges $\vec{uv}$ and $\vec{vu}$.

As c grows larger than zero, each of the *original* visibility edges potentially spawns as many as four bitangent visibility edges. These edges are the bitangents to the circles $B_c(u)$ and $B_c(v)$ (where $B_r(p)$ denotes a circle centered at p whose radius is r) that we name $\vec{uv}_{ll}$, $\vec{uv}_{lr}$, $\vec{uv}_{rl}$ and $\vec{uv}_{rr}$, according to the relative position (left or right) of the bitangent with respect to u and to v (see Figure 2(a)). The two bitangents $\vec{uv}_{ll}$

[1] Visibility edges are only *valid* when they are bitangents, otherwise they do not contribute to shortest paths in the visibility graph. However, as c grows larger these edges may become bitangents, so we need them in our data structure.

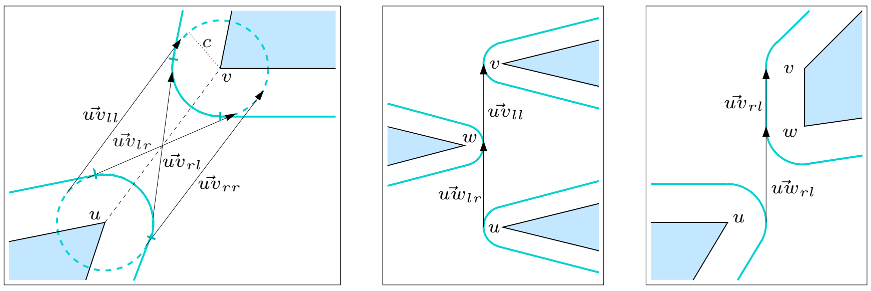


Figure 2: (a) The four possible bitangents to the circles $B_c(u)$ and $B_c(v)$ of radius c centered at two obstacle vertices u and v. Notice that in this specific scenario only the bitangent $\vec{uv}_{rl}$ is a valid visibility edge.
Visibility events involving u, v and w: (b) The dilated vertex w blocks the visibility of u and v. (c) As $\vec{uw}_{rl}$ becomes equally sloped with $\vec{uv}_{rl}$ (where vw is an obstacle edge), it becomes a valid visibility edge.

and $\vec{uv}_{rr}$ retain the same slope, while the slopes of the other two bitangents change for increasing c-values.

Note that for a given c-value, it is impossible that all four edges are valid. Our goal is to compute a *validity range* $R(e) = [c_{\min}(e), c_{\max}(e)]$ for each edge e, such that e is part of the $\mathrm{VV}^{(c)}$-diagram for each $c \in R(e)$. These validity ranges will be stored in an interval tree, so it is easy to obtain all valid edges for any given c-value. If an edge is valid, then it must be tangent to both circular arcs associated with its end-vertices. There are several reasons for an edge to change its validity status: (a) The tangency point of e to either $B_c(u)$ or to $B_c(v)$ leaves one of the respective circular arcs; (b) The tangency point of e to either $B_c(u)$ or to $B_c(v)$ enters one of the respective circular arcs; (c) The visibility edge becomes blocked by the interior of a dilated obstacle.

The important observation is that at the moment that a visibility edge $\vec{uv}$ gets blocked, it becomes tangent to another dilated obstacle vertex w, so essentially one of the edges associated with $\vec{uv}$ becomes equally sloped with one of the edges associated with $\vec{uw}$ (see Figure 2(b)). The first two cases mentioned above can be realized as events of the same nature, as they occur when one of the $\vec{uv}$ edges becomes equally sloped with $\vec{uw}_{lr}$ (or $\vec{uw}_{rl}$), when v and w are neighboring vertices in a polygonal obstacle — see Figure 2(c).

This observation stands at the basis of the algorithm we devise for constructing the VV-complex: We sweep through increasing c-values, stopping at critical *visibility events*, which occur when two edges become equally sloped. We note that the edge $\vec{uv}_{ll}$ (or $\vec{uv}_{lr}$) can only have events with arcs of the form $\vec{uw}_{ll}$ or $\vec{uw}_{lr}$, while the edge $\vec{uv}_{rl}$ (or $\vec{uv}_{rr}$) can only have events with arcs of the form $\vec{uw}_{rl}$ or $\vec{uw}_{rr}$. Hence, we associate two circular lists $\mathcal{L}_l(u)$ and $\mathcal{L}_r(u)$ of the left and right edges of the vertex u, respectively, both sorted by the slopes of the edges. Two edges participate in an event at some c-value only if they are neighbors in the list for infinitesimally smaller c. At these event points, we should update the validity range of the edges involved, and also update the adjacencies in their appropriate lists, resulting in new events.

As mentioned in Section 2, an endpoint of a visibility edge in the $\mathrm{VV}^{(c)}$-diagram may also be a chain point, so we must consider chain points in our algorithm as well. We therefore compute the Voronoi diagram of the polygonal obstacles, which is comprised of *Voronoi chains* that separate between neighboring Voronoi cells. The chains are sequences of *Voronoi arcs*, which are either line segments or circular arcs, and their endpoints are called *Voronoi vertices*. As a Voronoi chain is either *monotone* or has a single point with minimal clearance, we can associate at most two chain points with every Voronoi chain. Our algorithm will also have to compute the validity range for edges connecting a chain point with a dilated vertex or with another chain point. For that purpose, we will have a list $\mathcal{L}(p)$ of the outgoing edges of each chain point p, sorted by their slopes (notice that we do not have to separate the "left" edges from the "right" edges in this case).

3.1 The Preprocessing Stage

Given an input set $P_1, \ldots, P_m$ of polygonal obstacles as described above, we start by computing their visibility graph and classifying the visibility edges as valid (bitangent) or invalid. We examine each bitangent visibility edge uv: For an infinitesimally small c only one of the four $\vec{uv}$ edges it spawns is valid — we assign 0 to be the minimal value of the validity range of this edge (and of the opposite $\vec{vu}$ edge). As our algorithm is event-driven, we initialize an empty event queue $\mathcal{Q}$, storing events by their increasing c-order. For each obstacle vertex u we construct $\mathcal{L}_l(u)$ and $\mathcal{L}_r(u)$, based on the visibility edges we have just computed, and examine each pair of adjacent edges e_1, e_2 in $\mathcal{L}_l(u)$ and in $\mathcal{L}_r(u)$. We compute the c-value at which the adjacent e_1 and e_2 become equally sloped, if one exists, and insert the *visibility event* $\langle c, e_1, e_2 \rangle$ to $\mathcal{Q}$. We also compute the Voronoi diagram of the polygonal obstacles, and for each *non-monotone* Voronoi chain we locate the arc a that contains the minimal clearance value $c_{\min}$ of the chain in its interior and insert the *chain event* $\langle c_{\min}, a\rangle$ to $\mathcal{Q}$.

After this initialization step, we proceed to the event-handling step: While the event queue is not empty, we proceed by extracting the event in the front of $\mathcal{Q}$, associated with minimal c-value, and handle it according to its type.

Visibility events always come in pairs — that is, if $\vec{uv}$ becomes equally sloped with $\vec{uw}$, we will either have an event for the opposite edges $\vec{vu}$ and $\vec{vw}$, or for the opposite edges $\vec{wu}$ and $\vec{wv}$. We therefore handle a pair of visibility events as a single event. Let us assume that the edges $\vec{uv}$ and $\vec{uw}$ become equally sloped for a clearance value c', and at the same time the edges $\vec{vu}$ and $\vec{vw}$ become equally sloped (see Figure 2(b) and (c)). As the edges $\vec{uv}$ and $\vec{vu}$ now become blocked, we assign c' to be the maximal c-value of the validity range of $\vec{uv}$ and $\vec{vu}$. We also remove the other event involving $\vec{uv}$ (based on its other adjacency in $\mathcal{L}(u)$) from $\mathcal{Q}$, and delete this edge from $\mathcal{L}(u)$. We examine the new adjacency created in $\mathcal{L}(u)$ and insert its visibility event into the event queue $\mathcal{Q}$. We repeat this procedure for the opposite edge $\vec{vu}$. If the edge $\vec{uv}$ was valid before it was deleted and the edges $\vec{uw}$ and $\vec{vw}$ do not have a minimal validity value yet, assign c' to it, because these edges have become bitangent for this c-value (see Figure 2(c) for an illustration).

A *chain event* occurs when the value c equals the minimal clearance of a Voronoi chain χ_a, obtained on the arc a, which is equidistant from an obstacle vertex u and another obstacle feature. Let z_1 and z_2 be a's endpoints. We initiate two chain points $p_1(\chi_a)$ and $p_2(\chi_a)$ associated with the Voronoi chain χ_a. As c grows, $p_1(\chi_a)$ moves toward z_1 and $p_2(\chi_a)$ moves toward z_2. For all edges e incident to u, we compute the c-value c' for which e becomes incident to one of the chain points $p_i(\chi_a)$, and insert a *tangency event* associated with c' and e to the event queue. If a is equidistant from u and from another obstacle vertex v, we do the same for the edges incident to v. We also insert two *endpoint events* to the queue, associated with the clearance values obtained at z_1 and z_2, respectively.

When dealing with a chain event, we introduced two additional types of events: tangency events and endpoint events. A *tangency event* occurs when an edge $e = \vec{ux}$ (the endpoint x may either represent a dilated vertex or a chain point) becomes tangent to $B_c(u)$ at a chain point $p(\chi_a)$ associated with the Voronoi arc a, so we have to replace e by $\overrightarrow{p(\chi_a)x}$ associated with the chain point $p(\chi_a)$ and update the relevant adjacency lists and the priority queue accordingly. An *endpoint event* occurs when a chain point $p(\chi_a)$ reaches the endpoint z of the Voronoi arc a — in this case the edges associated with $p(\chi_a)$ should be transferred to the next arc in the chain (or possibly to the next chain, if z is a Voronoi vertex). We skip the fine technical details involved in handling these events, which can be found in the full version of this paper [5]. The total number of events is $O(n^2)$ and the time complexity of the algorithm is $O(n^2 \log n)$. A proof of correctness is provided in [5] as well.

3.2 The Query Stage

A query on the VV-complex is defined by a triple $\langle s, g, \hat{c}\rangle$, where s and g are the start and goal configurations, respectively, and $\hat{c}$ is the preferred clearance value. We assume that s and g themselves have a clearance larger than $\hat{c}$. Given a query, we start by computing the relevant portion of the Voronoi diagram: For each Voronoi chain we examine the clearance values of its end-vertices, as well as the chain minimum, and determine which portion of the chain (if at all) we should consider. This way we also obtain all the chain points for the given c-value $\hat{c}$.

Next we need to find the incident edges of s and g. This means that we should obtain two lists $\mathcal{L}(s)$ and $\mathcal{L}(g)$ containing the visibility edges emanating from s and g (respectively) to every visible circular arc and chain point. This is done using a radial sweep-line algorithm. We now start searching the graph we have implicitly constructed using a Dijkstra-like search to find the "shortest" path between s and g. Whenever we reach a vertex (a dilated polygon vertex or a chain point) we query the VV-complex with the given c-value $\hat{c}$ to obtain a list of its valid incident edges, and add g to this list if necessary. We proceed until the goal configuration g is reached.

The query time is dominated by the running time of Dijkstra's algorithm, which is $O(n \log n + k)$, where k is the number of edges encountered during the search. In practice, Dijkstra's algorithm turns out to be very fast, because hardly any geometric operations have to be performed anymore and we can therefore switch to machine-precision floating-point arithmetic.

References

[1] The CGAL project homepage. `www.cgal.org/`.

[2] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2nd edition, 2000.

[3] M. I. Karavelas. Segment Voronoi diagrams in CGAL, 2004. `www.cgal.org/UserWorkshop/2004/svd.pdf`.

[4] C. Ó'Dúnlaing and C. K. Yap. A "retraction" method for planning the motion of a disk. *J. Algorithms*, 6:104–111, 1985.

[5] R. Wein, J. P. van den Berg, and D. Halperin. The Visibility–Voronoi complex and its applications, 2004. `www.cs.tau.ac.il/~wein/publications/pdfs/vvc_TR.pdf`.

Constructing the City Voronoi Diagram Faster

Robert Görke* Alexander Wolff*

Abstract

Given a set S of n point sites in the plane, the City Voronoi diagram partitions the plane into the Voronoi regions of the sites, with respect to the City metric. This metric is induced by quickest paths according to the Manhattan metric and an accelerating transportation network that consists of c non-intersecting axis-parallel line segments. We describe an algorithm that constructs the City Voronoi diagram (including quickest path information) in $O((c+n)\text{polylog}(c+n))$ time using a wavefront expansion. For $c \in \Omega(\sqrt{n}\log^3(n))$ our algorithm is faster than an algorithm by Aichholzer et al. [2], which takes $O(n \log n + c^2 \log c)$ time.

1 Introduction

Let's assume Manhattan to be void of car traffic by the year 2050. Only a network of conveyors accelerates the movement of countless busy visitors in this huge pedestrian zone. As is known streets are arranged isothetically in Manhattan, so given a general direction, pedestrians can intuitively find a footpath to one of the many post offices. But time is precious, and thus a technique is required, telling an arbitrary pedestrian the quickest path to the post office that can be reached most quickly.

2 Concretization

We concretize the situation as follows. The transportation network $C = \{e_1, \ldots, e_c\}$ consists of c isothetic line segments, that are only allowed to touch at endpoints. Each segment e_i is assigned a speed $g_i > 1$. We require that the number of different speeds is constant. A segment can be accessed and left at any point. The n sites $S = \{\omega_1, \ldots, \omega_n\}$ are scattered arbitrarily in the plane. Movement off the network takes place with unit speed with respect to the Manhattan metric, while a segment e_i can be used (bidirectionally) to move with speed g_i. We define the distance d between two points a and b in the plane to be the temporal length of the quickest path Π between them: $d(a,b) = d_{\text{Manhattan}}(\Pi \setminus C) + \sum_{e_i \in C}(\frac{d_{\text{Manhattan}}(e_i \cap \Pi)}{g_i})$.

The definition of quickest paths induces a metric in the plane that we call *City metric*. As usual we define the *Voronoi region* $\text{reg}(\omega_i)$ of a site ω_i as the set of all points that are not closer to any other site ω_j. If we associate borders between regions in an arbitrary manner with one of the involved sites, we get a partition of the plane called the *City Voronoi diagram* $V_C(S)$. Given a query point $q \in \mathbb{R}^2$, the site in S closest to q can be determined by point location in time logarithmic in the complexity of $V_C(S)$. By virtue of our construction method we obtain a *refinement* $\mathcal{V}_C(S)$ [2] of the City Voronoi diagram that can then also report the quickest path to the closest site in additional time $O(L)$, with L being the path complexity. We now state our main result. The proof will be given at the end of this paper.

Theorem 1 (Construction of $\mathcal{V}_C(S)$) *Given an isothetic transportation network C with c edges, a constant number of different speeds on these edges and a set S of n sites, the refined City Voronoi diagram can be computed in $O((c+n)\log^5(c+n)\log\log(c+n))$ time using $O((c+n)\log^5(c+n))$ storage. The refined City Voronoi diagram answers queries asking for the quickest path to S in $O(L + \log(c+n))$ time, where L is the complexity of the path.*

3 Previous work

The City metric was first introduced by Abellanes et al. [1] who derived basic results for a single straight line as transportation network. Aichholzer et al. [2] presented an algorithm that constructs the City Voronoi diagram of n sites and c segments given a uniform network speed in $O(n \log n + c^2 \log c)$ time using $O(c+n)$ space. The resulting data structure answers quickest-path queries in $O(L + \log(c+n))$ time. In their algorithm the authors first prepare a set of time-stamped nodes in the plane using the continuous Dijkstra method [4]. Then carefully adapted straight skeleton figures scheduled at these nodes are computed by employing well known techniques for the construction of abstract Voronoi diagrams.

4 The wavefront expansion

Our algorithm constructs the City Voronoi diagram $V_C(S)$ by simulating the expansion of a wavefront

*Fakultät für Informatik, Universität Karlsruhe, `http://i11www.ira.uka.de/people`. R. G. supported by the European Commission within FET Open Projects DELIS, A. W. supported by grant WO 758/4-1 of the German Science Foundation (DFG) starting at the set S of sites. At time t the wavefront is the set of all points whose distance from S is t in the City metric. The key observation is, that during the course of the expansion each point of the plane is reached by the quickest possible path starting from S. In order to tell the quickest path from p to S we therefore need to note how the wavefront reached p and invert the path taken by the wavefront. This is done as follows. By storing where the wavefronts of different sites merge and by tracing vertices resulting from such mergings, we immediately obtain the partition $V_C(S)$, see Figure 1. We can trace the path of wavefront vertices in order to obtain a refinement of $V_C(S)$. Since the wavefront consists exclusively of vertices and straight line segments (due to the properties of the City metric), this refined City Voronoi diagram $\mathcal{V}_C(S)$ partitions $V_C(S)$ into regions of uniform wavefront expansion. Thus, if we store for each such region the direction the wavefront moved in, we can tell for all points of that region how to reach the youngest object of this region. By doing this repeatedly, we reach S, tracing back the expansion of the wavefront. See Figure 2 for an example. We discretize the continuous expansion of the wavefront at the points in time when an interaction, collision or an interference between the wavefront and the network or even another part of the wavefront happens. We call each of these points in time *events*. At an event the combinatorial shape of the wavefront changes.

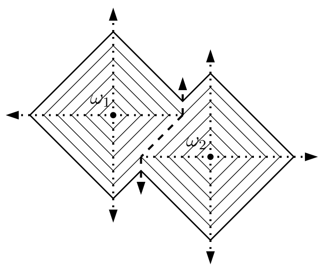


Figure 1: The wavefronts of two sites merge, tracing out a border.

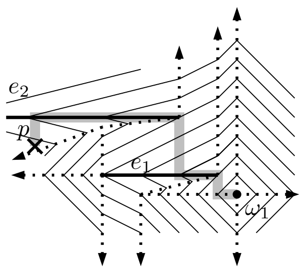


Figure 2: The path of the wavefront guides the way from p back to ω_1.

4.1 Events

We distinguish four types of events, depending on the situation. A vertex of the wavefront hitting a segment generates a type-A-event, while an edge of the wavefront sliding into a network node triggers a type-B-event. A type-C-event occurs when a wavefront edge shrinks to zero length and finally, a type-D-event is a collision of two parts of the wavefront. It is not hard to see the following:

Observation 1 *For any type of event the extent of changes on the wavefront is constant.*

As a consequence of this observation, we need to focus on the detection of events. An upcoming event can be detected by comparing for all edges and vertices of the wavefront the timestamp of their next collision. This comparison leads us to the notion of *virtual* events. A virtual event is an event that seems likely to occur during the wavefront expansion, but is then prevented by some other event with a lower timestamp, see Figures 3 and 4 for an example. We can even detect events that do happen, but still don't contribute to the complexity of $V_C(S)$. We call such events *redundant*, see Figure 5 for an example. Events that are neither redundant nor virtual are *relevant* and take part in shaping $V_C(S)$. Next we discuss an important result about the total number of relevant events.

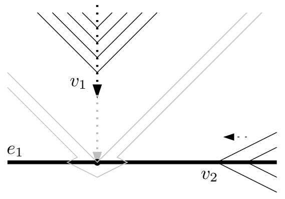


Figure 3: A type-A-event is pending.

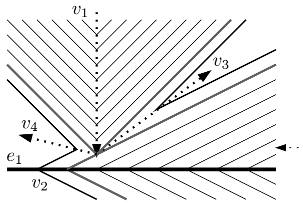


Figure 4: The event has been prevented.

4.2 The linear complexity

Adapting a result of Aichholzer et al. [2] to a constant number of network speeds we get the following result:

Theorem 2 *The number of edges, vertices and faces of the refined City Voronoi diagram $\mathcal{V}_C(S)$ is linear in the number c of segments and the number n of sites.*

This gives rise to the fact that the number of relevant events occurring during the wavefront expansion is also linear in $(c + n)$. Opposed to that, the number of redundant events can amount to $\Theta(c(c + n))$ (see Figure 5), and the number of virtual events can even add up to $\Theta(c + n)^2$. While these events are easy to identify, we cannot treat them explicitly. Thus we are left with the task of efficiently detecting the next event while implicitly ignoring irrelevant ones. In the next subsection we consider a unifying approach for detection of all four types of events.

4.3 The wavefront in space

We now add a third dimension (z-axis) to our situation, such that a positive z-component is added to the wavefront expansion which starts in the x-y-plane. Consequently wavefront vertices and edges trace out rays and polygons, respectively. Accordingly all network segments are extended to vertical unbounded rectangles and network nodes are extended to half-lines, both being unbounded in positive z-direction. If

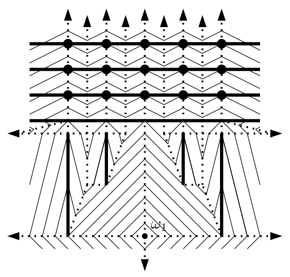


Figure 5: Cascade of redundant type-A-events (marked by disks).

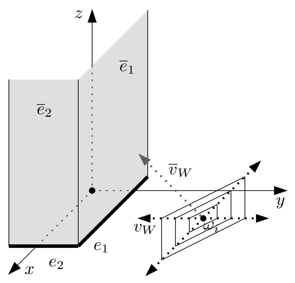


Figure 6: A type-A-event in space involving $\bar{v}_W$ and $\bar{e}_1$ is imminent.

the z-component of the wavefront expansion has unit speed we can easily tell the timestamp of an event by its z-coordinate in space. Figure 6 shows how the z-axis is added. Now, reconceiving the nature of each event in space, we can observe the following:

Observation 2 *In space any event can be described as a collision between a ray and a surface.*

Such collisions can be computed using ray-shooting techniques, but general methods for ray-shooting have unsatisfactory time bounds and we still need to take care of irrelevant events. The next section describes how we can efficiently detect upcoming events.

5 Maintaining the next event

Since each event is a collision of a ray and a polygon we can always determine the next event by maintaining the closest pair between these two changing sets of objects (polygons and rays).

5.1 The global prediction

Eppstein and Erickson [3] proposed a method of maintaining the closest pair among two changing sets R and B of objects according to a given distance measure d that can be computed in constant time. As a prerequisite the sets R and B need to support *minimization queries*, i.e. for any object $b \in B$ an object $r \in R$ minimizing $d(r, b)$ can be determined and vice versa. In our application R and B will be the foot points of rays and partially unbounded polygons, respectively. We use the following result:

Theorem 3 ([3]) *Suppose that after $P(n)$ preprocessing time, we can maintain a data structure of size $S(n)$ that supports insertions, deletions, and minimization queries, each in amortized time $T(n)$. Then after $O(P(n) + nT(n))$ preprocessing time, we can maintain the closest pair between R and B in $O(S(n))$ space, $O(T(n)\log(n))$ amortized insertion time, and $O(T(n)\log^2(n))$ amortized deletion time.*

Note that we can neglect the linear preprocessing time of the starting wavefront in our situation. Employing this theorem we are left with the lesser problem of efficiently performing both ray-shooting queries and their inverse, called *lowest-intersection queries*. Let us call the results of such queries *local* predictions and the result of the above theorem the *global* prediction. We now face the challenge of simplifying our data as to speed up minimization queries while taking implicit care of irrelevant events.

5.2 Simplification of wavefront data

The data we deal with for the purpose of local predictions comprises arbitrarily shaped, partially unbounded polygons in space. If we split these surfaces at each event along the current wavefront as depicted in Figure 7, we obtain *slabs* that are either triangles or quadrilaterals. We distinguish four types of slabs depending on the number of bounded edges and the presence of parallel edges. As long as a slab has not yet been involved in an event (except for the one that created the slab) we call it *active*. Analogously we define active rays. Inactive slabs can't take part in a relevant event. By Theorem 2 this augmentation of $\mathcal{V}_C(S)$ retains the linear complexity. We are now left with answering minimization queries for a linear number of triangles and unbounded quadrilaterals. Note that active slabs cover areas beyond the current wavefront. The key observation is, that we do not need to know exactly how far any slab has actually been traced out by the wavefront at any given time. The slabs have been designed to cover only those points of the plane that they would cover in the finished diagram, if they are not made inactive prematurely by some event involving them. The same holds for rays.

5.3 Orthogonalized sublocal queries

We now define slabs to be *similar* if their sides pairwise have the same angular position. For an example see Figure 8. Rays are similar if they merely point in the same direction and move at the same speed.

Lemma 4 *The number of classes of similarity of slabs and of rays is constant.*

Let us now consider an arbitrary combination of one class of slabs with one class of rays. We call the result of a minimization query involving all objects of exactly these two classes a *sublocal* prediction. Since by Lemma 4 the number of ray and slab classes is constant, the number of pairs of ray and slab classes is constant, too. Clearly, any local prediction can easily be computed out of sublocal predictions in constant time. Within a sublocal data structure considerable simplifications are possible. For each such data structure we can define a coordinate transformation f con-

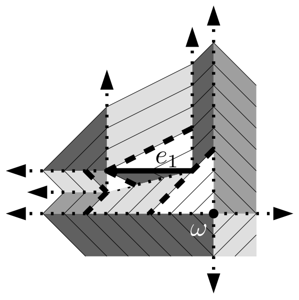


Figure 7: Division into slabs (dashed lines).

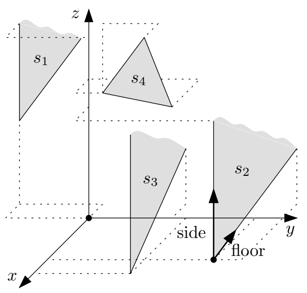


Figure 8: Only slabs s_1 and s_2 are similar.

sisting of at most one rotation and four concatenated shearings. First the rotation aligns the rays with the z-axis and one side of the slabs with the x-axis. Then step by step each side of the slabs is orthogonalized to two of the three axes. As shown in Figure 9, we end up with simple orthogonal range queries instead of ray-shooting or lowest-intersection queries. Note that in order to handle type-3 slabs (bounded triangles) we need to introduce the additional Ψ-axis (see Figure 10), adding one more level to the data structure.

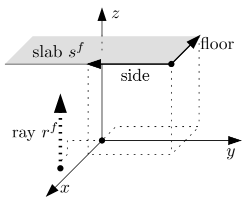


Figure 9: A transformed slab-ray pair.

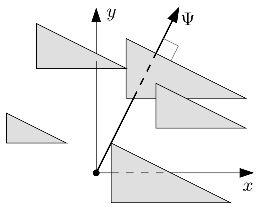


Figure 10: The additional Ψ-axis.

5.4 Feeding the global prediction

As stated earlier the global prediction relies on local predictions. Local predictions in turn are based on a constant number of sublocal queries, each being answered with a multidimensional orthogonal range query. Making use of Theorem 2 and of well-known results about multi level range trees and fractional cascading we can state the following:

Observation 3 *Sublocal data structures can each handle insertions, deletions and queries in $O(\log^3(c+n)\log\log(c+n))$ time using $O((c+n)\log^3(c+n))$ space. The same holds for local data structures.*

Comparing slabs with rays we observe that while slabs are static, rays are not and thus they cannot simply be represented by their static foot points p_{foot}. In order to do justice to the dynamic nature of rays we should in fact regard the (moving) tip of the rays. But instead of repeatedly advancing the tips of all rays we can simply apply a time corrected insertion: $p_{foot}^{new} := p_{foot} - (0,0,t)|\vec{v}|$, with t being the elapsed time and $\vec{v}$ being the direction of the ray. They key observation is that these modified foot points represent at all times the relative position of the tips of their rays, after the transformation f has been applied.

5.5 Ignoring irrelevant events

While the statement of Observation 3 is crucial to relevant events, we can show that due to the careful design of our slabs we don't need to add up any time for irrelevant events. As indicated in Figure 5, a redundant event is due to a wavefront vertex hitting a segment with equal or lower speed than segments hit by the same wavefront vertex earlier. If we simply refrain from forwarding local queries to sublocal data structures designed for segments with equal or lower speed, we implicitly ignore all redundant events. Due to the fact that slabs comprise only points they would actually reach if unhindered by events it is not hard to see that no virtual event will ever be globally predicted. Thus by Theorem 2 we get:

Lemma 5 *The total number of globally predicted events is $O(c+n)$.*

6 Proof of main result

Now we can put things together to prove Theorem 1. According to Lemma 5, $O(c+n)$ events are treated. Using Observation 3, Theorem 3 lets us predict each event in $O(\log^5(c+n)\log\log(c+n))$ time. Hence, the total time used for the global prediction is $O((c+n)\log^5(c+n)\log\log(c+n))$, which dominates the time needed to handle events (see Observation 1 and Lemma 5). Observation 3 and thus Theorem 3 require $O((c+n)\log^3(c+n))$ space.

References

[1] M. Abellanas, F. Hurtado, C. Icking, R. Klein, E. Langetepe, L. Ma, B. Palop del Río, and V. Sácristan. Proximity problems for time metrics induced by the l_1 metric and isothetic networks. In *Actas de los IX Encuentros de Geometría Computacional*, pages 175–182, Universidad de Girona, 2001.

[2] O. Aichholzer, F. Aurenhammer, and B. Palop del Río. Quickest paths, straight skeletons, and the City Voronoi diagram. In *Proc. 18th Symp. Computational Geometry*, pages 151–159. ACM Press, June 2002.

[3] D. Eppstein and J. G. Erickson. Raising roofs, crashing cycles, and playing pool: Applications of a data structure for finding pairwise interactions. *Discrete Computational Geometry*, 22(4):569–592, June 1999.

[4] J. S. B. Mitchell, D. M. Mount, and C. H. Papadimitriou. The discrete geodesic problem. *SIAM Journal on Computing*, 16(4):647–667, Aug. 1987.

Computing Transportation Voronoi Diagrams in Optimal Time

Yaron Ostrovsky-Berman [*]

Abstract

We present the first time-optimal algorithm for computing the Voronoi Diagram under the metric induced by a transportation network with discrete entry and exit points. For input with n sites, k stations, and e transportation lines, the algorithm computes the Voronoi Diagram in $O\big((n+k)\log(n+k)+e\big)$ time.

1 Introduction and related work

Shortest Path Maps (SPM) and Voronoi Diagrams are well known geometric tools for answering distance related queries. The SPM is a subdivision of space, which allows finding the shortest path from a source point to a query point. The Voronoi Diagram is a subdivision of space which allows finding the site closest to a query point (multiple sources). These tools are useful in Geographic Information Systems, where the realism of the results depends on the underlying terrain model.

There have been many attempts to model the complexity of real world terrain with a simple mathematical model. The most general of these is the weighted region model [6] in which the plane is divided into regions with weights corresponding to the difficulty of crossing the terrain. There is no efficient algorithm for finding shortest paths in this general model without approximating the solution. One important special case is shortest paths amidst polygonal obstacles in the plane, in which the obstacles have infinite weight and the free space has unit weight. Mitchell [5] gave the first sub-quadratic solution, which runs in $O(n)$ space and $O(n^{3/2+\epsilon})$ time. Hershberger and Suri [4] presented the first optimal time algorithm, with $O(n\log n)$ space and time complexity. Both employ the continuous Dijkstra paradigm. Abellanas et al. [1] survey recent results obtained for models of urban environments.

In this paper, we model public transportation networks which provide time saving routes with discrete entry and exit points. We describe the network by an undirected graph with positive edge weights proportional to travel time, and assume the entire plane is accessible by foot. This type of transportation network cannot be modelled with weighted regions because it allows crossing the transportation line at no time cost, but restricts entrance to and exit from the network to the fixed stations. The proposed model was introduced by Münch in his PhD thesis [7], which studies a network of airlifts over the Euclidean plane. The algorithm for computing the Voronoi Diagram in the induced airlift metric was briefly discussed in Aichholzer et al. [2] and later expanded by Palop in her PhD thesis [9]. The algorithm completes the network graph by assigning Euclidean weights to disconnected stations, and applies the discrete Dijkstra algorithm to compute the station weights. The station weights are used as input to the Additively Weighted Voronoi Diagram (AWVD) algorithm, from which the desired subdivision is obtained. The complexity of the algorithm is $O(k^2+n)$ space and $O\big(k^2+(n+k)\log(n+k)\big)$ time, where n is the number of sites and k is the number of stations. The authors also raised the question of whether there exists a matching lower bound for the time complexity.

Our previous work [8] improves upon this result by computing the station weights with an input sensitive algorithm having the same worst case time complexity and $O(n+k\log k+e)$ space complexity, where e is the number of transportation lines. The algorithm has provably better time bounds under realistic regularity assumptions on the input.

In this paper, we describe the properties of the metric induced by the transportation network, and settle the question of the time complexity by combining the continuous Dijkstra method with the reduction to AWVD to obtain optimal $O(k\log k+e)$ and $O((n+k)\log(n+k)+e)$ time algorithms for the SPM and the Voronoi Diagram, respectively.

2 Definitions and properties

Let $T\subset\mathbb{R}^2$ denote a set of station positions in the plane, and let E denote the connection relation between the stations, such that $(t_1,t_2)\in E$ if and only if $t_1,t_2\in T$ and the stations are connected by a line. Denote the positive weight of a transportation line $(t_1,t_2)\in E$ by $w(t_1,t_2)$ and define it to be infinity when the points are not connected stations. The graph $\langle T,E,w\rangle$ describes the transportation network. For the Voronoi diagram problem, the set $S\subset\mathbb{R}^2$ denotes the sites. Let $d(p,q)$ denote the Euclidean distance between two points in the plane. The transportation distance between two points is defined re-

[*]School of Engineering and Computer Science, The Hebrew University of Jerusalem, yaronber@cs.huji.ac.il

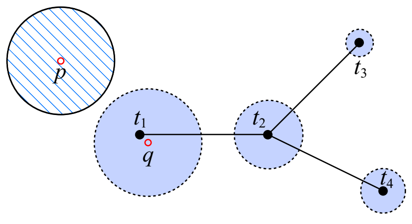


Figure 1: The unit disc. Stations t_1, t_2, t_3, t_4 are connected by solid transportation lines. The hatched disc is the unit disc of p, and the shaded discs comprise the unit disc of q.

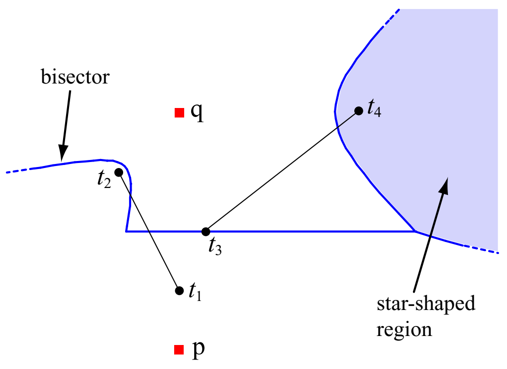


Figure 2: The bisector of two points. Solid circles are stations, the thin solid lines connecting them are transportation lines. The weight of the transportation lines is tenth of the Euclidean distance between the stations. The thick curve is the bisector of p and q. The shaded region is part of the bisector because $d_T(p, t_4) = d_T(q, t_4)$.

cursively as follows:

$$d_T(p, q) = \min\{d(p, q),\ w(p, q),\ \min_{t \in T}\{d_T(p, t) + d_T(t, q)\}\}$$

The unit disc of a point p in the metric induced by the transportation distance, $B_T(p) = \{x | d_T(p, x) \leq 1\}$, depends on the proximity of p to the network (Figure 1). This leads to the following properties:

1. The Voronoi cell of a site can have several connected components.

2. The transportation distance bisector of two points p and q, $b_T(p, q) = \{x | d_T(p, x) = d_T(q, x)\}$, consists of line segments (points whose shortest path to p and q is a straight line), hyperbolic arcs (points whose shortest path to p or q uses the transportation network), and star shaped regions bounded by line and hyperbolic segments (points whose shortest paths to p and q begins with a shared network station). Figure 2 illustrates these cases.

As shown in [8, 9], the shortest path map is constructed by computing the transportation distance from the source to all the stations, then using this distance as a negative weight in the AWVD of the source and the stations. In the AWVD, the distance between a point p and a site s with weight w_s is $d_{AWVD}(p, s) = d(p, s) - w_s$. Fortune [3] showed how to construct the AWVD in $O(n \log n)$ time. The transportation Voronoi diagram is constructed similarly to the SPM, only now the weights are assigned according to the *closest* site in transportation distance.

3 The continuous Dijkstra method

We now show how to modify the continuous Dijkstra method [4, 5] to handle the transportation distance metric instead of the geodesic distance induced by polygonal obstacles. For simplicity of presentation, we address the single source problem only. The extension to multiple sources is straightforward.

The continuous Dijkstra method simulates the propagation of a wavefront from the source point. At simulation time δ, the wavefront is the locus of points with transportation distance δ from the source. The wavefront is comprised of wavelets, which are points having the same predecessor in the shortest path to the source. In its purest form, the simulation advances between discrete events of the following type: 1. A wavelet collides with a station; 2. A wavelet collides with another wavelet; 3. A wavelet is engulfed by neighboring wavelets and disappears. In practice, identifying these events as they occur is difficult, and the method of [5] does not attempt to do so. Instead it ensures that when the events are discovered, there is only local work to do to fix the discrepancy. The method of [4] uses a clever subdivision of the plane that guides the wavefront propagation, and computes "approximate wavefronts" at the edges of this subdivision, which are used in the final stage to construct the shortest path map.

In the geodesic problem, an event of type 1 is a collision of the wavelet with an obstacle vertex. This vertex becomes the generator of a new wavelet. In the transportation metric, the station encountered is not a new generator, but its neighbors in the transportation network possibly become generators in the future (after a time equal to the connection time, which is the edge weight). We call the station that schedules new generators a *transporter*. For each station t we store the time $g(t)$ in which t becomes a generator (initialized to infinity), and the set $p(t)$ of predecessors of t in the shortest path to the source s (the cardinality of $p(t)$ is greater than one when the path is not unique). Suppose a wavelet hits station t at time δ. The simulation algorithm performs the update operations required by an obstacle vertex collision event, but instead of creating a new generator at t, it first checks if $\delta \leq g(t)$, and if so proceeds as described

```
Procedure propagate-transporter$(t, \delta)$
1.   set $g(t) = \delta$
2.   for-each $r$ with $(t, r) \in E$:
2a   if $\delta + w(t,r) < g(r)$ then
       - set $g(r) = \delta + w(t,r)$ and set $p(r) = t$
       - insert new-generator-event$(r)$ into event queue at time $g(r)$
2b   else-if $\delta + w(t,r) = g(r)$ then
       - set $p(r) = p(r) \cup t$
     end for-each
```

Table 1: Wavefront propagation through the network.

in Table 1, possibly scheduling new generators at the neighboring stations.

When the simulation time reaches a new-generator-event(r), a new wavelet with generator r is created (since the algorithm does not clear future events for the same r in step 2a, the simulation simply continues if r was already processed). The new wavelet automatically triggers an event of type 1 for a collision with r. Events of type 2 and 3 are handled as in the geodesic problem.

Observation 1 *The wavefront propagation algorithms of [4, 5] both detect collisions with an obstacle vertex t and set $g(t)$ correctly before the vertex is used as a generator, and this property holds when network stations are treated as obstacle vertices.*

We now prove that this observation holds after the modification made in propagate-transporter.

Lemma 1 *For every station t in the transportation graph, the propagation algorithm sets $g(t) = d_T(s,t)$ before t is used as a generator or a transporter.*

Proof. By induction on the number of links in the shortest transportation distance path from s to t. If there is one link, then the shortest path is Euclidean, and according to Observation 1 a wavelet leaving s will encounter t and set $g(t)$ correctly. If the path is longer, then by induction, the predecessor p of t has $g(p) = d_T(s,p)$ set before it is used as a generator or a transporter. If $(p,t) \notin E$ then part of the shortest path uses the transportation network, and p is the terminating station. Thus p was scheduled to be a generator by its predecessor in the path (step 2 of propagate-transporter). Since $d_T(s,p) < d_T(s,t)$, a wavelet is generated by p which, according to Observation 1, will collide with t and set $g(t)$ correctly. If $(p,t) \in E$ then p is a transporter and step 2 of propagate-transporter correctly sets $g(t) = g(p) + w(p,t) = d_T(s,t)$. □

Lemma 1 proves the correctness of the modification to the wavefront simulation. The rest of the algorithm is unchanged, thus when it terminates, the values of

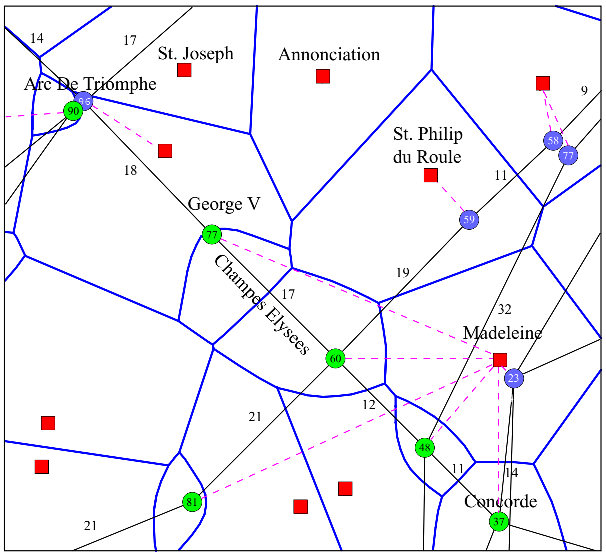


Figure 3: The transportation Voronoi diagram of the Paris Metro and churches. Solid discs are Metro stations, the number inside is the value of $d_T(s,t)$ for the closest church. Solid thin lines are the Metro lines, and numbers near their middle denote their weights. Solid squares are churches. A dashed line from a church to a station means that the church is closest to the station. Solid curves are the boundaries of the AWVD cells of the stations and the churches.

$g(t)$ equal $d_T(s,t)$ for all $t \in T$. As noted above, the AWVD of the source and the stations, with weights equal to the negative value of the transportation distance, gives the shortest path map of s. The predecessor information $p(t)$ is used backwards to determine the tree of shortest paths from s.

Figure 3 shows the transportation Voronoi diagram of part of the Paris Metro and nearby churches. The station weights were computed with our implementation of the reduction to AWVD in [8]. The AWVD was computed with the CGAL Apollonius graph class. Note that the Madeleine church controls more area because of its proximity to a Metro station. Its Voronoi cell has two connected components and consists of the the church's cell plus the cells of five nearby stations.

4 Complexity analysis

We now analyze the complexity of the algorithm, starting with the modification described in Section 3. In what follows n, k, and e stand for the number of sites, stations and transportation lines, respectively. We assume the event queue is implemented with a Fibonacci heap, which supports insertions and minimum extractions in amortized $O(1)$ and $O(\log n)$ time, respectively. Assuming the wavefronts are efficiently pruned (as is the case in [5, 4]), there are $O(1)$ events of type 1 per station. The modification to the event handler performs $O(1)$ comparison operations and $O(1)$ insertions to the queue per neighbor of t in the graph. Because each edge is visited $O(1)$ times, the total added complexity of the modification is $O(e)$.

The complexity of the entire algorithm depends on the continuous Dijkstra method used. The algorithm of Mitchell [5] results in optimal $O(n + k + e)$ space complexity and $O\big((n + k)^{3/2+\epsilon} + e\big)$ time complexity, while the algorithm of Hershberger and Suri [4] results in $O\big((n+k)\log(n+k)+e\big)$ space and time complexity, which is time-optimal.

From a practical standpoint, we note that the algorithms presented in [4, 5] are complicated and have not been implemented. Our previous algorithm [8] ignores events of types 2 and 3 altogether, and instead prunes the propagation of wavelets by bounding the radius of their effect on the transportation distance. The bound equals the difference between the maximal and minimal transportation distance from a station to a site, and thus decreases monotonically as the algorithm advances, lowering the cost of each iteration. On classes of transportation networks such as clusters or uniform distribution of stations and sites, the complexity of this method is provably lower than the quadratic worst case, and experiments show that this is true of most networks, including random networks and sites.

5 Conclusion

We have presented the first time-optimal algorithm for computing the transportation Voronoi Diagram. The existence of an optimal space and time algorithm is an open problem related to geodesic distance Voronoi Diagram.

To make the transportation network more realistic, the model can be augmented by access, connection, and waiting times of the stations and lines. See [8] for details. Furthermore, the combination of the algorithm proposed here with the original algorithm for the geodesic problem can be used for solving shortest path problems in the presence of polygonal obstacles as well as a transportation network, enriching the urban environment model.

References

[1] M. Abellanas, F. Hurtado, and B. Palop. Transportation networks and Voronoi Diagrams. In *International Symposium on Voronoi Diagrams in Science and Engineering*, Tokyo, 2004.

[2] O. Aichholzer, F. Aurenhammer, and B. Palop. Quickest paths, straight skeletons, and the city Voronoi Diagram. In *Proc. of the 18th annual symposium on Computational Geometry*, pages 151–159, 2002.

[3] S. Fortune. A sweepline algorithm for Voronoi Diagrams. *Algorithmica*, 2(2):153–174, 1987.

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

[5] J. S. B. Mitchell. Shortest paths among obstacles in the plane. *International Journal of Computational Geometry and Applications*, pages 309–332, 1996.

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

[7] O. Münch. *Das Voronoi-Diagramm in der Airline-Metrik: Untersuchung der strukturellen Eigenschaften und Veranschaulichung durch ein Java-Programm.* PhD thesis, FernUniversität Hagen, Fachbereich Informatik, 1998.

[8] Y. Ostrovsky-Berman. Transportation Voronoi Diagrams. Technical Report 2003-2, Leibniz Center for Research in Computer Sciences, 2003. URL: www.cs.huji.ac.il/~yaronber/

[9] B. Palop. *Algorithmic problems on proximity and location under metric constraints.* PhD thesis, U. Politécnica de Madrid, 2003.

Approximations of 3D Generalized Voronoi Diagrams

Imma Boada[*] Narcís Coll[*] Narcís Madern[*] J. Antoni Sellarès[*]

Abstract

We introduce a new approach to approximate generalized 3D Voronoi diagrams for different site shapes (points, spheres, segments, lines, polyhedra, etc) and different distance functions (Euclidean metrics, convex distance functions, etc). The approach is based on an octree data structure denoted Voronoi-Octree (VO) that encodes the information required to generate a polyhedral approximation of the Voronoi diagram. Since the medial axis of a polyhedron is a subset of a generalized Voronoi diagram the VO can also be used to encode it. Using the information of the VO we generate and visualize a polyhedral approximation of a generalized Voronoi diagram, and we also solve proximity problems that do not require an explicit representation of the Voronoi diagram such as nearest neighbor queries.

1 Introduction

Given a set of primitives, called Voronoi sites, the generalized Voronoi diagram partitions the space into regions, one per site, such that all points in a region have the same closest site according to some given distance function. Many variants of these diagrams can be considered: by taking sites of different shape or nature, associating weights to the sites, changing the underlying metrics, or using individualized distance functions for the sites [1, 2, 9]. Generalized Voronoi diagrams are widely used in many scientific fields such as computer graphics, geometric modelling, shape analysis, robot motion planning or scientific visualization [6].

Computing a generalized 3D Voronoi diagram involves the manipulation of high-degree algebraic surfaces and their intersections. The exact computation of the diagram poses many problems in terms of robustness and CPU time due to the high number of precision calculations that have to be made [9]. Since the exact computation is known to be hard, in most of the applications approximations of the real diagram within a predetermined precision are used.

There are few methods to approximate generalized 3D Voronoi diagrams and most of them are restricted to Euclidian distance. Lavender at al [8] proposed a hierarchical approach to compute an approximate Voronoi diagram of a set of general sites in arbitrary dimension. They represent the sites by an octree and the cells of the approximate diagram are obtained by considering the distance to the sites. The continuity of the boundary of the approximate diagram is not guaranteed by any theoretical study. Vleugels et al. [13] also proposed a hierarchical approach restricted to convex sites that adaptively subdivides the space into regular cells and computes the Voronoi diagram up to a given precision. Teichmann et al. [12] proposed a technique restricted to triangle sites that subdivides the space into tetrahedral cells. Sites are inserted into a standard octree and a polygonal approximation of the Voronoi diagram is computed by a wavefront propagation strategy. None of these mentioned works gives properties about the convergence to the exact diagram and the computational cost. Some recent papers, driven by applications and using different approaches, investigate the approximate and the exact practical computation of the medial axis of a polyhedron [3, 4, 5].

2 Our Approach

We aim to define a general approach to obtain polyhedral approximations of generalized Voronoi diagrams that simultaneously support approximate nearest neighbor queries in an efficient way. To reach our objective we consider that the octree is the most suitable data structure since it allows to encode information at different levels of detail. A straightforward way to built this octree is by subdividing recursively the region containing all the Voronoi sites and storing at each vertex of the node the information of its nearest site. Despite the simplicity of this strategy, it is costly to implement since it requires an evaluation of all sites against all sites at each step of the octree construction process. On the other hand, since we know that rarely a Voronoi region is adjacent to all the other Voronoi regions, it has no sense to take into account all the sites when computing the nearest site of a vertex node. To avoid processing all the sites at each step of the process we have designed a propagation strategy that detects when new information of the sites is known and also where it has to be stored to guarantee the correctness of the final codification. Although this strategy forces us to restrict our proposal to diagrams with connected Voronoi regions the cost of the method is reduced considerably.

[*]Institut d'Informàtica i Aplicacions, Universitat de Girona, Spain, {imma,coll,nmadern,sellares}@ima.udg.es. Partially supported by grant TIN2004-08065-C02-02

The new main idea of our approach is the combination of the subdivision and the propagation processes. Such a combination also provides an efficient way of dynamically maintaining the VO under the insertion or deletion of sites.

In the next sections we describe the VO construction and its dynamic maintenance.

2.1 Basic Definitions

The set of input sites is denoted as $\mathcal{S} = \{s_1, \cdots, s_n\}$ and R is the cubic region where the approximation of the generalized Voronoi diagram must be computed. Each site s is represented by a tuple $s =< G_s, D_s, P_s >$, where G_s defines the geometry of the site s, D_s is the function that gives the distance from any point p to s and P_s, called the base point of s, is a point such that $D_s(P_s) = 0$ and $P_s \in R$. Each site $s_i \in \mathcal{S}$ has an associated Voronoi region $VR(s_i)$. The generalized Voronoi diagram of $\mathcal{S}$, denoted $\mathcal{V}$, is defined as the partition of the plane induced by the Voronoi regions.

Let N be a node of a octree and s a site. We say that s is an *I-site* with respect to N when $P_s \in N$, a *V-site* with respect to N when a vertex v of N satisfies $v \in VR(s)$ and a *F-site* with respect to N when it is not a V-site and a face f of N satisfies $f \cap VR(s) \neq \emptyset$. The *sites of* N are the set of sites that are I-site, V-site or F-site with respect to N. A node N is a terminal node if it is completely contained in a Voronoi region, i.e. the total number of V-sites, I-sites and F-sites is one.

2.2 The VO Construction Algorithm.

The VO construction algorithm is based on a breadth first strategy using a priority queue sorted by the level of the node in the octree. This allows us to introduce and update the information of the sites stored in the VO when it is required in order to guarantee the completeness and correctness of the encoded information. Let R be a cubical region containing $\mathcal{S}$, Q the priority queue and L_M the maximal subdivision level of the VO fixed by the user. The pseudo-code of the algorithm is reported below (see Algorithm 1). In the pseudo-code the following primitives are used:

INITIALIZE(R,$\mathcal{S}$). Creates the root node from the vertices of R and assigns all the sites as I-sites. It computes the V-sites of the root node (i.e. the nearest I-site of each root vertex) and initializes Q with the root node.

UPDATE(N). Updates the V-site of each vertex of a node N with the nearest of its sites. The distance from the vertex v to a site s is computed using $D_s(v)$.

LEAF(N,L_M). Checks if node N is a leaf or not.

SUBDIVIDE(N). Creates the eight son nodes of a node N, properly distributes the I-sites and F-sites of N to them and computes the V-sites of the sons taking into account the sites of N.

PROPAGATE(N, Q). Checks the coherence between a node N and its adjacent nodes. For each vertex v of N with V-site s_1 locates the set of its adjacent nodes. Let N' be one of these nodes. If v is a vertex of N' with a different V-site s_2 at v, this procedure updates this V-site with s_1 and sends N' to Q. On the contrary, if v is not a vertex of N' (i.e. it is on a face) and s_1 is different to the V-site s_2 of this face, the procedure assigns s_1 to N' as an F-site and sends N' to Q.

COMPACT(N). Checks if all the sons of the node N are terminal. If they are, it prunes the son nodes.

Algorithm 1 OctreeConstruction(R, S, L_M)

```
octree VO;
node $N$;
queue $Q$;
VO.INITIALIZE($R$, $S$);
Push($Q$,VO.Root());
while No Empty($Q$) do
  $N$= Pop($Q$);
  VO.UPDATE($N$);
  if VO.LEAF($N$, $L_M$) then
    VO.PROPAGATE($N$, $Q$);
    VO.COMPACT($N$.PARENT());
  else
    VO.SUBDIVIDE($N$);
    for $i = 1$ to 8 do
      VO.PROPAGATE($N$.SON($i$), $Q$);
      if No VO.LEAF($N$.SON($i$), $L_M$) then
        Push($Q$, $N$.SON($i$));
      end if
    end for
  end if
end while
```

2.3 Polyhedral Approximation of the Voronoi diagram

To generate a polyhedral approximation of the Voronoi diagram we use an strategy based on the cuberille iso-surface extraction algorithm proposed by Herman and Liu [7] and enhanced and optimized by many others. Our algorithm proceeds in two steps. First, we select leaf nodes intersected by the boundaries of the Voronoi diagram. Nodes whose corner values are all inside the same Voronoi region are ruled out since no boundaries are contained within and thus they have no effect on the final approximation. Second, for each selected node we approximate the boundaries of the Voronoi diagram contained in it. To perform this approximation each selected node is subdivided in $2 \times 2 \times 2$ cells, one for each vertex of the node. For each one of these vertices we evaluate the three edges incident to it. We assume that an edge is intersected by a boundary of the Voronoi diagram when it has different sites in its extremes. According to the number of intersected edges incident to the vertex we approximate the boundary contained in the node by none, one, two or three of the internal faces of the cell assigned to such vertex.

Figure 1 shows some of the results obtained with the proposed approach.

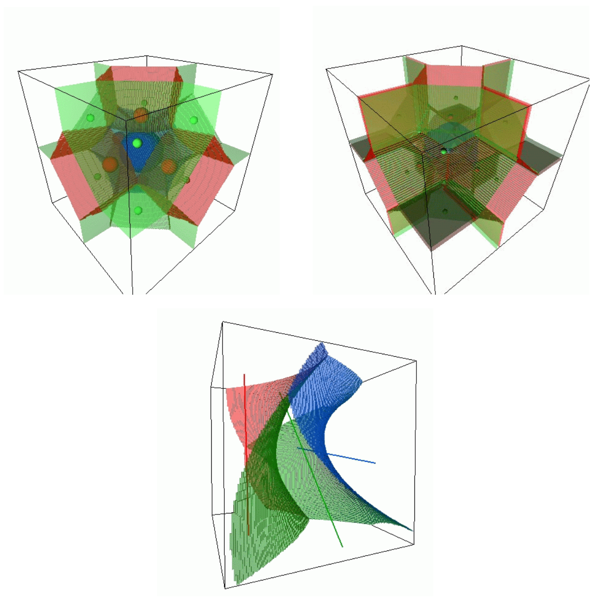


Figure 1: Voronoi diagrams of different scenes obtained with our method: spheres with Euclidean distance, points with L_1 distance and lines with Euclidean distance.

2.4 Computational Cost

To evaluate the VO construction algorithm some considerations have to be taken into account: (i) The mean number of intersection points between a surface of area A and a cubic grid of lines of size u (distance between two consecutive parallel lines of the grid) is $\frac{3A}{2u^2}$ [11]. (ii) If n is the number of sites, for each level we distribute the n sites to some nodes as I-sites. (iii) For each node we need to locate its neighbor nodes. This can be done in L_M worst time, but the expected time for locating neighbors is constant [10]. (iv) The proposed algorithm subdivides nodes that contain a piece of a bisector. We are going to assume that there exist a level L_T such that if $L_M > L_T$ all the sites are a V-site of some node and all the pieces of the bisectors are contained in the Voronoi diagram $\mathcal{V}$. (v) Let $l \leq L_M$ be a level of the octree, V the boundary of the Voronoi diagram and $V(l)$ the boundary of the Voronoi diagram of the sites that are V-site in some node of level l. Let $UV(l) = \bigcup_{i=1}^{l} V(i)$. From the last consideration we have that if $l > L_T$ then the piece of the bisector contained in a node of level l is contained in V, otherwise is contained in $UV(l)$.

Let A be the area of V, $A(l)$ the area of $UV(l)$, $N(L_M)$ the mean number of nodes generated by the construction algorithm of a VO of a maximum level L_M and $T(L_M)$ the mean running time of this algorithm. According to the previous considerations we obtain the following results.
If $L_M \leq L_T$ then $N(L_M) = O\left(\frac{4^{L_M+1}}{a^2}A(L_M)\right)$ and $T(L_M) = O\left(nL_M + \sum_{l=1}^{L_M} \frac{4^l}{a^2}A(l)\right)$. Otherwise, if $L_M > L_T$ then $N(L_M) = O\left(\frac{4^{L_M+1}}{a^2}A\right)$ and $T(L_M) = O\left(nL_M + \sum_{l=1}^{L_T} \frac{4^l}{a^2}A(l) + \frac{4^{L_M+1}-4^{L_T+1}}{a^2}A\right)$.

2.5 Dynamic Maintenance

The potential of the proposed VO has led us to consider the dynamic maintenance of this data structure as an essential operation. Given a VO of a Voronoi diagram approximation and a site s to be inserted the insertion algorithm applies an expansion process that goes from the leaf node containing the base point P_s to all the nodes containing a piece of the boundary of its associated Voronoi region. The algorithm proceeds as follows. First, it applies a top-down VO traversal to identify the leaf node N that contains P_s, enters s as an I-site of N and initializes with N a queue Q used to maintain the nodes to be processed. Then, Q is processed as in the VO construction process (see Algorithm 1). Given a VO of a Voronoi diagram approximation and a site s to be deleted, the deletion algorithm applies a contraction process. It starts with a node containing part of the boundary of the Voronoi region of s, and processes all the nodes containing part of this region. The deletion algorithm performs a top-down VO traversal to identify the leaf node N that contains P_s and erases s as I-site of N. Starting from N, it traverses the nodes that have s as a V-site until a node N' that contains part of the boundary of $VR(s)$ is reached. Next, it initializes with N' a queue Q which is processed as in the VO construction process but applying a new update procedure: a V-site is updated using the sites of the node and the sites of all adjacent nodes and excluding s (see Figure 2). The cost of both algorithms is proportional to the nodes intersected by the Voronoi region associated to the site to be inserted or deleted.

3 Medial Axis

To codify the medial axis in a VO we modify the VO construction algorithm in order to avoid the subdivision of the exterior nodes of the polyhedron and the nodes containing pieces of bisectors between adjacent

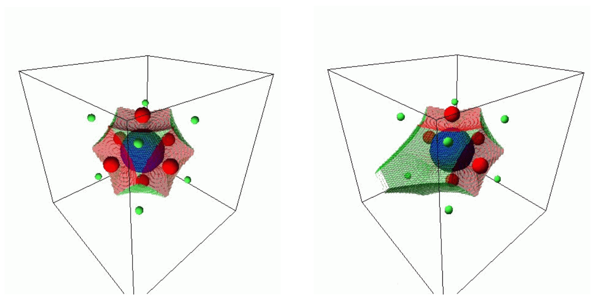


Figure 2: Results obtained by applying to scene of Figure 1(a) the deletion algorithm. Only the Voronoi region of the central sphere before and after the deletion is visualized.

sites. Such a modification implies the definition of two new terminal node criteria.

The first new criterion aims to avoid the subdivision of the exterior nodes. We associate a unitary vector n_s to each site s as follows. If s is a face then n_s is the normal vector to the face oriented outwards the polyhedron, if s is an edge then n_s is the sum of the vectors associated to its adjacent faces and if s is a point then n_s is the sum of the vectors associated to its adjacent faces. Let v be a vertex of a node and s its V-site with base point p_s. The vertex v is exterior to the polyhedron if $(v - p_s) \cdot n_s > 0$. A node N is a terminal node if all its vertices are exterior and any I-site is contained in it. The second new terminal criterion avoids the subdivision of the nodes containing pieces of bisectors between adjacent sites. A node N is a terminal node if all its edges have two V-sites adjacent in the polyhedron.

In Figure 3 we show the medial axis of a polyhedron. Note that what we obtain is a simplified medial axis [5].

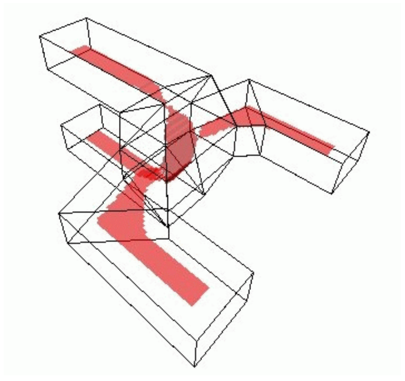


Figure 3: Polyhedron and its medial axis.

4 Nearest Neighbor Queries

The large variety of Generalized Voronoi diagrams that can be encoded by a VO and its hierarchical nature allows us to solve the nearest neighbor problem in many cases. Let $\mathcal{S} = \{s_1, \cdots, s_n\}$ be a set of sites and R a region containing $\mathcal{S}$. Given a query point $q \in R$, we want to find the site s such that $D_s(q) \leq D_{s'}(q)$ for all $s' \in \mathcal{S}; s' \neq s$. To approximately solve the problem it suffices to construct the VO of $\mathcal{S}$ in R, next determine in $O(L_M)$ time the VO leaf node containing the point q and select the nearest site between the set of V-sites of the node.

In a similar way we can solve other kind of queries such as the set of nearest sites of a given site and the closest pair of sites.

References

[1] F. Aurenhammer. Voronoi diagrams: A survey of a fundamental geometric data structure. In *ACM Computer Surveys* 23(3), pages 686–695, 1991.

[2] F. Aurenhammer and R. Klein. Voronoi diagrams. In J.R. Sack and J. Urrutia (Eds.), *Handbook of Computational Geometry*, pages 201–290, Elsevier, 2000.

[3] T. Culver, J. Keyser and D. Manocha. Exact Computation of the Medial Axis of a Polyhedron. In *CAGD* 21(1), pages 65–98, 2004.

[4] M. Etzion and A. Rappaport. Computing Voronoi skeletons of a 3-D polyhedron by space subdivision. In *Computational Geometry* 21, pages 87–120, 2002.

[5] M. Foskey, M. Lin, and D. Manocha. Efficient computation of a simplified medial axis. In *Journal of Computing and Information Science in Engineering* 3, pages 274-284, 2003.

[6] C. Gold. *The Voronoi Web Site*, http://www.voronoi.com/.

[7] G.T. Herman and K.H.. Liu. Three Dimensional Display of human Organs from Computed Tomograms. In *Computer Graphics and Image Processing* 9(1), pages 1–21, 1979.

[8] D. Lavender, A. Bowyer, J. Davenport, A. Wallis and J. Woodwark. Voronoi diagrams of set-thoretic solid models. In *IEEE Comput. Graph. Appl.* 12(5), pages 69–77, 1992.

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

[10] H. Samet. *Applications of Spatial Data Structures: computer graphics, image processing, and GIS.* Addison-Wesley, 1993.

[11] K. Sandau. How to estimate the area of a surface using the spatial grid. In *Acta Stereologica* 6/III, pages 31–36, 1987.

[12] M. Teichmann and S. Teller. Polygonal approximation of Voronoi diagrams of a set of triangles in three dimensions. *Technical Report 766, Laboratory of Computer science, MIT*, 1997.

[13] J. Vleugels and M. Overmars. Approximating Generalized Voronoi Diagrams in Any Dimension. In *Int. J. Computational Geometry and Applications* 8, pages 201–221, 1998.

Average case complexity of Voronoi diagrams of n sites from the unit cube[*]

Marcin Bienkowski[†] (young@upb.de) Valentina Damerow[†] (vio@upb.de)
Friedhelm Meyer auf der Heide[†] (fmadh@upb.de) Christian Sohler[†] (csohler@upb.de)

Abstract

We consider the expected number of Voronoi vertices (or number of Delaunay cells for the dual structure) for a set of n i.i.d. random point sites chosen uniformly from the unit d-hypercube $[0,1]^d$. We show an upper bound for this number which is linear in n, the number of random point sites, where d is assumed to be a constant. This result matches the trivial lower bound of n.

This is an open problem since several years. In 1991, Dwyer [2] showed that for a uniform distribution from the unit d-ball the average number of Voronoi vertices is linear in n and it is commonly assumed that this holds for any reasonable probability distribution.

1 Introduction

Voronoi diagrams are a fundamental structure in several fields of science besides mathematics and computer science such as physics, geology, agriculture, geography, etc. Named after the Russian mathematician Voronoi [10] they have been 'reinvented' by other researchers, e.g., by the physicists Wigner and Seits [11] or the meteorologist Thiessen [9].

The Voronoi diagram of a set $\mathcal{S}$ of n points – called sites – partitions space into n regions, one per site. The region of a site s consists of all points that are closer to s than to any other site. The straight-line dual of the Voronoi diagram in the plane and its extension to higher dimensions is called the *Delaunay triangulation*. A triangulation of a set $\mathcal{S}$ of sites is a complete partition of the convex hull of $\mathcal{S}$ into fully dimensional simplices having the sites as vertices. The Delaunay triangulation is the unique triangulation of the set of sites such that the circumsphere of every simplex contains no other site in its interior. The Voronoi diagram can be computed in linear time from the Delaunay triangulation, using the one-to-one correspondence between their faces.

Voronoi diagrams have the great advantage to be a rather simple but quite elegant structure with many extensions obtained by varying metric, sites, environment, and constraints. In computer science they are widely used in clustering, mesh generation, graphics, curve and surface reconstruction, and other applications.

A vast variety of basic and (relatively) simple algorithms exists for their construction such as the plane sweep, the divide-and-conquer, the incremental, and the gift-wrapping algorithm, see also Chapter 20 in the Handbook of Discrete and Computational Geometry [5]. In fact most of these algorithms are actually specialized convex hull algorithms since there is a close connection with convexity. Any $(d+1)$-dimensional convex hull algorithm can be used to compute a d-dimensional Delaunay triangulation. All these algorithms depend in their run time on the number of faces of the Delaunay triangulation. Unfortunately, in d dimensions this number is $\Theta(n^{\lceil d/2 \rceil})$ in the worst case [7, 8] (for the usual diagram with the Euclidean metric).

Recent research attempts to quantify situations when the complexity of the Voronoi diagram is low or when it is high [4]. The average case complexity was considered by Dwyer [2] who showed that for n i.i.d. random point sites chosen uniformly from the unit d-ball the expected number of Delaunay simplices is $\Theta(n)$. It has been conjectured that this bound also holds for any uniform distribution in a convex domain but until now no explicit proofs were given [2, 6].

For further reading on Voronoi diagrams and Delaunay triangulations we refer to the survey by Franz Aurenhammer [1], the book by Herbert Edelsbrunner [3] and Chapter 20 in the Handbook of Discrete and Computational Geometry [5].

2 Average case

In this section we will present an average case analysis for the number of Delaunay cells. Let $\mathcal{P}$ be a set of n i.i.d. random points chosen uniformly from the unit d-hypercube $[0,1]^d$. Let $\mathbf{D}(\mathcal{P})$ be the Delaunay triangulation of $\mathcal{P}$. Generally, we will use that

$$\mathbf{E}\big[\text{number of Delaunay simplices of } \mathbf{D}(\mathcal{P})\big] = \binom{n}{d+1} \cdot \mathbf{Pr}\big[\text{ c-ball}(\Delta) \text{ is empty}\big]$$

*Research is partially supported by DFG grant 872/8-2 and by the DFG Research Training Group GK-693 of the Paderborn Institute for Scientific Computation (PaSCo) and the International Graduate School of Dynamic Intelligent Systems.

†Heinz Nixdorf Institute, Computer Science Department, University of Paderborn, 33102 Paderborn, Germany

where Δ is a random d-simplex, i.e., it is the convex hull of $d+1$ random point sites chosen uniformly from $[0,1]^d$ and c-ball(Δ) is the smallest d-ball enclosing Δ.

Unfortunately, in general it is

$$\mathbf{Pr}\big[\text{c-ball}(\Delta) \text{ is empty}\big] \neq \big(1 - \text{vol}(\text{c-ball}(\Delta))\big)^{n-(d+1)}$$

for the following reason. All random point sites are from inside $[0,1]^d$ while some part of c-ball(Δ) might lie outside of $[0,1]^d$. Of course, the probability for a random point site to be in the outer part of c-ball(Δ) is equal to 0 and therefore we must not consider the outer part. This causes the main difficulty in our analysis namely to bound the volume of c-ball$(\Delta) \cap [0,1]^d$.

Fortunately, we can show the following crucial lemma though we postpone the proof to Section 3.

Lemma 1 *Let Δ be a random d-simplex, i.e., its $d+1$ vertices are i.i.d. random points chosen uniformly from $[0,1]^d$. Then for any constant $a \in [0,1]$ it is*

$$\mathbf{Pr}\left[\text{vol}\left(\text{c-ball}(\Delta) \cap [0,1]^d\right) \leq a\right] \;\leq\; \text{const}_d \cdot a^d$$

where const_d is a constant depending only on d.

Based on this lemma we will now establish the main theorem of this section.

Theorem 2 *For n i.i.d. random points sites chosen uniformly from $[0,1]^d$ it holds that*

$$\mathbf{E}\big[\textit{number of Delaunay simplices}\big] \;=\; \mathcal{O}(n) \;.$$

Proof. The main idea of the proof is to consider (classes of) simplices with a 'large' circumball that are very likely to have another point site in their circumball, i.e., these simplices are not Delaunay simplices. Then we show that the remaining simplices with a 'small' circumball are very few.

Let us assume w.l.o.g. that n is a power of 2. Let us consider the $\binom{n}{d+1}$ possible simplices that have $d+1$ of the given n random point sites as vertices. For the simplices with 'large' circumball we define classes $\mathcal{S}_0, \ldots, \mathcal{S}_{\log n-1}$ s.t. for a simplex Δ we have that

$$\Delta \in \mathcal{S}_i \;\Leftrightarrow\; \frac{1}{2^{i+1}} \;<\; \text{vol}\left(\text{c-ball}(\Delta) \cap [0,1]^d\right) \;\leq\; \frac{1}{2^i} \;.$$

From Lemma 1 it follows immediately that

$$\begin{aligned}\mathbf{Pr}\big[\Delta \in \mathcal{S}_i\big] \;&\leq\; \mathbf{Pr}\left[\text{vol}\left(\text{c-ball}(\Delta) \cap [0,1]^d\right) \leq \frac{1}{2^i}\right] \\ &\leq\; \text{const}_d \cdot \left(\frac{1}{2^i}\right)^d \;.\end{aligned}$$

The probability for a simplex $\Delta \in \mathcal{S}_i$ to be a Delaunay simplex is

$$\begin{aligned}\mathbf{Pr}\big[\text{c-ball}(\Delta) \text{ is empty} \mid \Delta \in \mathcal{S}_i\big] \;&\leq\; \left(1 - \frac{1}{2^{i+1}}\right)^{n-(d+1)} \\ \leq\; \left(\frac{1}{e}\right)^{\frac{n-(d+1)}{2^{i+1}}} \;&\leq\; \left(\frac{1}{2}\right)^{\frac{n-(d+1)}{2^{i+1}}} \;.\end{aligned}$$

Now we can bound the expected number of Delaunay simplices for each class $\mathcal{S}_i$.

For $0 \leq i \leq \log n - 1$ it is

$$\begin{aligned}&\mathbf{E}\big[\text{number of Delaunay simplices} \in \mathcal{S}_i\big] \\ &\quad\leq\; \binom{n}{d+1} \cdot \mathbf{Pr}\big[\Delta \in \mathcal{S}_i\big] \\ &\qquad\qquad \cdot \mathbf{Pr}\big[\text{c-ball}(\Delta) \text{ is empty} \mid \Delta \in \mathcal{S}_i\big] \\ &\quad\leq\; \binom{n}{d+1} \cdot \text{const}_d \cdot \left(\frac{1}{2^i}\right)^d \cdot \left(\frac{1}{2}\right)^{\frac{n-(d+1)}{2^{i+1}}} \;.\end{aligned}$$

The expected number of Delaunay simplices for all classes $\mathcal{S}_0, \ldots, \mathcal{S}_{\log n-1}$ is

$$\begin{aligned}&\sum_{i=0}^{\log n-1} \mathbf{E}\big[\text{number of Delaunay simplices} \in \mathcal{S}_i\big] \\ &\leq\; \binom{n}{d+1} \cdot \text{const}_d \sum_{i=0}^{\log n-1} \left(\frac{1}{2}\right)^{i\cdot d + \frac{n-(d+1)}{2^{i+1}}} \\ &=\; \binom{n}{d+1} \cdot \text{const}_d \sum_{i=0}^{\log n-1} \left(\frac{1}{2}\right)^{(\log n-(i+1))\cdot d + \frac{n-(d+1)}{2^{\log n-(i+1)+1}}} \\ &=\; \binom{n}{d+1} \cdot \text{const}_d \cdot \frac{1}{n^d} \sum_{i=0}^{\log n-1} \left(\frac{1}{2}\right)^{2^i\cdot\left(1-\frac{d+1}{n}\right)-(i+1)\cdot d} \\ &\leq\; n \cdot \text{const}_d \cdot \Big((d+2) \cdot 2^{(d+3)\cdot d} + 1\Big) \;=\; \mathcal{O}(n) \;.\end{aligned}$$

The last step follows immediately if $d+2 \geq \log n - 1$ since

$$\begin{aligned}\sum_{i=0}^{d+2} \left(\frac{1}{2}\right)^{2^i\cdot\left(1-\frac{d+1}{n}\right)-(i+1)\cdot d} \;&\leq\; \sum_{i=0}^{d+2} 2^{(i+1)\cdot d} \\ &\leq\; (d+2) \cdot 2^{(d+3)\cdot d} \;.\end{aligned}$$

In the other case it is

$$\sum_{i=d+3}^{\log n-1} \left(\frac{1}{2}\right)^{2^i\cdot\left(1-\frac{d+1}{n}\right)-(i+1)\cdot d} \;\leq\; \sum_{i=d+3}^{\log n-1} \left(\frac{1}{2}\right)^i \;\leq\; 1 \;,$$

where we assume that $n \geq 2 \cdot (d+1)$.

The expected number of remaining simplices with 'small' circumball can be bounded using lemma 1, too. Let $\mathcal{S}_{\text{re}}$ denote the set of simplices s.t. for a simplex Δ we have

$$\Delta \in \mathcal{S}_{\text{re}} \;\Leftrightarrow\; \text{vol}\left(\text{c-ball}(\Delta) \cap [0,1]^d\right) \;\leq\; \frac{1}{n} \;.$$

Then it is

$$\begin{aligned}&\mathbf{E}\big[\text{number of simplices } \in \mathcal{S}_{\text{re}}\big] \\ &\quad\leq\; \binom{n}{d+1} \cdot \mathbf{Pr}\left[\text{vol}\left(\text{c-ball}(\Delta) \cap [0,1]^d\right) \leq \frac{1}{n}\right] \\ &\quad\leq\; n^{d+1} \cdot \text{const}_d \cdot \frac{1}{n^d} \;\;\leq\;\; n \cdot \text{const}_d \;.\end{aligned}$$

Now we can combine everything and by linearity of expectation we get that

$$\begin{aligned}&\mathbf{E}\big[\text{number of Delaunay cells}\big]\\ &\leq \sum_{i=0}^{\log n-1} \mathbf{E}\big[\text{number of Delaunay simplices} \in \mathcal{S}_i\big]\\ &\qquad\qquad + \mathbf{E}\big[\text{number of simplices } \in \mathcal{S}_{\text{re}}\big]\\ &\leq n\cdot \text{const}_d \cdot \Big((d+2)^{(d+3)\cdot d}+2\Big) = \mathcal{O}(n)\ ,\end{aligned}$$

which concludes the proof of Theorem 2. □

3 Proof of Lemma 1

Let $p_1,\ldots,p_{d+1} \in [0,1]^d$ be the vertices of simplex $\Delta = \Delta(p_1,\ldots,p_{d+1})$, i.e., Δ is the convex hull of $p_1,\ldots,p_{d+1}$. The volume of c-ball(Δ) is given by $\mathcal{V}_d \cdot r^d$ where $r = r(\Delta)$ is the radius of the circumball of Δ and $\mathcal{V}_d = \frac{\pi^{d/2}}{\Gamma(1+d/2)}$ is the volume of the unit d-ball. We can approximate the radius $r(\Delta)$ and the volume of c-ball(Δ) by the following observation:

Observation 1 *It holds that*

$$\begin{aligned}2\cdot r(\Delta) &\geq \max_{1\leq i<j\leq d+1} \|p_i - p_j\|_2\\ &\geq \max_{1\leq i<j\leq d+1} \|p_i-p_j\|_\infty\\ &=: \text{maxwidth}(\Delta)\end{aligned}$$

and therefore it is

$$\text{vol}\left(\text{c-ball}(\Delta)\right) \geq \mathcal{V}_d \cdot \frac{1}{2^d}\cdot \text{maxwidth}(\Delta)^d\ .$$

In other words we approximate the volume of c-ball(Δ) by a fraction of the volume of a smallest hypercube containing all the point sites $p_1,\ldots,p_{d+1}$, cf. Figure 1.

In a next step we will reformulate our random process. Instead of considering $d+1$ many d-dimensional random variables (= point sites) we will combine the random variables coordinate-wise leading to d sets of $d+1$ random numbers each. In more detail, let us again consider the point sites $p_1,\ldots,p_{d+1} \in [0,1]^d$ where $p_i = (p_i^{(1)},\ldots,p_i^{(d)})$ for $1\leq i\leq d+1$. Let $\mathcal{P}_1,\ldots,\mathcal{P}_d$ be the sets s.t. $\mathcal{P}_j = \{p_1^{(j)},\ldots,p_{d+1}^{(j)}\}$ for $1\leq j\leq d$ and let

$$\text{width}(\mathcal{P}_j) := \max \mathcal{P}_j - \min \mathcal{P}_j$$

denote the maximal distance between two elements in $\mathcal{P}_j$. We can now define the variable maxwidth in another way as

$$\text{maxwidth}(\mathcal{P}_1,\ldots,\mathcal{P}_d) := \max_{1\leq j\leq d} \text{width}(\mathcal{P}_j)\ ,$$

which is consistent with the earlier definition, i.e., it is maxwidth(Δ) = maxwidth($\mathcal{P}_1,\ldots,\mathcal{P}_d$). (Therefore we sometimes write only maxwidth.)

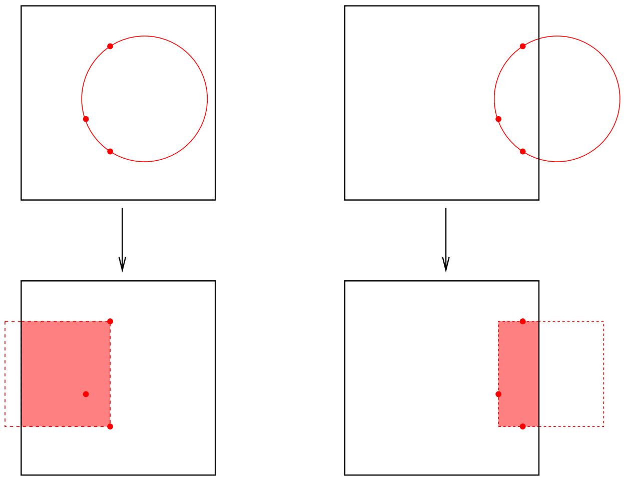


Figure 1: The 2 dimensional case: 3 point sites and their circumcircles in the unit square. Generally, the position of the smallest hypercube containing all point sites is not uniquely defined. When intersected by $[0,1]^d$ consider the hypercube that has smallest intersection volume.

Since we actually want to bound the volume of c-ball(Δ) $\cap [0,1]^d$, we consider the (smallest) hypercube containing all point sites that has minimal volume when intersected by $[0,1]^d$. Therefore, we introduce the variable value that indicates how much each dimension contributes to the volume of the minimal intersection between a smallest hypercube containing all the point sites and $[0,1]^d$. If for a fixed dimension the coordinates of all point sites lie close to 0 (or 1) then the dimension contributes less than maxwidth to the volume, namely only the distance of the maximal coordinate to 0 (or the minimal coordinate to 1), cf. also figure 1. In other words, the hypercube then sticks out of $[0,1]^d$ in this dimension.

We define now the value of set $\mathcal{P}_j$ to be

$$\text{value}(\mathcal{P}_j) := \begin{cases} \text{maxwidth}(\mathcal{P}_1,\ldots,\mathcal{P}_d) & \\ \qquad \textbf{if } \max\mathcal{P}_j - \text{maxwidth} \geq 0 \\ \qquad \text{and } \min\mathcal{P}_j + \text{maxwidth} \leq 1 & \\ \min\{\max\mathcal{P}_j, 1-\min\mathcal{P}_j\} & \textbf{else}\ .\end{cases}$$

With these definitions we can formulate the following lemma.

Lemma 3 *It holds that*

$$\begin{aligned}&\text{vol}\left(\text{c-ball}(\Delta)\cap[0,1]^d\right)\\ &\qquad \geq \min\left\{\mathcal{V}_d\cdot\left(\frac{1}{2}\right)^{2d}, \frac{1}{d!}\right\}\cdot \prod_{j=1}^{d} \text{value}(\mathcal{P}_j)\ .\end{aligned}$$

Due to space limitations we defer the proof of Lemma 3 to a later full version of this paper.

From now on we will consider the following random process: we have d sets $\mathcal{P}_1, \ldots, \mathcal{P}_d$ of $d+1$ i.i.d. random numbers chosen uniformly from the interval $[0,1]$. From Lemma 3 it follows that if we show that

$$\mathbf{Pr}\left[\prod_{j=1}^{d} \text{value}(\mathcal{P}_j) \leq a\right] = \mathcal{O}\left(a^d\right) \ , \qquad (1)$$

Lemma 1 is also shown.

In order to show (1) we will now establish two lemmas. The first one covers the case that maxwidth is smaller than $\sqrt[d]{a}$, then it follows immediately that $\prod_{j=1}^{d} \text{value}(\mathcal{P}_j) \leq a$. The second lemma covers the case that maxwidth is larger than $\sqrt[d]{a}$ and we have to spend some more effort to show (1).

Lemma 4 *For any value $a \in [0,1]$ it holds that*

$$\mathbf{Pr}\left[\text{maxwidth}(\mathcal{P}_1, \ldots, \mathcal{P}_d) \leq \sqrt[d]{a}\right] \leq \mathcal{O}\left(a^d\right) \ .$$

Proof. It suffices to bound the probability that $\text{width}(\mathcal{P}_j) \leq \sqrt[d]{a}$ for $1 \leq j \leq d$. For set $\mathcal{P}_j$ we fix the two elements with the maximal distance, i.e., we fix $\max \mathcal{P}_j$ and $\min \mathcal{P}_j$ where the distance between both mustn't exceed $\sqrt[d]{a}$. The remaining $d-1$ elements in $\mathcal{P}_j$ must have values between $\max \mathcal{P}_j$ and $\min \mathcal{P}_j$. Now we can write

$$\mathbf{Pr}\left[\text{width}(\mathcal{P}_j) \leq \sqrt[d]{a}\right] \leq$$
$$(d+1) \cdot d \cdot \int_0^1 \int_{\max\{0, Y-\sqrt[d]{a}\}}^{Y} (Y-X)^{d-1} \ \mathrm{d}X \ \mathrm{d}Y \qquad (2)$$

where the outer intergral denotes the range of element $\max \mathcal{P}_j (= Y)$ and the inner integral the range of element $\min \mathcal{P}_j (= X)$. The integration boundaries assure that their distance is at most $\sqrt[d]{a}$. The integrand $(Y-X)^{d-1}$ denotes exactly the probability that all remaining $d-1$ elements of $\mathcal{P}_j$ are between Y and X. The factor before the integral is due to fixing the maximal and minimal element in $\mathcal{P}_j$.

In order to solve this integral we will split it up in the following way to remove the maximum expression from the integration boundary of the inner integral.

$$\begin{aligned}
& \int_0^1 \int_{\max\{0, Y-\sqrt[d]{a}\}}^{Y} (Y-X)^{d-1} \ \mathrm{d}X \ \mathrm{d}Y \\
= \ & \int_0^{\sqrt[d]{a}} \int_0^{Y} (Y-X)^{d-1} \ \mathrm{d}X \ \mathrm{d}Y \\
& + \int_{\sqrt[d]{a}}^{1} \int_{Y-\sqrt[d]{a}}^{Y} (Y-X)^{d-1} \ \mathrm{d}X \ \mathrm{d}Y \\
= \ & \frac{1}{d} \cdot \left(\int_0^{\sqrt[d]{a}} Y^d \ \mathrm{d}Y + \int_{\sqrt[d]{a}}^{1} a \ \mathrm{d}Y \right) \\
= \ & \frac{1}{d} \cdot \left(\frac{1}{d+1} \cdot a^{\frac{d+1}{d}} + a - a^{\frac{d+1}{d}} \right) \leq \frac{1}{d} \cdot a
\end{aligned}$$

It follows that

$$\mathbf{Pr}\left[\text{width}(\mathcal{P}_j) \leq \sqrt[d]{a}\right] \leq (d+1) \cdot a \quad \Rightarrow$$
$$\mathbf{Pr}\left[\text{maxwidth} \leq \sqrt[d]{a}\right] \leq \mathcal{O}\left(a^d\right) \ .$$

□

Lemma 5 *For any value of $a \in [0,1]$ it holds that*

$$\mathbf{Pr}\Big[\text{maxwidth}(\mathcal{P}_1, \ldots, \mathcal{P}_d) > \sqrt[d]{a} \quad \textit{and}$$
$$\prod_{j=1}^{d} \text{value}(\mathcal{P}_j) \leq a\Big] \leq \mathcal{O}\left(a^d\right) \ .$$

The proof of Lemma 5 is very involved and rather lengthy. We defer it also to a full version of this paper.

From Lemma 4 and Lemma 5 it follows that Equation (1) holds and thus Lemma 1 is shown.

References

[1] Aurenhammer, F. Voronoi diagrams – a survey of a fundamental geometric data structure. *ACM Comput. Surv. 23* (1991), pp. 345–405.

[2] Dwyer, R. A. Higher-dimensional Voronoi diagrams in linear expected time. In *Proceedings of the 5th Annual ACM Symposium on Computational Geometry* (1989), pp. 326–333.

[3] Edelsbrunner, H. *Algorithms in Combinatorial geometry.* Springer Verlag, Berlin, 1987.

[4] Erickson, J. Nice point sets can have nasty Delaunay triangulations. In *Proceedings of the 17th Annual ACM Symposium on Computational Geometry* (2001), pp. 96–105.

[5] Goodmann J. E. and O'Rourke, J. *Handbook of Discrete and Computational Geometry.* CRC Press LLC, 1997.

[6] Golin, M. J. and Na, H.-S. On the average complexity of 3D-Voronoi diagrams of random points on convex polytopes. *Computational Geometry 25* (2003), pp. 197 231.

[7] Klee, V. On the complexity of d-dimensional Voronoi diagrams. *Archiv. Math. 34* (1980), pp. 75–80.

[8] Seidel, R. On the number of faces in higher-dimensional Voronoi diagrams. In *Proceedings of the 3rd Annual ACM Symposium on Computational Geometry* (1987), pp. 181–185.

[9] Thiessen, A. H. Precipitation averages for large areas. *Monthly Weather Review 39* (1911), pp. 1082–1084.

[10] Voronoi, G. Nouvelles applications des parametres continus à la theorie des formes quadratique, deuxième mémoire: recherches sur les parallélloèdres primitifs. *J. Reine und Angewandte Mathematik 134* (1908), pp. 198–287.

[11] Wigner, E. and Seits, F. On the constitution of metallic sodium. *Physical Reviews 43* (1933), pp. 804–810.

Lower Bounds for Kinetic Sorting

Mohammad Ali Abam[*] Mark de Berg[†]

Abstract

Let S be a set of n points moving on the real line. The kinetic sorting problem is to maintain a data structure on the set S that makes it possible to quickly generate a sorted list of the points in S, at any given time. We prove tight lower bounds for this problem, which show the following: with a subquadratic maintenance cost one cannot obtain any significant speed-up on the time needed to generate the sorted list (compared to the trivial $O(n \log n)$ time), even for linear motions.

1 Introduction

Background. In many areas of computer science one has to store, analyze and manipulate geometric data. More and more often this involves objects in motion. Hence, the study of geometric data structures for moving objects has recently attracted a lot of attention in computational geometry, especially since Basch *et al.* [6] introduced the *kinetic-data-structure* (KDS, for short) framework [3, 4, 6, 8, 9].

A KDS is a structure that maintains a certain attribute of a set of continuously moving objects—the convex hull of moving objects, for instance, or the closest distance among moving objects. It consists of two parts: a combinatorial description of the attribute and a set of certificates with the property that as long as the outcomes of the certificates do not change, the attribute does not change. It is assumed that each object follows a known trajectory so that one can compute the failure time of each certificate. Whenever a certificate fails—we call this an *event*—the KDS must be updated. The KDS remains valid until the next event. See the excellent survey by Guibas [8] for more background on KDSs and their analysis.

Computing the convex hull of a set of points in the plane is a classic problem in computational geometry. It is therefore not surprising that the kinetic maintenance of the convex hull of a set of n moving points in the plane was already studied by Basch *et al.* [6]. They designed a KDS that needs to be updated $O(n^{2+\epsilon})$ times and each time takes $O(\log^2 n)$ time.

[*]Department of Computing Science, TU Eindhoven, m.a.abam@tue.nl

[†]Department of Computing Science, TU Eindhoven, mdberg@win.tue.nl

In some applications it may be necessary to maintain the attribute of interest explicitly. If one uses a KDS for collision detection, for instance, any external event—a collision in this case— must be reported. In such cases, the number of changes to the attribute is a lower bound on the number of events to be processed. Since the convex hull of n linearly moving points can change $\Omega(n^2)$ times [5], this means that any KDS that maintains an explicit representatie of the convex hull must process $\Omega(n^2)$ events in the worst case. Hence, the convex-hull KDS of Basch *et al.* [6], which indeed maintains the convex hull explicitly, is close to optimal in the worst case.

In other applications, however, explicitly maintaining the attribute at all times may not be necessary; the attribute is only needed at certain times. This is for instance the case when a KDS is used as an auxiliary structure in another KDS. The auxiliary KDS is then used to update the main KDS efficiently when a certificate of the main KDS fails. In this case, even though the main KDS may have to be maintained explicitly, the attribute maintained by the auxiliary KDS only needs to be available at certain times. This leads us to view a KDS as a query structure: we want to maintain a set S of moving objects in such a way that we can reconstruct the attribute of interest efficiently whenever this is called for. This makes it possible to reduce the maintenance cost, as it is no longer necessary to update the KDS whenever the attribute changes. On the other hand, a reduction in maintenance cost will have an impact on the query time, that is, the time needed to reconstruct the attribute. Thus there is a trade-off between maintenance cost and query time. Our main goal is to study such trade-offs for kinetic convex hulls.

Our results. As stated above, our main interest lies in trade-offs between the maintenance cost of a kinetic convex-hull structure and the time to reconstruct the convex hull at any given time. In this short abstract, however, we restrict our attention to the simpler *kinetic sorting problem*: maintain a KDS on a set of n points moving on the real line such that at any time we can quickly reconstruct a sorted list of the points. We prove in Section 2 that already for the kinetic sorting problem one cannot get good trade-offs: even for linear motions, the worst-case maintenance cost is $\Omega(n^2)$ if one wants to be able to do the reconstruction in $o(n)$ time. Note that with $\Omega(n^2)$ maintenance cost, we can explicitly maintain the sorted list at all times, so that the reconstruction cost is zero. Thus interesting trade-offs are only possible in a very limited range of the spectrum, namely for reconstruction costs between $\Omega(n)$ and $O(n \log n)$. For this range we also prove lower bounds: we show that one needs $\Omega(n^2/m)$ maintenance cost if one wants to achieve $o(n \log m)$ reconstruction cost, for any m with $2 \leq m \leq n$. (See Section 2.1 for a definition of our lower-bound model.) We also give a matching upper bound.

Related work. Some existing KDSs—the kinetic variants of various range-searching data structures [2, 3, 4, 9], for instance—do not maintain a uniquely defined attribute such as the convex hull, but they maintain a query data structure. In this setting the KDS is, of course, a query structure as well. Our setting is different because we are studying the maintenance of a single, uniquely defined, attribute such as the convex hull.

One of the main results of our paper is a lower bound on the trade-offs between reconstruction time and maintenance cost for the kinetic sorting problem. Lower bounds for trade-offs between query time and maintenance cost were also given by De Berg [7], but he studied the kinetic dictionary problem, where one wants to maintain a dictionary on a set S of n points moving on the real line. He showed that any kinetic dictionary with worst-case query time $O(Q)$ must have a worst-case total maintenance cost of $\Omega(n^2/Q^2)$, even if the points move linearly.

2 The kinetic sorting problem

Let $S = \{x_1, \cdots, x_n\}$ be a set of n point objects[1] moving continuously on the real line. In other words, the value of x_i is a continuous function of time, which we denote by $x_i(t)$. We define $S(t) = \{x_1(t), \cdots, x_n(t)\}$. For simplicity, we write S and x_i instead of $S(t)$ and $x_i(t)$, respectively, provided that no confusion arises. The kinetic sorting problem asks to maintain a structure on S such that at any given time t we can quickly generate a sorted list for $S(t)$. We call such a structure a *sorting KDS*.

We focus on trade-offs between the sorting cost and the maintenance cost: what is the worst-case maintenance cost if we want to guarantee a sorting cost of $O(Q)$, where Q is some parameter, under the assumption that the point objects follow trajectories that can be described by bounded-degree polynomials.

2.1 The lower-bound model

We shall prove our lower bounds for the kinetic sorting problem in the comparison-graph model introduced by De Berg [7], which is defined as follows. A *comparison graph* for a set S of numbers is defined as a directed graph $\mathcal{G}(S, A)$ such that if $(x_i, x_j) \in A$, then $x_i < x_j$. The reverse is not true: the fact that $x_i < x_j$ does not mean there must be an arc in $\mathcal{G}$. The idea is that the comparison graph represents the ordering information encoded in a sorting KDS on the set S: if $(x_i, x_j) \in A$, then the fact that $x_i < x_j$ can be derived from the information stored in the KDS, without doing any additional comparisons.

Maintenance cost. The operations we allow on the comparison graph are insertions and deletions of arcs. For the maintenance cost, we only charge the algorithm for insertions of arcs; deletions are free. Following De Berg [7], we therefore define the maintenance cost as the total number of such arcs ever inserted into the comparison graph, either at initialization or during maintenance operations.

We say that the arc $(x_i, x_j) \in A$ *fails* at time t if $x_i(t) = x_j(t)$.The arcs in the comparison graph essentially act as certificates, and their failures trigger events at which the KDS needs to be updated.

Query cost. A query at time t asks to construct a sorted list on the points in the current set S (that is, $S(t)$). We shall consider two different measures for the query cost.

The comparison-graph sorting model. The first measure is in a very weak model, where we only charge for the minimum number of comparisons needed to obtain a sorted list, assuming we have an oracle at our disposal telling us exactly which comparisons to do. This is similar to the query cost used by De Berg when he proved lower bounds for the kinetic dictionary. For the sorting problem this simply means that the query cost is equal to the number of pairs $x_i, x_j \in S$ that are adjacent in the ordering and for which there is no arc in the comparison graph.

The algebraic decision-tree model. In this model we also count the number of comparisons needed to sort the set S, but this time we not have an oracle telling us which comparisons to do. We shall use the following basic fact: Suppose the number of different orderings of S that are compatible with the comparison graph at some given time is N. Then the cost to sort S in the algebraic decision-tree model is at least $\log N$.

2.2 A lower bound in the comparison-graph sorting model

The point objects in our lower-bound instance will move with constant (but different) velocities on the real line. Hence, if we view the line on which the point objects move as the x-axis and time as the t-axis, then the trajectories of the point objects are straight lines in the tx-plane. We use ξ_i to denote the line in the

[1]We use the term "point objects" for the points in S to distinguish them from other points that play a role in our proofs.

tx-plane that is the trajectory of x_i. It is somewhat easier to describe the lower-bound instance in the dual plane. We shall call the two axes in the dual plane the u-axis and the v-axis. We use the standard duality transform, where a line $\xi : x = at + b$ in the tx-plane is mapped to the point $\xi^* : (a, -b)$ in the dual plane, and a point $p : (a, b)$ in the primal plane is mapped to the line $p^* : v = au - b$ in the dual plane.

Now let $p_1, \ldots, p_n$ be the vertices of a regular n-gon in the dual plane that is oriented such that the diagonal $p_{l-1}p_{l+1}$ connecting the two neighbors of the leftmost vertex p_l is almost parallel to the v-axis and has negative slope. The trajectories $\xi_1, \cdots, \xi_n$ in our lower-bound instance are the primals of the vertices p_i, that is, $\xi_i^* = p_i$. In the remainder of this section we will prove a lower bound on the maintenance cost of any comparison graph for this instance whose sorting cost (in the comparison-graph sorting model) is bounded by Q, where Q is a parameter with $0 \leq Q < n$.

For any pair of vertices p_i, p_j, let ℓ_{ij} denote the line passing through p_i and p_j. Since the p_i are the vertices of a regular n-gon, the lines ℓ_{ij} have only n distinct slopes. Note that ℓ_{ij} corresponds to the intersection of ξ_i and ξ_j in the tx-plane, with the slope of ℓ_{ij} being equal to t-coordinate of the intersection. This implies that the intersection points of the trajectories in the tx-plane have only n distinct t-values. Let $t_1, \cdots, t_n$ be the sorted sequence of these t-values. The times $t_1, \ldots, t_n$ define $n + 1$ open time intervals $(-\infty, t_1), (t_1, t_2), \cdots, (t_n, +\infty)$. Since no two trajectories intersect inside any of these intervals, the order of the point objects is the same throughout any interval. We say that x_i is *directly below* x_j in such an interval if $x_i(t) < x_j(t)$ for times t in the interval and there is no other point object x_k in between them in that interval. Furthermore, we call a vertex p_i a *lower vertex* if it lies on the lower part of the boundary of the n-gon, and we call p_i an *upper vertex* if it lies on the upper part of the boundary of the n-gon; the leftmost and rightmost vertices are neither upper nor lower vertices.

Lemma 1 *(i) If p_i is a lower vertex or the leftmost vertex, then the object x_i is below any other object x_j in exactly one time interval. If n is odd, this also holds for the rightmost vertex.*
(ii) If p_i is an upper vertex, then x_i is directly below any other point object x_j in at least one interval. If n is even, this also holds for the rightmost vertex.

We can now prove the lower bound. Suppose that we have a comparison graph on the point objects whose sorting cost is Q during each of the time intervals defined above. This implies that during each such time interval, there must be at least $n - Q - 1$ arcs (x_i, x_j) in the comparison graph such that x_i is directly below x_j. In total, $(n + 1)(n - Q - 1)$ arcs are needed over all $n + 1$ time interval. Some arcs, however, can be used in more than one interval. For x_i, let k_i be the number of arcs of the form (x_i, x_j) that are used. For any of the $\lceil n/2 \rceil$ objects x_i for which case (i) of Lemma 1 applies, all these arcs are distinct. For the remaining $\lfloor n/2 \rfloor$ objects case (ii) applies and so at least $k_i - 2$ arcs are distinct. Hence, the total number of arcs inserted over time is at least $(n + 1)(n - Q - 1) - 2\lfloor n/2 \rfloor \geq n(n - Q - 2)$. We get the following theorem.

Theorem 2 *There is an instance of n point objects moving with constant velocities on the real line, such that any comparison graph whose worst-case sorting cost in the comparison-graph cost model is Q, must have maintenance cost at least $n(n - Q - 2)$, for any parameter Q with $0 \leq Q < n$.*

2.3 A lower bound in the algebraic decision-tree model

In the previous section we gave a lower bound for the maintenance cost for a given sorting cost Q in comparison-graph sorting model. Obviously, this is also a lower bound for the algebraic decision-tree model. Hence, the results of the previous section imply that for any sorting cost $Q = o(n)$ in the algebraic decision-tree model, the worst-case maintenance cost is $\Omega(n^2)$. Since with $O(n^2)$ maintenance cost we can process all swaps—assuming the trajectories are bounded-degree algebraic, so that any pair swaps at most $O(1)$ times—this bound is tight: with $O(n^2)$ maintenance cost we can achieve sorting cost zero. What remains is to investigate the range where the sorting cost is $o(n \log m)$, where $1 < m \leq n$.

Lemma 3 *There is a constant c such that if the sorting cost of a comparison graph is at most $cn \log m$, then there is a path in the comparison graph whose length is at least $n/m^{1/3}$.*

Next we describe the lower bound construction. As before, it will be convenient to describe the construction in the dual plane. To this end, let $G_a := \{0, 1, \cdots, a - 1\}^2$ be the $a \times a$ grid. The trajectories of the point objects in our lower-bound instance will be straight lines in the tx-plane, such that the duals of these lines are the grid points of $G_{\sqrt{n}}$. (We assume for simplicity that n is a square number.) Before we proceed, we need the following lemma.

Lemma 4 *Let $p = (p_x, p_y)$ be a grid point of $G_{\sqrt{n}}$ and $p_x, p_y \leq a$, where $a \leq \sqrt{n}/2$. Let ℓ_p be the line through the origin and p. The number of different lines passing through at least one point of $G_{\sqrt{n}}$ and being parallel to ℓ_p is at most $4a\sqrt{n}$.*

Theorem 5 *There is an instance of n point objects moving with constant velocities on the real line such that, for any m with $1 < m \leq n$, any comparison graph whose worst-case sorting cost in the algebraic decision-tree model is $Q = o(n \log m)$, must have maintenance cost $\Omega(n^2/m)$.*

Proof. Let $a := \sqrt{n}/(8m^{1/3})$. Consider the comparison graph at some time $s + \varepsilon$ with $s = p_x/p_y$, where $p_x, p_y \leq a$ and $\varepsilon > 0$ is sufficiently small. Suppose the sorting cost at time s is $o(n \log m)$. Then the sorting cost will be at most $cn \log m$ for any constant c, so by Lemma 3 there must be a path in the comparison graph of length at least $n/m^{1/3}$. We claim (and will prove below) that at least half of the arcs in this path are between point objects x_i, x_j such that ξ_i^* and ξ_j^* lie on a common line of slope s. The number of distinct values for s is equal to the number of pairs (p_x, p_y) where p_x and p_y are integer numbers between 0 and $a-1$ (including 0 and $a-1$) and $GCD(p_x, p_y) = 1$. Because of symmetry, we count the number of pairs (p_x, p_y) with the property $p_x \leq p_y$. For a nonnegative integer i, let $\varphi(i)$ be the number of nonnegative integers that are less than i and relatively prime to i. Then the number of pairs (p_x, p_y) with the desired properties is $\sum_{i=1}^{a-1} \varphi(i)$. It is known [10] that this summation is $\Theta(a^2)$. Then, the total number of arcs needed over all times of the form $p_x/p_y + \varepsilon$ with $p_x, p_y \leq a$ is at least $n/(2m^{1/3}) \cdot \Theta(a^2) = \Omega(n^2/m)$, which proves the theorem.

It remains to prove the claim that at least half of the arcs in the path are between point objects x_i, x_j such that ξ_i^* and ξ_j^* lie on a common line of slope s. Note that the sorted order of the point objects $x_i(s+\varepsilon)$ corresponds to the sorted order of the orthogonal projections of the points ξ_i^* onto a line with slope $-1/(s+\varepsilon)$. If $\varepsilon > 0$ is sufficiently small, then the projections of all the points lying on a common line of slope s will be adjacent in this order. Let's group the point objects x_i into subsets such that any two point objects x_i, x_j for which ξ_i^* and ξ_j^* lie on a common line of slope s are in the same subset. Then, at time $s+\varepsilon$, any path in the comparison graph can enter and leave a subset at most once. By Lemma 4 the number of subsets is at most $4a\sqrt{n}$. Hence, the number of arcs connecting point objects in the same subset is at least

$$n/m^{1/3} - 4a\sqrt{n} \quad = \quad n/(2m^{1/3}),$$

as claimed. □

2.4 Upper bounds for the kinetic sorting problem

It is straightforward to obtain a KDS that matches the lower bounds of the previous section: Partition the set S into m subsets of size at most n/m in an arbitrary manner, and maintain each subset in a sorted array. This gives the following result.

Theorem 6 *Let S be a set of n point objects moving on the line, where any pair of points swaps $O(1)$ times. For any m with $1 < m \leq n$, there is a data structure with maintenance cost $O(n^2/m)$ such that at any time a sorted list of the points in S can be constructed in $O(n \log m)$ time.*

3 Conclusions

We have studied trade-offs for the kinetic sorting problem, which is to maintain a KDS on a set of points moving on the real line such that one can quickly generate a sorted list of the points, at any given time. We have proved a lower bound for this problem showing the following: with a subquadratic maintenance cost one cannot obtain any significant speed-up on the time needed to generate the sorted list (compared to the trivial $O(n \log n)$ time), even for linear motions.

This negative result gives a strong indication that good trade-offs are not possible for a large number of geometric problems—Voronoi diagrams and Delaunay triangulations, for example, or convex hulls—as the sorting problem can often be reduced to such problems. In [1], we show that the convex hull can be maintained more efficiently if it has only few vertices.

References

[1] M.A Abam and Mark de Berg. Kinetic sorting and kientic convex hulls. *Submited to ACM Sympos. on Computational Geometry*, 2005.

[2] P. Agarwal, L. Arge, and J. Erickson. Indexing moving points. In *Proc. ACM Sympos. Principles Database Syst.*, pages 175–186, 2000.

[3] P. Agarwal, L. Arge, J. Erickson, and H. Yu. Efficient tradeoff schemes in data structures for querying moving objects. In *Proc. 12th European Sympos. on Algorithms*, pages 4–15, 2004.

[4] P. Agarwal, J. Gao, and L. Guibas. Kinetic medians and kd-trees. In *Proc. 10th European Sympos. on Algorithms*, pages 5–16, 2002.

[5] P. Agarwal, L. Guibas, J. Hershberger, and E. Veach. Maintaining the extent of a moving point set. In *Proc. 5th Workshop Algorithms and Data Structurs*, pages 31–44, 1997.

[6] J. Bash, L. Guibas, and J. Hershberger. Data structure for mobile data. *J. Algorithms*, 31:1–28, 1999.

[7] M. de Berg. Kinetic dictionaries: How to shoot a moving target. In *Proc. 11th European Sympos. on Algorithms*, pages 172–183, 2003.

[8] L. Guibas. Kinetic data structure: a state of art report. In *Proc. 3rd Workshop Algorithmic Found. Robot.*, pages 191–209, 1998.

[9] J. Hershberger and S. Suri. Kinetic connectivity of rectangles. In *Proc. 15th ACM Sympos. Comput. Geom.*, pages 237–246, 1999.

[10] E. Weinstein. *CRC Concise Encyclopedia of Mathematics.* CRC Press, 1999.

Uncertainty Envelopes [*]

Yaron Ostrovsky-Berman Leo Joskowicz

Abstract

We introduce a new class of problems: computing the set of all the convex combinations of sites when their position is uncertain and depends linearly on shared parameters which vary according to a uniform distribution. The boundary of the set, called the uncertainty envelope, is useful in optimizing processes where there is uncertainty on the site positions and the objective functions. We provide upper bounds on the combinatorial complexity of the uncertainty envelope, and present the first efficient algorithm for its computation in the general case.

1 Introduction

Suppose an investor wishes to spend his budget on importing several products with fixed attributes (market value, import tax, maintenance, transportation costs, ...). Choosing a combination of products which maximizes his profit function is a standard optimization problem. However, when the product attributes are subject to change due to variation in market parameters (currency exchange rates, fuel and insurance costs, ...), there is uncertainty involved in any investment. The problem then becomes that of finding the best tradeoff between profit and risk.

This scenario is common to many fields involving uncertainty or change with the following model properties:

- *proportionality* - the attributes of the products are proportional to their portion of the budget.
- *divisibility* - the portion of the budget for each product is allowed to assume any fractional value.
- *linearity* - the attributes depend linearly on the market parameters.
- *independency* - the market parameters vary independently.

For optimization problems, the following additional properties hold:

- *uniformity* - the parameters vary according to a uniform distribution with fixed intervals.
- *multiple fuzzy objective functions* - there are multiple choices of the objective function form, where function coefficients have uncertainty that varies in a range.

[*]School of Engineering and Computer Science, The Hebrew University of Jerusalem, {yaronber,josko}@cs.huji.ac.il

We formalize the problem as follows. Let $S = \{s_1, s_2, \ldots, s_n\}$ be a set of sites (products, materials, components), and let $p = (p_1, p_2, \ldots, p_m)^T$ be the market parameters uncertainty vector, which is limited to the domain $\Delta = [-\delta_1, \delta_1] \times [-\delta_2, \delta_2] \times \ldots \times [-\delta_m, \delta_m]$. Each site s_i is associated with a nominal (current) value $b_i \in \mathbb{R}^d$ and an $m \times d$ sensitivity matrix A_i describing the attribute sensitivity to variations in p. The position of site s_i is given by $v_i(p) = A_i p + b_i$. We define the *uncertainty envelope* as the boundary of the union of the convex hulls of the sites when the uncertainty vector spans its domain:

$$\mathcal{E} = \partial\{\sum_{i=1}^{n} \lambda_i v_i(p) | \lambda_i \geq 0, \sum_{i=1}^{n} \lambda_i = 1, p \in \Delta\}$$

The uncertainty envelope provides a global view of the problem domain which is most useful in analyzing worst case variation of unknown or uncertain combination of the sites. Since there are possibly several objective functions for optimization, and each function may have uncertainty, the envelope allows sensitivity analysis on each of the solutions.

The class of problems we study depends on two key parameters: the number of sites n and the dimension d. In the following special cases, the problem has known solutions. When there is no uncertainty ($\Delta = \{0\}$), the envelope bounds the convex hull of $\{b_i\}_{i=1}^{n}$, the domain of the standard optimization problem. In one dimension ($d = 1$, any n), the envelope is a segment whose endpoints are determined by the maximum and minimum of $\{b_i \pm a_j^i \delta_j\}$, where a_j^i is the j^{th} entry of A_i. When there is only one site ($n = 1$, any d), the envelope bounds the d-zonotope defined by the sensitivity matrix A and the uncertainty domain Δ. Zonotopes play a crucial role in analyzing and computing uncertainty envelopes. When the sensitivity matrices of the sites are independent, that is the indices of their non-zero column vectors are mutually exclusive, the uncertainty envelope is the boundary of the convex hull of the uncertainty zonotopes of the sites. In the general case ($n, d \geq 2$), the envelope is the boundary of the volume swept by the zonotope as it spans the convex combinations of all the sites. For a solution to the special case $d = 2$, $n = 2$ see [4].

In this extended abstract, we provide upper bounds on the combinatorial complexity of uncertainty envelopes and present the first algorithm for their computation in the general case. The results are sum-

$dim \backslash \#sites$		1	2	3	$n > 3$
1	comb	$O(1)$			$O(1)$
	time	$O(m)$			$O(nm)$
2	comb	$\Theta(m)$	$O(\lambda_6(m^2))$		$O(n^2\lambda_6(n^2m^2))$
	time	$O(m \log m)$	$O(\lambda_6(m^2)\log(m))$		$O(n^2\lambda_6(n^2m^2))$
3	comb	$\Theta(m^2)$	$O(m^{6+\epsilon})$	$O(m^{8+\epsilon})$	$O((n^3m^4)^{2+\epsilon})$
	time	$\Theta(m^2)$	$O(m^{6+\epsilon})$	$O(m^{8+\epsilon})$	$O((n^3m^4)^{2+\epsilon})$
$d > 3$	comb	$\Theta(m^{d-1})$	$n \leq d: \ O(d^d m^{d+n-2})^{d-1+\epsilon}$		$n > d: \ O(\binom{n}{d} d^d m^{2d-2})^{d-1+\epsilon}$
	time	$\Theta(m^{d-1})$	$n \leq d: \ O(d^d m^{d+n-2})^{2d-4+\epsilon}$		$n > d: \ O(\binom{n}{d} d^d m^{2d-2})^{2d-4+\epsilon}$

Table 1: Combinatorial complexity of uncertainty envelopes and run time complexity of our algorithm. Except for $d > 3$, the algorithm's storage requirement is identical to the combinatorial complexity. The function $\lambda_s(m)$ is the upper bound on the complexity of a Davenport-Schnitzel sequence of order s on m symbols (nearly linear).

marized in Table 1. The algorithm identifies all the topologies assumed by the zonotope as it sweeps along the convex combinations of the sites, and then it sweeps individual facets that participate in topological changes. This is significantly better than the straightforward approach of sweeping the convex hull of the sites along trajectories determined by the faces of the uncertainty domain hyperbox Δ, which has exponential complexity in the number of parameters m.

This abstract is organized as follows. Section 2 reviews zonotope topology and computation. In Section 3, we describe an arrangement of surfaces which encodes the topology of the zonotope over all convex combinations of the sites. In Section 4 we use the arrangement to compute a small subset of the faces of hyperbox Δ which contribute to the uncertainty envelope. The general algorithm is described in Section 5. Section 6 described future work and open problems. Throughout this abstract we assume that the input is in general position, that is the the sensitivity matrix columns of the sites are pairwise linearly independent.

2 Zonotope topology

We now describe uncertainty envelopes for one site ($n = 1$). As the j^{th} parameter spans the interval $[-\delta_j, \delta_j]$, the site moves along a translate of the line segment $conv\{-a_j\delta_j, a_j\delta_j\}$, where a_j is the j^{th} column of the sensitivity matrix A. Since the parameters affect the site independently, the envelope is the boundary of the Minkowsky sum of the m line segments, translated by the nominal value b. This is the definition of a *zonotope*, a convex centrally symmetric polytope [1]. We review relevant zonotope properties below.

There is a geometric correspondence between zonotopes in $\mathbb{R}^d$ and hyperplane arrangements in $\mathbb{R}^{d-1}$. The correspondence is realized by the following transform: let $H_j = \{x | \langle x, a_j \rangle = 0\}$, the hyperplane normal to a_j and passing through the origin, and let $H_0 = \{x | x_d = 1\}$. Transform each column vector $a_j = (a_{j1}, a_{j2}, \ldots, a_{jd})^{\perp}$ to the $(d-2)$-dimensional hyperplane $h_j = H_j \cap H_0$ whose equation is:

$$a_{j1}x_1 + a_{j2}x_2 + \ldots + a_{j(d-1)}x_{d-1} + a_{jd} = 0 \qquad (1)$$

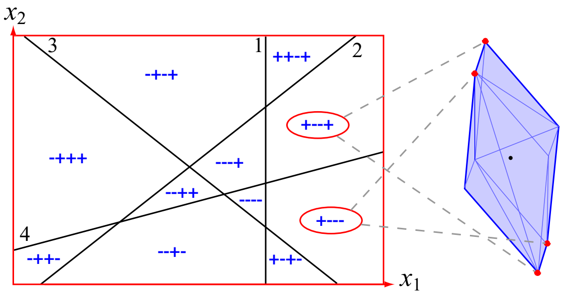


Figure 1: The 3D zonotope (right) and its topology arrangement (left) corresponding to the sensitivity matrix $A = \begin{pmatrix} 1 & -2 & 1 & -1 \\ 0 & 2 & 1 & 3 \\ -1 & -1 & 2 & 5 \end{pmatrix}$. The j^{th} line equation is determined by substituting a_j into Eqn. 1. The marked neighboring 2-cells correspond to the neighboring zonotope vertices.

Eqn. 1 transforms the generators of the zonotope, $a_j \in \mathbb{R}^d$, to a hyperplane arrangement in $\mathbb{R}^{d-1}$ which has the same topology as the zonotope in the following sense. The sign vector $\sigma = (\sigma_1, \ldots, \sigma_m) \in \{-1, 0, 1\}^m$ of an arrangement cell containing a point x is determined according to the signs of $\langle x, a_j \rangle$ for $1 \leq j \leq m$. Each k-cell of the arrangement corresponds to two symmetric (antipodal) $(d-k-1)$-cells of the zonotope. Specifically, the $(d-1)$-cells of the arrangement with sign vector σ correspond to the vertices $v^+ = b + \sum_{j=1}^{m} a_j\delta_j\sigma_j$ and $v^- = b - \sum_{j=1}^{m} a_j\delta_j\sigma_j$, which achieve the maximum of $\langle x, v \rangle$ over x (equivalent to a direction) in the cell and v in the zonotope. The sign vectors of neighboring $(d-1)$-cells differ in one entry only, and they correspond to neighboring vertices on the zonotope. Thus it is possible to compute the vertex representation of the zonotope in optimal $\Theta(m^{d-1})$ time. Figure 1 shows an example in 3D.

3 Swept zonotope topology

We now describe the topology of the zonotope as it is swept between the n sites. For clarity of explanation, we focus on the case $n = d$. In Section 5 we show how to solve for the general case.

When $n = d$, the uncertainty envelope bounds the

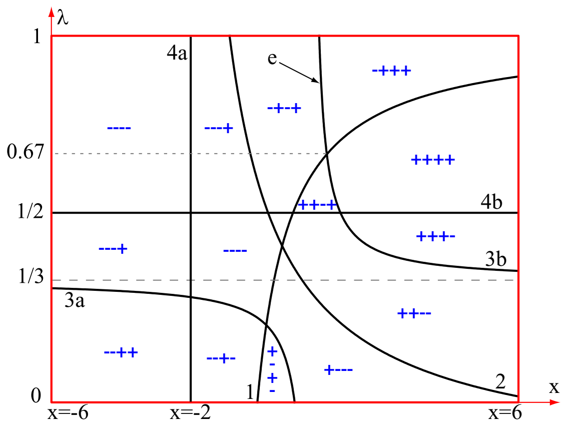


Figure 2: Dynamic topology arrangement of $A_1 = \begin{pmatrix} 0.7 & 0.1 & -0.4 & -1 \\ 0.5 & -0.7 & 0.1 & -2 \end{pmatrix}$, $A_2 = \begin{pmatrix} 0 & 0.7 & 0.8 & 1 \\ -0.6 & 1 & -0.7 & 2 \end{pmatrix}$. Each of the curves is obtained by substituting the corresponding matrix columns into Eqn. 2. The curves corresponding to columns 3 and 4 demonstrate degeneracies which occur when a subset of the coefficients of x in Eqn. 2 are zero for some value of $\lambda \in \Lambda$.

union of $(d-1)$-simplices which are the convex hulls of sites whose positions vary within their respective zonotopes. Consider a convex combination of the sites defined by the coefficients $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n)$. The domain of the coefficients is $\Lambda = [0,1]^n \cap \{(\lambda \in \mathbb{R}^n | \sum_{i=1}^n \lambda_i = 1\}$. The position of a combination of sites is given by $w(p,\lambda) = \sum_{i=1}^n \lambda_i v_i(p) = \sum_{i=1}^n (\lambda_i A_i p + \lambda_i b_i)$. Let the combined sensitivity matrix be $\tilde{A}(\lambda) = \sum_{i=1}^n \lambda_i A_i$. Then the topology of the combined sites zonotope, defined by $\tilde{A}(\lambda)$ and denoted $Z(\lambda)$, changes as λ varies. Replacing a_{ji} in Eqn. 1 with the entries of $\tilde{A}(\lambda)$, we obtain:

$$\sum_{i=1}^{n} \lambda_i a^i_{j1} x_1 + \ldots \sum_{i=1}^{n} \lambda_i a^i_{j(d-1)} x_{d-1} + \sum_{i=1}^{n} \lambda_i a^i_{jd} = 0 \quad (2)$$

where a^i_{jk} denotes the k^{th} entry in the j^{th} column of matrix A_i. Since $\lambda_n = 1 - \sum_{i=1}^{n-1} \lambda_i$, this is an algebraic surface of degree two in the variables $(x_1, x_2, \ldots, x_{d-1}, \lambda_1, \lambda_2, \ldots, \lambda_{n-1})$. We denote the dimension of the embedding space by $D = 2(d-1)$.

The arrangement of the m surfaces is called the *dynamic topology arrangement* (DTA), because it encodes the topology of the zonotope of the combined site as its center moves along $conv\{b_i\}$. Every D-cell of the DTA represents values of x for which the sign vector is the same, defining two antipodal vertices of the zonotope $Z(\lambda)$ for λ within the cell. The intersection of two D-cells corresponds to two antipodal edges of the zonotope. With the general position assumption, an intersection of k D-cells which is also the intersection of exactly $k-1$ surfaces corresponds to two antipodal $(k-1)$-cells of the zonotope. The complexity of the entire DTA is $\Theta(m^D)$. Figure 2 shows a two dimensional DTA.

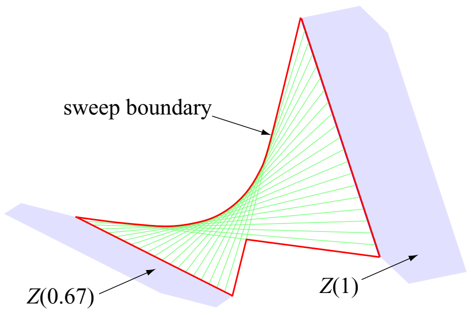


Figure 3: Sweeping the line segment corresponding to the edge e in Figure 2 through the subspace $P_C = (-\delta_1, \delta_2, 0, \delta_4) + conv\{-e_3, e_3\}$. Note that only a subset of Λ needs to be swept (from 0.67 to 1). The solid polygons are the zonotopes at $\lambda = 0.67$ and $\lambda = 1$. The segment is shown at 20 intermediate values of p_3. The closed curve bounds the swept area.

4 The sweep subspace

The uncertainty envelope is the boundary of the union of all the instances of the simplex determined by $p \in \Delta$. Although it is sufficient to include only instances with $p \in \partial\Delta$, the boundary has an exponential number of cells m.

We now show how the DTA reduces this number to a polynomial in m. When λ traces a path in its domain Λ, the zonotope undergoes a general sweep in Euclidean space. Weld and Leu [5] show that the volume swept by a compact d-manifold in $\mathbb{R}^d$ undergoing a general sweep is equal to the union of the volumes swept by its boundary and one location of the compact d-manifold in the sweep. The boundary facets of the zonotope generally correspond to the intersection of d D-cells of the DTA, i.e. $(d-1)$-cells of the arrangement. Points incident on the facet differ only in the value of $d-1$ out of m parameters in the uncertainty vector p. The indices of these parameters are exactly those with zero sign in the $(d-1)$-cell of the DTA corresponding to the facet.

Let F be a facet of the zonotope and let C be the DTA cell it corresponds to. Let $\sigma_C = (\sigma_1, \ldots, \sigma_m)$ be the sign vector of C, and let I_C be the set of indices with zero sign in σ_C. We define the sweep subspace of the cell as $P_C = (\sigma_1\delta_1, \ldots \sigma_m\delta_m) + \sum_{i \in I_C} conv\{-e_i\delta_i, e_i\delta_i\}$, where $\{e_i\}$ are the standard basis vectors, and the plus and summation signs are Minkowski additions of sets. Since a point in F is attained by some value of $\lambda \in \Lambda$ and $p \in P_C$, the sweeping of the facet through all values of Λ is equivalent to sweeping the simplex through all the values of P_C. Therefore, to obtain the uncertainty envelope, it suffices to sweep the simplex through the values of P_C defined by all the $(d-1)$-cells of the DTA. For the computation and approximation of swept volumes see e.g. [3, 5]. Figure 3 shows an example in 2D.

```
1.  Construct the dynamic topology arrangement $\mathcal{A}$.
2.  Compute sign vector of an arbitrary $(d-1)$-cell $C_{init} \in \mathcal{A}$.
3.  Traverse the $(d-1)$-cells of $\mathcal{A}$ in BFS order, starting from $C_{init}$:
    a. Update the sign vector of current cell $C$: $\sigma_C = (\sigma_1, \ldots, \sigma_m)$.
    b. Find indices $I_C$ of zero signs in $\sigma_C$, defining sweep subspace $P_C$.
    c. Initialize swept volume $\mathcal{V}$ with simplex corresponding to arbitrary vertex $p \in P_C$.
    d. For each $i \in I_C$:
       - Sweep $\mathcal{V}$ from current $p = (p_1, \ldots, p_i, \ldots, p_m)$ to $p_{-i} = (p_1, \ldots, -p_i, \ldots, p_m)$.
       - Set $\mathcal{V}$ to new swept volume, set $p = p_{-i}$.
    e. Insert the boundary of $\mathcal{V}$ into arrangement of surfaces $\mathcal{H}$.
4.  Compute uncertainty envelope as inner boundary of the outer $d$-cell of $\mathcal{H}$.
```

Table 2: Algorithm for computing uncertainty envelopes

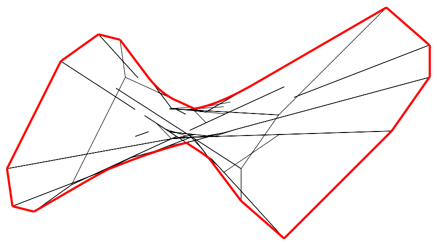


Figure 4: The arrangement $\mathcal{H}$ of swept boundaries of the input of Figure 2 with $b_1 = (0,0)$, $b_2 = (1,0)$, $\delta_1 = \delta_2 = 0.1$. The uncertainty envelope is the thick closed curve bounding $\mathcal{H}$. Notice that many of the curves do not contribute to the outer cell of the arrangement.

5 Algorithm

Table 2 presents the algorithm for computing the uncertainty envelope, based on the properties described in the previous sections. The algorithm starts by computing the DTA defined by Eqn. 2. It then traverses the DTA, iterating on the $(d-1)$-cells. For each cell C, it computes its sign vector, which defines the sweep subspace P_C. It then computes the volume swept by the simplex when p spans P_C. The collection of surfaces which bound the swept volumes of all the simplices form an arrangement $\mathcal{H}$, whose outer cell defines the uncertainty envelope.

The complexity of the algorithm, shown in Table 1, depends predominantly on the combinatorial and computational complexity of arrangements and their substructures. For $d > 3$, the current best algorithm for arrangement traversal and single cell computation is not optimal. For the most recent survey of arrangements, see [2].

Planar envelopes: The curves bounding the swept areas of line segments consist of line segments and sections of parabolas (Figure 3). Figure 4 shows an example.

3D envelopes: The swept volume of a line segment in space is a hyperbolic paraboloid. The envelope of a swept plane in space is a ruled and developable surface [5]. Thus, the sweep boundary of a triangle consists of both types of surfaces, which are swept again in the second iteration of step 3d to produce surfaces of $\mathcal{H}$.

Cases of $n \neq d$: When $n < d$ the DTA has dimension $d+n-2$, and the algorithm sweeps $(n-1)$-simplices. For $n > d$, the uncertainty envelope is the boundary of the volume swept by the convex hull of the n sites when p spans Δ. Since the facets of the convex hull change during the sweep, we must consider all $\binom{n}{d}$ possible facets, and apply steps $1-3$ of the algorithm for each facet before applying step 4.

6 Conclusion

While the number of swept volumes depends on the complexity of the DTA, not all the N sweep boundaries inserted in step 3e contribute to the outer cell. Constructing an example in which $O(N)$ surfaces participate in the envelope, or proving a better bound, is an open problem. Furthermore, the properties of the outer cell of $\mathcal{H}$ suggest that the worst case bound on its complexity may be lower.

For efficiency of computation, the swept volumes in 3D can be approximated by polyhedrons [3]. Finally, the use of spatial data structures may reduce the number of facets considered when $n > d$.

References

[1] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*, Springer-Verlag, Heidelberg, 1987.

[2] D. Halperin. Arrangements. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 24, pages 529–562. CRC Press LLC, Boca Raton, FL, 2004.

[3] Y. J. Kim, G. Varadhan, M. C. Lin, and D. Manocha. Fast swept volume approximation of complex polyhedral models. In *Symposium on Shape modeling and Applications*, 2003.

[4] Y. Ostrovsky-Berman and L. Joskowicz. Tolerance envelopes of planar mechanical parts. In *9th ACM Symposium on Solid Modeling and Applications*, pages 135–143, Genova, Italy, 2004.

[5] J. D. Weld and M. C. Leu. Geometric representation of swept volumes with application to polyhedral objects. *International Journal of Robotics Research*, 9(5):105–117, 1990.

Covering Point Sets with Two Convex Objects

J. Miguel Díaz-Báñez[*] Carlos Seara[†] J. Antoni Sellarès[‡] Jorge Urrutia[§] Inma Ventura[¶]

Abstract

Let P_{2n} be a point set in the plane with n red and n blue points. Let C_R and C_B (S_R and S_B) respectively be red and blue colored and disjoint disks (axis-parallel squares). In this paper we prove the following results. Finding the positions for C_R and C_B that maximizes the number of red points covered by C_R plus the number of blue points covered by C_B can be done in $O(n^3 \log n)$ time. Finding two axis-parallel unit-squares with disjoint interiors that maximizes the sum of the red points covered by S_R plus the number of blue points covered by S_B can be done in $O(n^2)$ time.

1 Introduction

Consider a point set P_{2n} with n *red* and n *blue* points, and a red and a blue coin (not necessarily of the same size) denoted by C_B and C_R respectively. In this paper we study the following problem. Place C_R and C_B on the plane such that the number of red points covered by C_R plus the number of blue points covered by C_B is maximized. We allow C_B and C_R to cover some red (resp. blue) points, but require them to have disjoint interiors. We also consider the problem of finding two isothetic squares S_B and S_R of given sizes with disjoint interiors such that the number of red points covered by S_R plus the number of blue points covered by S_B is maximized. We consider also a similar problem of finding two isothetic squares S_B and S_R of given sizes with disjoint interiors such that the number of red points covered by S_R plus the number of blue points covered by S_B is maximized. In what follows, and to avoid repetitions, a bi-colored point set is a point set such that all its elements are colored red or blue.

In [10] the following problem is considered: given a bi-colored point set P_n on the plane, find an axis-parallel box that does not contain blue points and maximizes the number of red points it covers. An $O(n^2 \log n)$ time algorithm is presented for solving this problem.

In [1] a similar problem is studied: given a bi-colored point set P_n in $\mathbb{R}^d$, find a ball that maximizes the number of red points it contains without containing any blue point in its interior. For $d = 2$, this problem is solved in $O(n^2 \log n)$ time. Monochromatic variants of these problems were studied in [5, 8].

In Pattern Recognition and Classification problems, a natural method to select prototypes to represent a class is to performer cluster analysis on the training data [7]. The clustering can be obtained by using simple geometric shapes such as circles or boxes. Recent papers deal with the *maximum bi-chromatic covering problem*. The problem is the following: given a bi-chromatic point set, the goal is to maximize the number of points of a given color, say red, covered by a given object while avoiding points of the second color. In [1] and [13] circles and boxes respectively are considered for the classification.

In some cases, requiring that the red (resp. blue) covering ball avoid blue (resp. red) points may is some cases lead to invalid classifications and make the results obtained useless. This scenario can arise when facilities interfere with each other, but their possible users are scattered randomly on the plane. A possible solution to this problem is to allow blue points in a red ball and red points in a blue ball. In this paper we introduce this criterion with fixed sizes for the circles or boxes.

A central problem in facility location is to find the best location for a facility to serve a set of users. Maximal covering disk problems are often the main criteria used in facility location where in a natural way a point is served by a facility if it is within a given *distance* from it. In some cases the metric used is the Euclidean distance, and in others the l_∞ (box) metric. Many of these problems also arise in operations research [11]. The problem for locating a maximum covering circle of a given size in a monochromatic set was studied in [6].

[*]Dpto. Matemática Aplicada II, Universidad de Sevilla, Spain, dbanez@us.es. Partially supported by grant BFM2003-04062.

[†]Dpt. Matemàtica Aplicada II, Universitat Politècnica de Catalunya, Spain, carlos.seara@upc.edu. Partially supported by projects DGES-SEUID-PB98-0933, MCYT-FEDER-BFM2002-0557 and Gen-Cat-2001SGR00224.

[‡]Institut d'Informàtica i Aplicacions, Universitat de Girona, Spain, sellares@ima.udg.es. Partially supported by project TIN2004-08065-C02-02.

[§]Instituto de Matemáticas, Universidad Nacional Autónoma de México, urrutia@matem.unam.mx. Supported by CONACYT of México, Proyecto 37540-A.

[¶]Dpto. de Matemáticas, Universidad de Huelva, Spain, iventura@us.es. Partially supported by grant BFM2003-04062.

2 Two disjoint disks

Let P_{2n} be a planar bi-colored point set with n blue and n red points. In this section we study the problem of placing two disks, one red disk C_R and one blue disk C_B in such a way that their interiors are disjoint, and the number of red points covered by the red disk plus the number of blue points covered by the blue disk is maximized. In this section we prove:

Theorem 1 *Finding the positions for C_R and C_B such that the number of red points covered by C_R plus the number of blue points covered by C_B is maximized can be done in $O(n^3 \log n)$ time.*

To facilitate our presentation we will assume that the disks are the same size, i.e., equal radii, say r. Assume first that C_R and C_B are placed on the plane such that C_R (resp. C_B) covers a subset R_1 (resp. B_1) of red (resp. blue) points of P_{2n}.

In what follows, when we say that a disk C_R or C_B has some points on its boundary, we shall assume that those points have the same color as the disk. With this in mind, we state the following result (see Figure 1):

Lemma 2 *The disks C_R and C_B can be moved to a new position in the plane such that either: i) one of the disks has at least two points on its boundary and the other one has at least one point on its boundary, or ii) each disk has only one point on its boundary and these points are collinear with the centers of the disks.*

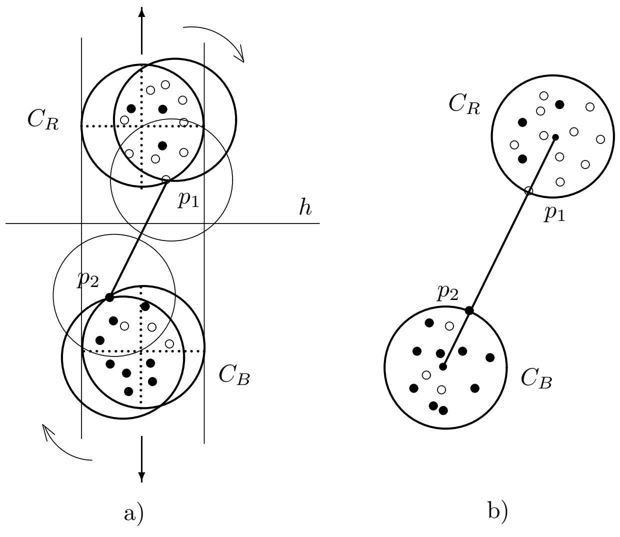


Figure 1: Points in the boundaries of the disks.

Constructing an arrangement of red and blue circles

Let P_n be a bi-colored point set on the plane. For each red point r_i (resp. blue point b_j) we consider the red (resp. blue) circle with center in r_i (resp. b_j) and radius r. Let $\mathcal{A}$ be the arrangement of the $2n$ red and blue circles. As the circles are closed Jordan curves such that any of them intersect in at most two points, the following results can be applied to the arrangement $\mathcal{A}$.

Theorem 3 [12, 4, 5] *The following statements hold for $\mathcal{A}$: 1) The combinatorial complexity of a single face in $\mathcal{A}$ is at most $\lambda_2(n) = O(n)$. 2) The combinatorial complexity of the zone of a circle in $\mathcal{A}$ is $O(n\alpha(n))$. 3) $\mathcal{A}$ can be computed in $O(n^2)$ time or using the sweep-line algorithm of Bentley and Ottmann [3] in $O(n^2 \log n)$ time.*

Notice that a circle can contribute more than one arc to a cell of $\mathcal{A}$. Associate in $\mathcal{A}$ the pair of numbers (nr_j, nb_j) to each cell j such that nr_j (resp. nb_j) is the number of red (resp. blue) *convex arcs* which belong to the boundary of j. Clearly a circle of radius r with center at any point x in a cell j covers exactly nr_j red points and nb_j blue points (Figure 2). Using a line-sweep algorithm, we can compute the pair (nr_j, nb_j) associated to each cell j in $O(n^2 \log n)$ time.

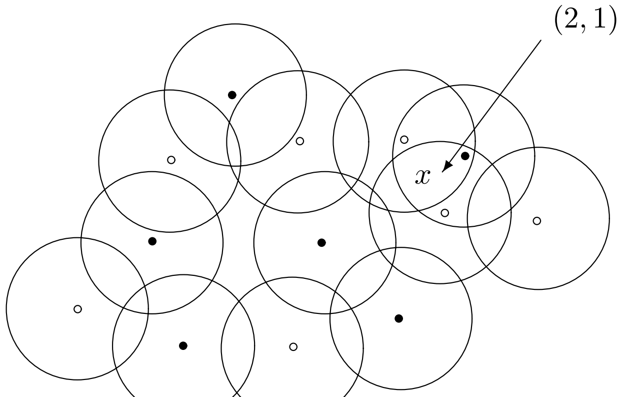


Figure 2: Arrangement $\mathcal{A}$.

Pre-processing a circle-arrangement

For each point $p_k \in P_{2n}$, the locus a_k of centers of circles passing through p_k with radius r defines a circle with center at p_k and radius r.

By Theorem 3, each a_k is split into at most $O(n)$ arcs, each associated to a cell j of $\mathcal{A}$. We label each of these sub-arcs with the pair (nr_j, nb_j) of its associated cell. We store this information for a_k in a *segment tree* [2] so that we can obtain for any sub-arc SC of a_k the sub-arc of $\mathcal{A}$ contained in a_k that intersects SC and maximizes the number nr_i or nb_j, which is the maximum number of red or blue points covered by a circle of radius r with center in a_k, in $O(\log n)$ time. For each circle a_k, the segment tree uses $O(n \log n)$ space and can be built in $O(n \log n)$

time. Doing this at a pre-process stage and storing the information for all the circles of $\mathcal{A}$ takes $O(n^2 \log n)$ time and $O(n^2 \log n)$ space.

The algorithm

We now show how to compute the centers of the two disks C_B and C_R such that the sum of the number of red points and blue points covered respectively by C_B and C_R is maximized.

TWO-DISKS-EQUAL-RADII-ALGORITHM

1. Compute $\mathcal{A}$. For each cell j of $\mathcal{A}$ compute the pair (nr_j, nb_j) and the segment trees for the circles a_k above.
2. By Lemma 2 we can find C_B and C_R as follows:
   (a) For any red point p_i and any blue point p_j compute the circles $C_{R(i)}$ and $C_{B(j)}$ (having p_i and p_j respectively on their boundaries) of radius r with centers on the line joining p_i to p_j, and at distance $|p_i - p_j| + 2r$. In $O(n)$ time, compute the number or red and blue points contained in $C_{R(i)}$ and $C_{B(j)}$ respectively. Keep the pair that maximizes the number of red points in C_{R_i} plus the number of blue points in $C_{B(j)}$.
   (b) For any circle $C_{R(i,j)}$ of radius r and center $c(i,j)$ that passes through a pair of red points p_i, p_j in P_{2n}, do the following: for any blue point p_k at distance greater than or equal to r from $c(i,j)$, compute in constant time, the sub-arc a'_k of the circle a_k consisting of all the points of a_k at distance greater than or equal to r from $c(i,j)$. Using the information stored in the *segment tree* of a_k in $O(\log n)$ time, get the sub-arc of a_k in $\mathcal{A}$ that intersects a'_k with the largest nb_i, which corresponds to a disk C_B with radius r and center on a'_k that maximizes the number of blue points it contains. Compute the numbers of red and blue points covered by $C_{R(i,j)}$ and C_B respectively, keeping the best solution, see Figure 3. Do the same for circles of radius r that pass through two blue points, and keep the best overall solution.

Clearly the best of the solutions obtained in 2(a) and 2(b) solves our problem.

Analysis of the algorithm. Step 1 of the algorithm can be done in $O(n^2 \log n)$ time, step 2(a) can be done in $O(n^3)$ time. Finally, step 2(b) takes $O(n \log n)$ time per pair of points, i.e., $O(n^3 \log n)$ total time, assuming a pre-process of $O(n^2 \log n)$ time for computing the *segment trees*. Thus the algorithm runs

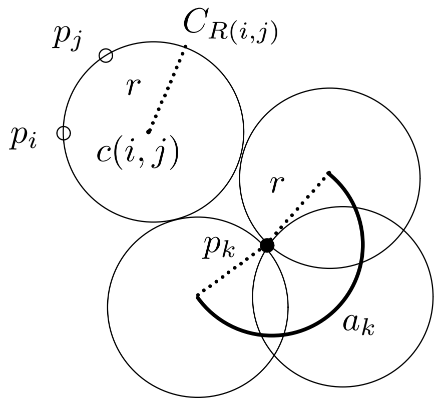


Figure 3: Locus a_k of centers of circles C_2.

in $O(n^3 \log n)$ time. It is easy to see that with some slight modifications we can drop the condition that the two circles have the same size. This concludes the proof of Theorem 1.

3 Two disjoint axis-parallel squares

In this section we study a similar problem in which instead of finding two circles C_R and C_B, we now want two isothetic squares S_B and S_R of given sizes with disjoint interiors such that the number of red points covered by S_R plus the number of blue points covered by S_B is maximized. As in the previous section, we restrict ourselves to squares of the same size, i.e. unit squares. This condition can be lifted easily, leaving the results unchanged. It is easy to see that a similar result is valid for two disjoint quadrilaterals with pairwise parallel sides with given directions. Assume that the red points of P_{2n} are labeled $r_1, \ldots, r_n$ such that for $i < j$ the x-coordinate x_i of r_i is smaller than the x-coordinate of r_j. Assume a similar labeling for the blue points $b_1, \ldots, b_n$ of P_{2n}.

3.1 Two disjoint axis-parallel unit squares

Let S_R and S_B be the elements of an optimal solution. Observe first that since S_R and S_B are isothetic, there is a horizontal or vertical line l that separates them. Assume that l is vertical, and that S_R is to the left and S_B to the right of l. Slide S_R to the left until its right side meets a red point. Observe that since S_R is in an optimal solution, the number of red points it covers does not change. Thus we can assume that S_R contains a red point on its right side. Similarly we can assume that S_B contains a blue point on its left edge. Using these observations, we now outline a process to find an optimal pair S_R and S_B in $O(n^2)$ time. First order the red and the blue points of P_{2n} according to its y-coordinate.

For each red point $r_i \in P_{2n}$, find the unit square containing r_i on its right side which contains the maximum number of red points in P_{2n}. This can be done in linear time as follows.

First find the set R_i of red points to the left of r_i within the vertical strip ST of unit width bounded to its right by l. Find the unit square S contained in ST such that r_i is on its top side, and count the number of red points it contains. Now slide S from bottom to top (keeping it within ST) until it reaches a position in which r_i lies on its bottom edge. While sliding S, keep track of the number of red points it contains as they enter or leave S. This can be done easily in linear time using the order of the red points of P_{2n} according to its y-coordinate. At the end of this process, we have identified the square containing r_i on its right side that contains the maximum number m_{r_i} of red points.

In a similar way we can process the blue points of P_{2n} such that for each blue point b_j we find a square containing b_j on its left edge, and containing the maximum number m_{b_j} of blue points.

For each r_i let $mr(i) = \max\{m_{r_k}; k \leq i\}$. The set $\{mr(i) : i = 1, \ldots, n\}$ can be found in linear time with a single scan of the set of red points from left to right. In a similar way, we can find the set of values $\{mb(j) : j = 1, \ldots, n\}$ such that $mb(j) = \max\{m_{b_k} : k \geq j\}$.

Using the lists $mr(1), \ldots, mr(n)$, $mb(1), \ldots, mb(n)$ and traversing all the points (blue and red) of P_{2n} from left to right we can find an optimal pair of squares S_R and S_B in linear time. For brevity, the details are left to the reader. Full details appear in a longer version of this paper. Proceeding in a similar way when S_B is to the left of l and S_R to its right, or when a horizontal line separates S_R and S_B we obtain the following theorem.

Theorem 4 *Finding two axis-parallel unit-squares with disjoint interiors such that the sum of the red points covered by S_R plus the number of blue points covered by S_B is maximized can be done in $O(n^2)$ time.*

Theorem 5 *The two axis-parallel unit-squares with disjoint interiors such that the sum of the the red points covered by S_R plus the number of blue points covered by S_B is maximized requires $\Omega(n \log n)$ time under the algebraic computation tree model.*

The lower bound can be proved by a reduction to the uniform gap problem [9].

To close we would like to mention that using similar techniques as those in this section, we can obtain the following result:

Theorem 6 *Let P_n be a set of n points in the plane. The problem of finding three axis-parallel rectangles (not necessarily of the same size) with disjoint interiors such that the number of points of P_n covered by them is maximized can be solved in $O(n^3)$ time.*

Acknowledgments

These problems were posed and partially solved during the *Second Spanish Workshop on Geometric Optimization*, July 5–10, 2004, El Rocio, Huelva, Spain.

References

[1] B. Aronov, S. Har-Peled. On approximating the depth and related problems. *Proceedings 16th Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA 2005), 2005.

[2] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf. *Computational Geometry, Algorithms and Applications.* Springer, 1997.

[3] J. L. Bentley, T. A. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Trans. Comput.* C-28, 1979, pages 643–647.

[4] A. Calatayud. Problemas geométricos de localización. *PhD Thesis, Universidad Politécnica de Madrid*, 2004.

[5] B. Chazelle, D. T. Lee. On a circle placement problem. *Computing*, Vol. 36, 1986, pages 1–16.

[6] Z. Drezner. On a modified one-center model. *Management Science*, 27, 1981, pages 848–851.

[7] R. Duda, P. Hart, D. Stork. *Pattern Classification. John Wiley and Sons, Inc.*, New York, 2001.

[8] M. J. Katz, K. Kedem, M. Segal. Improved algorithms for placing undesirable facilities. *Computers & Operations Research*, 29, 2002, pages 1859–1872.

[9] D. T. Lee, Y. F. Wu. Geometric complexity of some location problems. *Algorithmica*, 1, 1986, pages 193–211.

[10] Y. Liu, M. Nadiak. Planar case of the maximum box and related problems. In *Proceeding of the Canadian Conference on Computational Geometry*, CCCG'03, Halifax, Nova Scotia, 2003, pages 11–13.

[11] R. F. Love, J. G. Morris, G. O. Wesolowsky. *Facilities Location. North-Holland*, Amsterdam, 1988, chapter 3.3, pages 51–6.

[12] M. Sharir, P. K. Agarwal. *Davenport-Schinzel sequences and their geometric applications. Cambridge University Press*, 1995.

[13] M. Segal. Planar maximum box problem. *Journal of Mathematical Modelling and Algorithms*, 3, 2004, pages 31–38.

Maximum Line-Pair Stabbing Problem and its Variations[*]

Sebastian Böcker Veli Mäkinen[†]

Abstract

We study the *Maximum Line-Pair Stabbing Problem*: Given a planar point set S, find a pair of parallel lines within distance ϵ from each others such that the number of points of S that intersect (stab) the area in between the two lines is maximized. There exists an algorithm that computes maximum stabbing in $O(|S|^2)$ time and space. We give a more space-efficient solution; the time complexity increases to $O(|S|^2 \log |S|)$, but the space reduces to $O(|S|)$. Our algorithm also extends to a dual problem where one searches for a line stabbing maximum number of variable size circles; as far as we know, this problem has previously been studied only on fixed size circles.

A variant of the stabbing problem equals a one-dimensional point set matching problem under translations, scalings, and errors. We study a version of this problem, where the matching has to be a one-to-one mapping. Existing techniques based on incremental maintenance of maximum matching using augmenting paths yield $O((mn)^3)$ time solution, where m and n are the sizes of the point sets to be matched. Our new algorithm achieves $O((mn)^2(m+n))$ time. The improvement is based on an observation that in our case the match-graph has a regular shape, and the maximum matching can be updated more efficiently.

1 Introduction

There are three dual ways to describe the *Maximum Line-Pair Stabbing Problem* (see Fig. 1):

- Given a set of points, find a pair of parallel lines within distance ϵ from each others such that the number of points in between the lines (including them) is maximum. For short, the resulting line-pair is said to stab maximum number of points.
- Given a set of circles with diameter ϵ, find a line that goes through maximum number of them.
- Given a set of lines, find a vertical line segment starting at point (x, y) of length $\delta(x)$ that crosses maximum number of lines. (Function δ will be defined later.)

[*]Technische Fakultät, Universität Bielefeld, Germany, {boecker,veli}@cebitec.uni-bielefeld.de. Supported by the ‘‘Deutsche Forschungsgemeinschaft’’ (BO 1910/1-3) within the Computer Science Action Program.

[†]Also partially supported by the Academy of Finland.

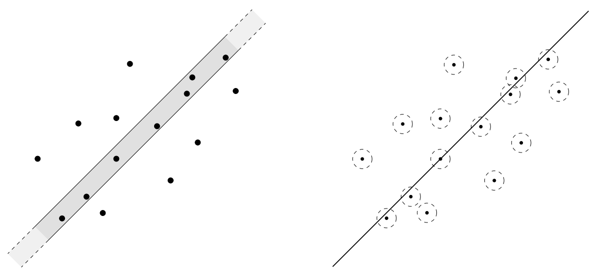


Figure 1: Stabbing points with a line-pair (left) and stabbing circles with a line (right).

Our motivation to study the problem comes from the calibration of Mass Spectrometry data [4]. There we basically have two sets of real values, one corresponding to the theoretical reference spectrum, and one for measured spectrum. We know that there is a linear function mapping the measured values close to the reference values. One way to find such transformation is to map each pair (reference value, measured value) into a plane, and find a line-pair stabbing maximum number of points.

The maximum stabbing problem has been studied earlier by Chin, Wang, and Wang [6]. They gave a quadratic time and space algorithm that is based on the third dual interpretation given above.

We use the same dual mapping in our algorithm as [6], but in a different way. The consequence is that we are able to manage with linear space, but the time complexity increases by a logarithm factor.

There is also an indirect way to solve the problem using geometric range query data structures. This solution is conceptually easiest to understand, but it also uses nearly quadratic space.

Returning back to the motivation [4] on two sets of real values, we note that the maximum stabbing problem is an over-generalization of that problem; instead of restricting by ϵ the Euclidean distance between the two lines, one can restrict the distance in the one-dimensional projection. As such the problem is not well-defined as it has a degenerate optimum solution. We study a robust variant of this problem, where we require that the linear function mapping one set to the other maximizes the number of one-to-one matches. This problem can be solved using existing techniques for incremental bipartite matching. We give a new, one order of magnitude faster, solution to the problem.

2 Maximum Stabbing using Half-Space Range Counting Queries

Let $S \subset \mathbb{R}^2$ be a point set and $\epsilon > 0$ a given parameter defining an instance of the maximum line-pair stabbing problem throughout this paper. In the following, we derive an $O(|S|^2 \log |S|)$ time solution to the problem using data structures for half-space range searching [1].

Let us denote by $\mathcal{L}$ the set of feasible line-pairs, i.e. those pairs of parallel lines within distance ϵ from each others. The solution is as follows:

(i) Build the data structure of Chazelle [5] as pointed out in [1, Theorem 4.4, p. 24] for half-space range counting queries on S. The data structure can be constructed in $O(|S|^2)$ time and it occupies $O(|S|^2/\log^2|S|)$ space. For any line, it gives the number of points on either side of it in $O(\log|S|)$ time.

(ii) Construct the set $L \subset \mathcal{L}$ of *representative* line-pairs as follows. On each pair of points of S, pair the line going through these points with the two parallel lines at ϵ distance from it. Add these two line-pairs to L. Also, on each pair (p, s) of points of S having distance greater or equal to ϵ, add the (two) line-pair(s) to L formed by the parallel lines at ϵ distance from each others such that one line goes through p and one through s.

(iii) Make two queries on the data structure on each line-pair in L, to compute how many points of S intersect the area in between the lines.

(iv) Choose the line-pair that obtains the maximum count.

The correctness of the method follows using a shifting argument to show that it is enough to consider the set of representative line-pairs.

The algorithm makes $O(|S|^2)$ queries each taking $O(\log|S|)$ time, giving overall running time $O(|S|^2\log|S|)$. The space usage is $O(|S|^2/\log^2|S|)$.

3 Direct Solution to Maximum Stabbing Problem

We solve the problem using the third dual interpretation mentioned in the introduction. We map each point $p = (p_x, p_y) \in S$ into a line $p^* : y = p_x x - p_y$. A line $\ell : y = mx + b$ is mapped into a point $\ell^* = (m, -b)$. Consider another line $\kappa : y = mx + (b + \delta(m))$ parallel to ℓ, where $\delta(m)$ is defined so that the distance of κ and ℓ is ϵ. That is, $\delta(m) = \epsilon\sqrt{1^2 + m^2}$. This line is mapped into a point $\kappa^* = (m, -b - \delta(m))$. One easily notices, that the parallel lines in between κ and ℓ, are mapped into a line segment $m \times [-b - \delta(m), -b]$. Hence, our original problem equals that of finding a line segment $m \times [-b - \delta(m), -b]$ such that the number of lines it intersects is maximal, over all choices of m and $-b$. We are ignoring vertical line-pairs for the moment.

Let us now derive the algorithm that finds the optimum values for m and $-b$ in time $O(|S|^2\log|S|)$ and space $O(|S|)$. Using again a shifting argument we notice that $(m, -b)$ can be chosen so that $-b = p_x m - p_y$ for some $(p_x, p_y) \in S$. Still our space of choices for m and $-b$ is infinite. To make it finite, we notice that fixing two lines partitions the range of m along one line into constant number of relevant ranges.

To be more precise, let us denote by S^* the set of lines that are images of points in S. Consider two lines $p^* : p_x x - p_y$ and $q^* : q_x x - q_y$ of S^*, and the case where $(m, -b)$ is chosen so that $-b = p_x m - p_y$. Equation

$$p_x m - p_y - \delta(m) = q_x m - q_y \tag{1}$$

has only constant number of solutions (if any), which means that there are only constant number of ranges $R \subseteq \mathbb{R}$ such that $m \in R$ if and only if point $(m, q_x m - q_y)$ is included in the line segment $m \times [p_x m - p_y - \delta(m), p_x m - p_y]$. Furthermore, the solutions to (1) can be found in constant time. Now, let $\mathbf{R}(p^*)$ be the multiset of ranges of line p^* formed by repeating the above process for each $q^* \in S^*$. After sorting the endpoints (y-coordinates) of the ranges in $\mathbf{R}(p^*)$ into ascending order, attaching to each endpoint value $+1$ or -1 depending on whether the endpoint is a start or end of a range, one can scan through the endpoints keeping a counter how many ranges are active at each phase. Endpoints with the same coordinates are sorted so that those associated with $+1$ precede those associated with -1. The endpoint associated with the largest count gives the optimal choice for $(m, -b)$ restricted to values such that $-b = p_x m - p_y$. The same process can be repeated constructing $\mathbf{R}(p^*)$, sorting it, and computing the largest count, for each $p^* \in S$. The optimum choice for $(m, -b)$ corresponds then to the overall largest count.

We have now found the optimum line segment $m \times [-b - \delta(m), -b]$ for the dual problem and hence we have found the line-pair stabbing the maximum number of points of S for the primal problem. The space requirement is that needed for storing set of ranges $\mathbf{R}(p^*)$ at each phase, i.e. $O(|S^*|) = O(|S|)$. The time requirement is that of sorting $\mathbf{R}(p^*)$ on each $p^* \in S^*$, i.e. $O(|S| \cdot |S|\log|S|) = O(|S|^2\log|S|)$. Finally, we note that if the solution is a vertical line-pair, our algorithm does not find it because the mapping to the dual plane does not apply for such lines. However, in this case, a simple vertical sweep is enough to compute the maximum stabbing over the linear number of relevant vertical line-pairs. This will only take $O(|S|\log|S|)$ time, which is negligible.

In the full version, we show how to handle variable size circles in the second dual interpretation, giving:

Theorem 1 *Given a set C of variable size circles on the plane, one can find the line going through maximum number of them in time $O(|C|^2 \log |C|)$ and space $O(|C|)$. The maximum line-pair stabbing problem is a special case, and can be solved within the same time and space bounds.*

4 One-Dimensional Point Set Matching under Translations, Scalings, and Errors

As explained in the introduction, when restricting to ϵ distances measured in the one-dimensional projection, the stabbing problem has a simpler interpretation as a point set matching problem: Given two sets of real values, i.e. one-dimensional point sets, $A, B \subset \mathbb{R}$, find a linear function $f : \mathbb{R} \to \mathbb{R}$ such that $|M_f(A, B, \epsilon)|$ is maximum, where M_f is a matching of A and B such that for each $(a, b) \in M_f$ holds $|f(a) - b| \leq \epsilon$. We call this variant of the stabbing problem the *linear one-dimensional point set matching* problem.

To solve the problem, consider a set F of representative linear functions constructed as follows: Let $B(\epsilon) = \{p - \epsilon, p + \epsilon \mid p \in B\}$. For each quadruple (a', a, b', b) such that $a', a \in A$ with $a' < a$ and $b', b \in B(\epsilon)$ with $b' < b$, add function $f(x) = \frac{b-b'}{a-a'}(x - a') + b'$ to F. Each function in F defines a translation and scaling that maps two points of A into ϵ distance from some points of B. Conditions $a' < a$ and $b' < b$ prevent reflections. Again using the shifting argument, one observes that this is the sufficient set of transformations to be examined. The size of this set is $O((mn)^2)$, where $m = |A|$ and $n = |B|$. To find the optimum transformation f, we construct all M_f for $f \in F$ incrementally, and choose the f that corresponds to the largest M_f: For each representative translation $b' - a'$, where $a' \in A$ and $b' \in B(\epsilon)$, construct the set of scale ranges $R(a', b') = \{[\frac{b-\epsilon-b'}{a-a'}, \frac{b+\epsilon-b'}{a-a'}] \mid a \in A, b \in B\}$. Sort the endpoints of ranges in $R(a', b')$ into increasing order, and scan through them incrementing and decrementing a counter analogously as explained before. This gives the optimum scale for the fixed translation. Repeating the process for all representative translations gives the overall optimum transformation. Noticing that the scale ranges corresponding to a fixed $a \in A$ can be obtained in sorted order by scanning through sorted B, the algorithm can be implemented to run in $O((mn)^2 \log m)$ time by merging the m sorted lists at each phase.

4.1 One-to-one Mapping Case

The solution M_f obtained with the above algorithm does not necessarily define a proper mapping between A and B; a point of A may be mapped into ϵ distance from several points of B, and vice versa. In fact, on most instances there is a degenerate optimum solution mapping all points of A into ϵ-distance from one point of B. In applications, such degenerated cases can be avoided by restricting the search space. However, a more rigorous way to define the problem is to search for M_f that contains the largest one-to-one mapping.

A brute-force algorithm to solve the one-to-one mapping case is as follows: At each phase of the previous algorithm that constructs sets M_f incrementally, construct a bipartite graph G_f having edges between $a \in A$ and $b \in B$ if and only if $(a, b) \in M_f$. Solve the maximum matching problem on each G_f, and choose f corresponding to the overall largest maximum matching.

Notice that the graphs G_f change only by one edge at each incremental step. Alt et al. [3, p. 246] describe a solution to a similar geometric problem exploiting this fact: As the maximum matching can only change by one at each phase, it is enough to check whether the new graph has an augmenting path. If so, use it to produce the new maximum matching. Otherwise the maximum matching does not change. Checking for an augmenting path takes $|M_f|$ time. In the worst case each M_f is of size $O(mn)$, and the whole algorithm takes $O((mn)^3)$ time.

The above technique yields the best known upper bound for the problem studied in [3]. Many consequent papers have studied different variations of the problem to avoid the costly maximum matching computation, like assuming disjoint error regions, transformations minimizing the Hausdorff distance, etc. For references, see survey [2].

Our problem, however, has an extra property that allows a more efficient way to find the maximum matchings. We will next show how to find the maximum matching in $O(m + n)$ time using a greedy method. To describe this solution, we first introduce some helpful notions to characterize our problem.

We say that a binary matrix $B(1 \ldots m, 1 \ldots n)$ containing values $\mathbf{0}$ and $\mathbf{1}$ is a *staircase matrix* if the following conditions hold:

(i) Each row of the matrix contains at most one *run* of $\mathbf{1}$s, i.e. a maximal range of consecutive cells each containing value $\mathbf{1}$.

(ii) Let i' and i, $i' < i$ be two rows containing a run of $\mathbf{1}$s. Let the run at row i' cover indexes $c_{i'}, c_{i'} + 1, \ldots, d_{i'}$ and the run at row i cover indexes $c_i, c_i + 1, \ldots, d_i$. Then $c_{i'} \leq c_i$ and $d_{i'} \leq d_i$.

Notice that from *(i)* and *(ii)* follows identical conditions on columns, i.e. B is a staircase matrix if and only if B^{T} is staircase matrix.

Let $A = a_1 a_2 \cdots a_m$ and $B = b_1 b_2 \cdots b_n$ be the two point sets to be matched. We assume that the point sets are given sorted in ascending order. Let us consider a fixed M_f such that $f(x) = s(x - a') + b'$, where $s = \frac{b-b'}{a-a'} \geq 0$. Consider a *match matrix* $M(1 \ldots m, 1 \ldots n)$ having $M(i,j) = \mathbf{1}$ if $(a_i, b_j) \in M_f$, otherwise $M(i,j) = \mathbf{0}$. It follows easily from definitions (proof omitted):

Lemma 2 *The match matrix M is a staircase matrix.*

On fixed translation and scale, our problem reduces to finding a maximum size one-to-one matching R of rows and columns of M such that for each $(i,j) \in R$ holds $M(i,j) = \mathbf{1}$. Let R^* be one such maximum matching. We next show that one can obtain in $O(m+n)$ time a matching R that is as good as any R^*. To show this, we first prove (in Lemma 3) that there is always an *order-preserving* matching $\hat{R}$ that is as good as R^*; matching R is order-preserving if, for every pair $(i', j'), (i,j) \in R$, we have $i' \leq i$ if and only if $j' \leq j$. Then Lemma 4 constructs the algorithm.

Lemma 3 *There is a maximum matching $\hat{R}$ that is order-preserving.*

Proof. We show that there is an algorithm to convert any maximum matching R^* into an equally good order-preserving matching in finite number of steps: Let $\check{R}$ be the set of all pairs in R^* that have at least one *conflict* with a pair in R^*: $(i', j') \in \check{R}$ if and only if there is $(i,j) \in R^*$ such that pairs (i', j') and (i,j) conflict the order-preserving condition. Let $(i_{min}, j') \in \check{R}$ be the pair having the smallest index i_{min}, and (i, j_{min}) the conflicting pair for (i_{min}, j') having the smallest index j_{min}. Since $M(i_{min}, j') = M(i, j_{min}) = \mathbf{1}$, we infer $M(i_{min}, j_{min}) = M(i, j') = \mathbf{1}$ from the staircase property. Hence, we can exchange the pairs to remove the conflict, i.e. $R^* \leftarrow (R^* \setminus \{(i_{min}, j'), (i, j_{min})\}) \cup \{(i_{min}, j_{min}), (i, j')\}$. After the exchange, R^* is still a maximum matching, and contains one more pair, namely (i_{min}, j_{min}), that is not conflicting with any other pair. Moreover, when the process is repeated with the new R^*, the new set $\check{R}$ contains only pairs (i,j) such that $i > i_{min}$, where (i_{min}, j') is the pair selected in the previous step. This follows from the fact that R^* is a one-to-one matching. Thus, after at most $|A|$ rounds, set $\check{R}$ is empty, and R^* is order-preserving and has the same matching score as the original one. □

Lemma 4 *A maximum matching R can be obtained greedily in $O(m+n)$ time.*

Proof. Let $\hat{R}$ be any a order-preserving maximum matching. Such $\hat{R}$ exists by Lemma 3. Let (i', j') be the first pair in $\hat{R}$, having the smallest index i'. We can replace it by $(i', c_{i'})$ where c_x denotes the first column having $M(x, c_x) = \mathbf{1}$ at row x. Let (i,j) be the second pair in $\hat{R}$. We can replace it by $(i, \max(c_{i'} + 1, c_i))$. Such replacement is always possible due to the staircase property. The same replacement can be applied inductively, obtaining a new order-preserving maximum matching, where the matching column for each row in the originating matching $\hat{R}$ is picked greedily. To complete the argument, we note that one can pick the rows also greedily due to the staircase property without changing the cardinality of the matching: We can traverse row by row choosing the first available matching column, comparing three values: previous matched column, start of the run of $\mathbf{1}$s at the row, and end of the run of $\mathbf{1}$s at the row. This takes overall $O(m+n)$ time, and we obtain a maximum matching R. □

Recall the incremental algorithm that updates the graph G_f scanning scales from left to right for a fixed translation. We can instead represent the graph G_f as a match matrix M. As proven before, M is a staircase matrix in each scale. Deleting or inserting an edge in G_f corresponds to updating the value of a cell in M. Each update extends or reduces a run of $\mathbf{1}$s at some row, and hence we can maintain for each row pointers to the start and end of the run in constant time. We can repeat the greedy algorithm of Lemma 4 at each scale. We conclude:

Theorem 5 *The linear one-dimensional one-to-one point set matching problem on two point sets A and B of sizes m and n, respectively, can be solved in $O((mn)^2(m+n))$ time.*

Acknowledgment

We wish to thank the anonymous referee pointing out the reference [6].

References

[1] P. K. Agarwal and J. Erickson. Geometric range searching and its relatives. In *Advances in Discrete and Computational Geometry*, Contemporary Mathematics 223, Am. Math. Soc. Press, 1999, pp. 1–56.

[2] H. Alt and L. Guibas. Discrete Geometric Shapes: Matching, Interpolation, and Approximation, In *Handbook of Computational Geometry*, Elsevier, pp. 121–153, 1999.

[3] H. Alt, K. Mehlhorn, H. Wagener, and E. Welzl. Congruence, Symmetry, and Similarity of Geometric Objects. *Discr. & Comp. Geom.*, 3(1988):237–256.

[4] S. Böcker. Calibrating Time-of-Flight Mass Spectra. Manuscript, October 2004.

[5] B. Chazelle. Cutting hyperplanes for divide-and-conquer. *Discr. & Comp. Geom.*, 9(1993), 145–158.

[6] F. Y. L. Chin, C. A. Wang, and F. L. Wang. Maximum Stabbing Line in 2D Plane. In *Proc. COCOON*, Springer-Verlag LNCS 1627, pp. 379–388, 1999.

The Fastest Way to View a Query Point in Simple Polygons

Ramtin Khosravi[*] Mohammad Ghodsi[†]

Abstract

In this paper, we study the problem of finding the shortest path from a given source point in a simple polygon to some point visible from a given query point. We will present an algorithm based on the notion of funnels in simple polygons. The algorithm preprocesses the input containing a simple polygon and a source point to produce a data structure to answer the queries in logarithmic time. The time and space required for preprocessing is quadratic in size of the simple polygon.

1 Introduction

The shortest path problem is a well-known problem for the domain of simple polygons. The problem is studied with several variations. One of the these variations is to constrain the shortest path to view a query point from at least one point on the path [6, 5]. Another similar constraint is studied in [4] where the path is required to meet a target polygon where an $O(n)$ algorithm is given for the problem. In this paper, we study the problem of finding the shortest path must be taken from a given source point in a simple polygon to view a query point.

To define the problem more precisely, let P be a simple polygon with n vertices. Suppose a source point s is given in P. The goal is to preprocess the input to answer queries of this type: given a query point $q \in P$, find the shortest distance one needs to travel from s to see q. More precisely, we want to find a point c visible from q that has the shortest distance from s. Note that if the query point q is visible from s, the point c is the s itself, so throughout the paper, we assume the query point q is given somewhere outside the visibility polygon of s.

The query can be answered in $O(n)$ time without preprocessing [4], so our goal is to find a logarithmic query time. Since the complexity of the path may be $O(n)$, we define two types of queries: one to find out the shortest distance, and another to report the shortest path. Our goal is to answer the first type of queries in logarithmic time, and the second type in $O(k + \log n)$, where k is the length of the optimal path. In this paper, we consider the first type of query and will provide comments on the second type when necessary.

The algorithm relies on the notion of funnel defined in [1]. In section 2 we study some properties of the funnel related to the problem under considerations. Our algorithm, presented in section 3, is based on the fact that the desired path always ends on a window of the visibility polygon of q. So, during the preprocessing phase, we compute a partition of all possible windows of visibility polygons in P into sets such that knowing the set a window belongs to, we can answer the ending point of the path efficiently.

2 Basic Properties

We use the notation V_p for the visibility of a point $p \in P$, $\pi(x, y)$ for the shortest path between two points x and y inside P, SPT(s) for the shortest path tree from s, and SPM(s) for the shortest path map of P with respect to s. Removing V_q from P results in a number of disconnected regions we call invisible regions. Each invisible region has exactly one edge in common with V_q called a *window*. Since s is invisible from q, it lies in an invisible region. It is easy to see that the point c lies on the window w separating s from V_q. More precisely, c is the point on w that has the shortest distance to s (Fig. 1). From now on, we refer to such a point c as the *optimal point* $c(w)$ to show explicitly the window it belongs to.

We use the notion of *funnel* as defined in [1]. Assume a and b be the endpoints of w. Define the funnel $F(w)$ as $\pi(r, a) \cup \pi(r, b)$ where r is the deepest common ancestor of a and b in the last common vertex between the two paths $\pi(s, a)$ and $\pi(s, b)$ when considered from s to a and b (Fig. 2). We assume the vertices on the funnel are named $a = v_0, v_1, \ldots, v_k, v_{k+1} = b$ in the ordered traversal from a to b. The region enclosed between $F(w)$ and w can be decomposed into triangular regions by extending the edges of $F(w)$ to intersect w. Assume the extension of the edge $v_i v_{i+1} (0 \le i \le k)$ intersects w in x_i (hence, $x_0 = a$ and $x_k = b$). The shortest path from s to points on the segment $x_i x_{i+1}$ passes through v_i as the last vertex. Denote the sequence of angles between the extension edges and the window w by $(\theta_0, \theta_1, \ldots, \theta_k)$, such that $\theta_i = \angle b x_i v_i (0 \le i \le k-1)$ and θ_k is $180° - \angle abv_k$.

[*]Department of Computer Engineering, Sharif University of Technology, and IPM School of Computer Science, ramtin@mehr.sharif.edu

[†]Department of Computer Engineering, Sharif University of Technology, and IPM School of Computer Science, ghodsi@.sharif.edu

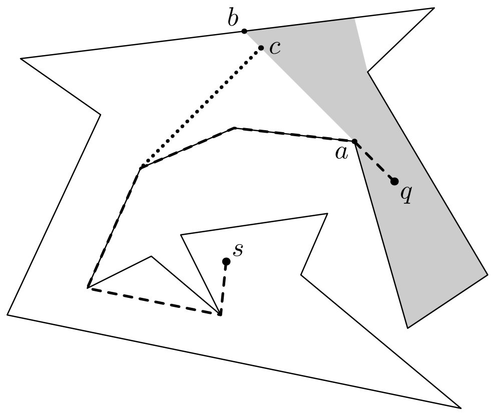


Figure 1: The shaded part is V_q. The window ab separates s and V_q. The shortest path between s and q is shown with heavy dashed segments. c is the point with shortest distance to s that is visible from q.

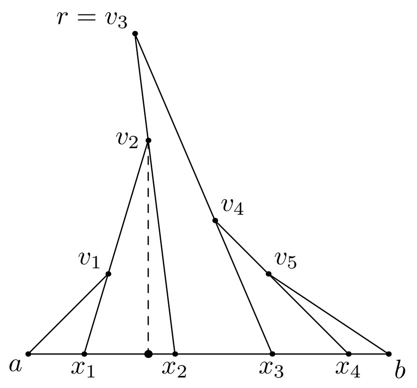


Figure 2: The funnel $F(w)$ over the window $w = ab$. The optimal point $c(w)$ is the foot of the perpendicular from v_2 to ab.

Outward convexity of the paths from the cusp of the funnel to its endpoints leads to the following observation:

Observation 1 *The sequence of angles between the extension edges and the window w $(\theta_0, \theta_1, \ldots, \theta_k)$ is an increasing sequence.*

We can characterize the optimal contact point $c(w)$ in the following way. For the sequence of angles mentioned in the above observation, one of the following cases holds:

1. There exists an angle $\theta_i = 90°$. In this case, $c(w) = x_i$.
2. There exists a pair of adjacent angles $\theta_i < 90°$ and $\theta_{i+1} > 90°$. In this case, $c(w)$ is the foot of the perpendicular from v_{i+1} to w.
3. All angles in the sequence are greater than $90°$. In this case, $c(w) = a$.
4. All angles in the sequence are less than $90°$. In this case, $c(w) = b$.

3 The Algorithm

When receiving a query point q, we take the following two steps:

1. Compute the window w that separates s from V_q.
2. Compute the optimal point $c(w)$.

Both steps must be done in logarithmic time to have an efficient query-time. To find the window in the first step, observe that the window separating s from V_q is specified by the last vertex of P of the shortest path from s to q (Fig. 1). Thus, having computed SPM(s) during the preprocessing phase, we can find the window w in $O(\log n)$ time using a standard point-location algorithm. For the second step, we must precompute the optimal contact point on all segments in P that can be a window of a query point. We refer to such a segment of P as a *separating window.* Informally, a separating window is a window of the visibility polygon of an arbitrary point x that separates from V_x.

To specify the set of all windows separating s from possible query points, we consider each reflex vertex of P and find the set of separating windows having that vertex as an endpoint. Assume a is a reflex vertex of P. Considering all possible query points inside P, we may have a set of windows associated with a which are defined by the rays emanating from a in two angular intervals between the extension of each edge incident to a and the other edge (Fig. 3(a)). Not all the windows defined by the two intervals mentioned are separating. To restrict the set to separating windows, consider x as the last vertex the shortest path $\pi(s, a)$ passes through. If x lies outside both angular intervals, then there is no separating window around a. Otherwise, assume that it lies inside the interval defined by the extension of e_1 and e_2 where e_1 and e_2 are the edges incident to a (Fig. 3(b)). The angular interval defining the set of separating windows around a is bounded by the extension of e_1 and the ray emanating from a passing through the x. We denote this set of separating windows by *Sep*(a).

For a reflex vertex a, our goal is to partition *Sep*(a) into a number of sets such that knowing the set w belongs to, we can find the optimal point $c(w)$ efficiently. To do this, we use a radial sweep around a. There are two kinds of events in the sweep process: angles at which the last vertex of $\pi(s, c)$ changes (*interval events*), and angles at which the structure of the funnel changes (*funnel events*). These events defines the desired partition. We consider the two types of events subsequently.

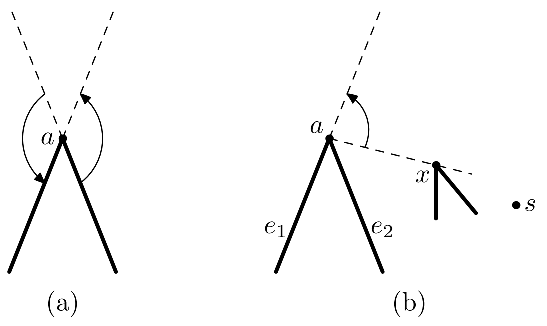


Figure 3: (a) Two angular intervals defining windows around a reflex vertex a. (b) The single interval defining separating windows around a. x is the last vertex on $\pi(s, a)$.

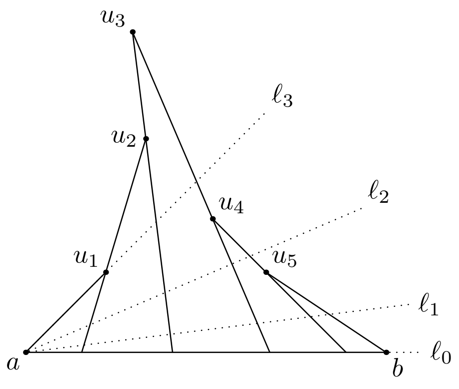


Figure 4: Partition of the separating windows around a. The angular interval between ℓ_0 and ℓ_3 defines the set of all separating windows. ℓ_1 and ℓ_2 are perpendiculars to v_3v_2 and v_3v_4 extensions respectively.

3.1 Interval Events

Suppose $Sep(a)$ is defined by the angular interval $[\alpha, \alpha']$. For $\alpha \leq \varphi \leq \alpha'$, let w_φ denote the window with endpoint a and angel φ. Consider the perpendiculars from a to two adjacent extension edges v_ix_i and $v_{i+1}x_{i+1}$. For a window w_ϕ that lies between these two perpendiculars, we can say $\theta_i < 90°$ and $\theta_{i+1} > 90°$. So, the optimal point $c(w_\phi)$ is the foot of the perpendicular from v_{i+1} to w_ϕ. For example, in Figure 4, ℓ_1 and ℓ_2 are perpendiculars to the extensions of v_3v_2 and v_3v_4 respectively. Hence, for an arbitrary window between ℓ_1 and ℓ_2, the optimal point is the foot of the perpendicular from v_3 to the window. This way, the set of perpendiculars to the extension edges defines the interval events in which the last vertex of $\pi(s, c)$ changes. Since rotating a window towards the cusp of the funnel reduces the angles made between the extension edges and the window, the number of interval events is bounded by $O(n)$.

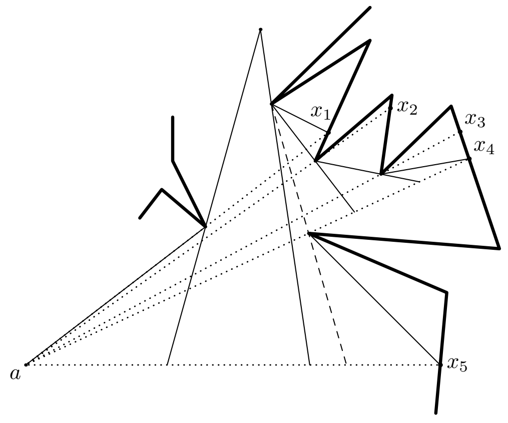


Figure 5: Funnel events for a reflex vertex a. The angular interval between ax_1 and ax_5 defines the set of all separating windows. The sweep starts from ax_1. Thick solid lines are the edges of P, thin solid lines are the extension segments, dotted lines show the swap window in different positions, and the dashed line is an extension segment added after visiting ax_4.

3.2 Funnel Events

Observe that the set of separating windows ($Sep(a)$) is a subset of the visibility polygon of a, V_a bounded to w_α and $w_{\alpha'}$. It is easy to see that the moments at which the sweep window passes through the vertices of V_a, the structure of the funnel changes. So, for any reflex vertex, there are at most $O(n)$ funnel events. The problem is to update the funnel efficiently at these moments. We assume SPT(s) is computed priory as well as SPM(s) and V_a such that we may traverse the vertices of V_a in order. The key to efficient update is to start the sweep process from the separating window closest to s. According to our notation, either w_α or $w_{\alpha'}$ has this property. Here, we assume w_α is the one.

Initially, we compute the funnel over w_α. As the sweep window rotates around a, we may encounter a new vertex from V_a, such as p. If p is not a reflex vertex of P, the effect of this event is only the change in the edge of P on which the non-fixed endpoint of the sweep window moves. Otherwise, we have encountered a new node in SPT(s) and this may cause a vertex added to the funnel. It is possible that the newly added vertex deletes parts of the funnel too. This happens when the parent of the added vertex was not a leaf before adding the new vertex. For example, in Figure 5, the initial funnel is built over ax_1. New vertices are added to the funnel as the sweep window meets the points x_2 and x_3. At x_4, a new vertex is added with the dashed extension segment introduced and some vertices are deleted from the funnel.

Note that we must keep track of the last reflex vertex of V_a visited. Having this, we can compute the change that should be made to the funnel in constant time. Also, we must have the vertices of the initial funnel in sorted order. This is possible if we make the recursive calls made during the DFS traversal in the shortest path algorithm sorted in some fixed direction (e.g. clockwise). Since the funnel structure used in the algorithm is stored in a *finger search tree* [2], the list of vertices in the funnel can be arranged in sorted order in linear time.

For a vertex a, we have a set of $O(n)$ angular intervals in sorted order so that upon receiving a query window, we can find the interval it belongs to using binary search. Finding the interval the window belongs to, we can compute its optimal point in constant time.

Summarizing the above discussions, we take the following steps during the preprecessing phase:

1. Compute SPT(s) and SPM(s)
2. For each reflex vertex a do the following:
   (a) Compute $Sep(a)$.
   (b) Compute the portion of V_a bounded by $Sep(a)$.
   (c) Compute the initial funnels and the extension edges
   (d) Perform the radial sweep starting from the closest separating window to s.

The first step can be done in $O(n)$ time using the algorithm of [1]. Step (a) involves finding the last vertex on $\pi(s, a)$ and $\pi(t, a)$ which can be done in $O(\log n)$. The visibility computation in step (b) can be done using the linear time algorithm of [7, 3]. The funnel and the extension edges in step (c) are derived directly from the SPMs in $O(n)$ time. Step (d) involves computing and handling both types of events which are $O(n)$ in total and needs constant time per event. This leads to $O(n)$ preprocess for each reflex vertex. The partition for the reflex vertex is of size $O(n)$. As there are $O(n)$ reflex vertices, the total preprocessing time is $O(n^2)$ and $O(n^2)$ space is needed to store the partitions.

Upon receiving a query, we find the window ($w = ab$) separating s from the query point q in $O(\log n)$ time (by finding q in SPM(s)). We perform a binary search on the partition associated with the reflex vertex a. Finding the optimal point $c(w)$ takes constant time. So we have our main result as the following:

Theorem 1 *Given a simple polygon P and a source point s inside P, we can preprocess the input in $O(n^2)$ time and the same space to answer the queries of this type in $O(\log n)$ time: given a query point q invisible from s, find the point c with minimum distance to s that is visible from q. The path from s to c can be reported in additional time proportional to the length of the path.*

4 Conclusion

We presented an algorithm to preprocess the input polygon in $O(n^2)$ time and space to answer the queries to find the shortest distance to a point visible from a query point in logarithmic time. Possible extensions to this problem involves studying the problem when the path is required to end to a given target point (t) and the query point must be seen from a point during the path. This makes us study the behavior of the optimal point based on two the funnels related to s and t made over the window which is not a direct extension of the behavior regarding one funnel. Another extension is to consider the query not just a point, but another geometric object like a segment. For a segment, the algorithm can still work considering the appropriate window from the weak visibility polygon of the segment. For more complex objects like a polygon, the challenge is to find the appropriate window from the visibility polygon. Once the window is computed, the rest of the algorithm is similar to that of a point.

References

[1] 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.

[2] L. J. Guibas, E. McCreight, M. Plass, and J. Roberts. A new representation for linear lists. In *Proc. 9th Annu. ACM Sympos. Theory Comput.*, pages 49–60, 1977.

[3] B. Joe and R. B. Simpson. Correction to Lee's visibility polygon algorithm. *BIT*, 27:458–473, 1987.

[4] R. Khosravi and M. Ghodsi. Shortest paths in simple polygons with polygon-meet constraints. *Inform. Process. Lett.*, 91:171–176, 2004.

[5] R. Khosravi and M. Ghodsi. Shortest paths with single-point visibility constraints. *submitted to Scientia Iranica*, 2004.

[6] R. Khosravi, M. Ghodsi, and M. Taghdiri. Shortest point-visible paths on polyhedral surfaces. In *Proc. of the 10th International Conference on Computing and Information (ICCI'2000)*, 2000.

[7] D. T. Lee. Visibility of a simple polygon. *Comput. Vision Graph. Image Process.*, 22:207–221, 1983.

The Minimum Area Spanning Tree Problem

Paz Carmi[*] Matthew J. Katz[†]

Abstract

We define and study the Minimum Area Spanning Tree (MAST) problem. Given a set $\mathcal{P}$ of n points in the plane, find a spanning tree $\mathcal{T}$ of $\mathcal{P}$ of minimum area, where the area of a spanning tree is the area of the union of the $n-1$ disks whose diameters are the edges in $\mathcal{T}$. We prove that the minimum spanning tree of $\mathcal{P}$ is a constant-factor approximation for MAST. We then apply this result to obtain a constant-factor approximation for the Minimum Area Range Assignment (MARA) problem and for the Minimum Area Connected Disk Graph (MACDG) problem. The former problem is a variant of the power assignment problem in radio networks, and the latter problem is a related natural problem.

1 Introduction

We introduce and study the Minimum Area Spanning Tree (MAST) problem. Given a set $\mathcal{P}$ of n points in the plane, find a spanning tree of $\mathcal{P}$ of minimum area, where the area of a spanning tree $\mathcal{T}$ of $\mathcal{P}$ is defined as follows. For each edge e in $\mathcal{T}$ draw the disk whose diameter is e. The *area* of $\mathcal{T}$ is then the area of the union of these $n-1$ disks. Although this problem seems natural (see also applications below), we are not aware of any previous work on this problem.

One of the main results of this paper (presented in Section 2) is that the minimum spanning tree of $\mathcal{P}$ is a constant-factor approximation for MAST. This is an important property of the minimum spanning tree as is shown below. (See, e.g., [3, 5] for background on the minimum spanning tree.)

We apply the result above to two problems from a class of problems that has received considerable attention. The first problem is a variant of the power assignment problem (also called the range assignment problem). Let $\mathcal{P}$ be a set of n points in the plane, representing n transmitters-receivers (or transmitters for short). In the standard version of the power assignment problem one needs to assign transmission ranges to the transmitters in $\mathcal{P}$, so that (i) the resulting communication graph is strongly connected (that is, the graph in which there exists a directed edge from $p_i \in \mathcal{P}$ to $p_j \in \mathcal{P}$ if and only if p_j lies in the disk D_{p_i} is strongly connected, where the radius of D_{p_i} is the transmission range, r_i, assigned to p_i), and (ii) the total power consumption (i.e., the cost of the assignment of ranges) is minimal, where the total power consumption is $\sum_{p_i \in \mathcal{P}} \text{area}(D_{p_i})$.

The power assignment problem is known to be NP-hard (see Kirousis et al. [6] and Clementi et al. [2]). Kirousis et al. [6] also obtain a 2-approximation for this problem, based on the minimum spanning tree of $\mathcal{P}$, and this is the best approximation known.

Consider now the variant of the power assignment problem where the second requirement above is replaced by (ii') the area of the union of the disks $D_{p_1}, \ldots, D_{p_n}$ is minimal. We refer to this problem as the Minimum Area Range Assignment (MARA) problem. In general, the presence of a foreign receiver (whether friendly or hostile) in the region $D_{p_1} \cup \cdots \cup D_{p_n}$ is undesirable, and the smaller the area of this region, the lower the probability that such a foreign receiver is present. In Section 3 we prove that the range assignment of Kirousis et al. (that is based on the minimum spanning tree) is also a constant-factor approximation for MARA.

Another related and natural problem for which we obtain a constant-factor approximation (in the full version of this paper) is the following. Let $\mathcal{P}$ be a set of n points in the plane. For each point $p \in \mathcal{P}$, draw a disk D_{p_i} of radius 0 or more, such that, (i) the resulting disk graph is connected (that is, the graph in which there exists an edge between $p_i \in \mathcal{P}$ and $p_j \in \mathcal{P}$ if and only if $D_{p_i} \cap D_{p_j} \neq \emptyset$ is connected), and (ii) the area of the union of the disks $D_{p_1}, \ldots, D_{p_n}$ is minimal. We refer to this problem as the Minimum Area Connected Disk Graph (MACDG) problem. (See, e.g., [4, 7] for background on intersection graphs and disk graphs in particular.)

A potentially interesting property concerning the area of the minimum spanning tree that is obtained as an intermediate result in Section 2 is that the depth of the arrangement of the disks corresponding to the edges of the minimum spanning tree is bounded by some constant. Notice that this property does not follow immediately from the fact that the degree of the minimum spanning tree is at most 6, as is shown in Figure 2.

Finally, all the above results hold in any fixed dimension d (with obvious modifications).

[*]Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel, carmip@cs.bgu.ac.il. Partially supported by a Kreitman Foundation doctoral fellowship.

[†]Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel, matya@cs.bgu.ac.il. Partially supported by grant no. 2000160 from the U.S.-Israel Binational Science Foundation.

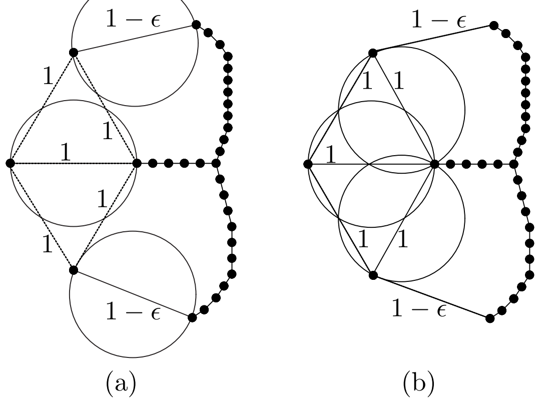


Figure 1: A minimum spanning tree is not necessarily a minimum area spanning tree. (a) The minimum spanning tree. (b) A minimum area spanning tree.

2 MST is a constant-factor approximation for MAST

Let $\mathcal{T}$ be any spanning tree of $\mathcal{P}$. For an edge e in $\mathcal{T}$, let $D(e)$ denote the disk whose diameter is e. Put $D(\mathcal{T}) = \{D(e) \mid e \text{ is an edge in } \mathcal{T}\}$, $\bigcup_{\mathcal{T}} = \bigcup_{e \in \mathcal{T}} D(e)$, and $\sigma_{\mathcal{T}} = \sum_{e \in \mathcal{T}} \text{area}(D(e))$. Let MST be a minimum spanning tree of $\mathcal{P}$. MST is not necessarily a solution for the Minimum Area Spanning Tree (MAST) problem; see Figure 1. In this section we prove that MST is a constant-factor approximation for MAST, that is, $\text{area}(\bigcup_{\text{MST}}) = O(\text{area}(\bigcup_{\text{OPT}}))$, where OPT is an optimal spanning tree, i.e., a solution to MAST.

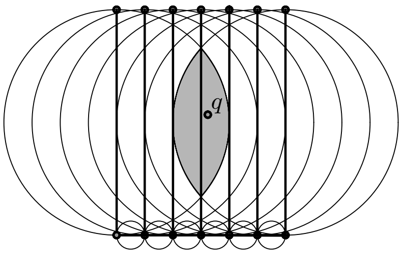


Figure 2: A spanning tree $\mathcal{T}$ of degree 3, and a point q (in the interior of a cell of the arrangement of the disks in $D(\mathcal{T})$) of depth $O(n)$.

We begin by showing another interesting property of MST, namely, that the depth of any point p in the interior of a cell of the arrangement of the disks in $D(\text{MST})$ is bounded by a small constant. This property does not follow directly from the fact that the degree of MST is bounded by 6; see Figure 2. Let MST_p be a minimum spanning tree for $\mathcal{P} \cup \{p\}$. We need the following known and easy claim.

Claim 1 *We may assume that there is no edge (a, b) in MST_p, such that, (a, b) is not in MST and both a and b are points of $\mathcal{P}$.*

Proof. Assume there is such an edge (a, b) in MST_p. Consider the path in MST between a and b. At least one of the edges along this path is not in MST_p. Let e be such an edge. $|e| \le |(a, b)|$, since otherwise (a, b) would have been chosen by the algorithm that computed MST (e.g., Kruskal's minimum spanning tree algorithm [1]). Therefore, we may replace the edge (a, b) in MST_p by e, without increasing the total weight of the tree. □

An immediate corollary of this claim is that we may assume that if e is an edge in MST_p but not in MST, then one of e's endpoints is p.

The proof of the following lemma appears in the full version of this paper.

Lemma 1 $\sigma_{\text{MST}} \le 5\,\text{area}(\bigcup_{\text{MST}})$.

Remark. A more careful analysis allows one to replace the 5 in the statement of the lemma above by 4 or perhaps even by 3. However, for our purpose 5 is good enough.

Let OPT be an optimal spanning tree of $\mathcal{P}$, i.e., a solution to MAST. We use OPT to construct another spanning tree, ST, of $\mathcal{P}$. Initially ST is empty. Let e_1 be the longest edge in OPT. Draw two concentric disks C_1 and C_1^3 around the mid point of e_1 of diameters $|e_1|$ and $3|e_1|$, respectively. Compute a minimum spanning tree of the points of $\mathcal{P}$ lying in C_1^3, using Kruskal's algorithm [1]. Whenever an edge is chosen by Kruskal's algorithm, it is immediately added to ST. See Figure 3. Let S_1 denote the set of edges that have been added to ST in this (first) iteration.

Next, let e_2 be the longest edge in OPT, such that at least one of its endpoints lies outside C_1^3. As for e_1, draw two concentric disks C_2 and C_2^3 around the mid point of e_2 of diameters $|e_2|$ and $3|e_2|$, respectively. Apply Kruskal's minimum spanning algorithm to the points of $\mathcal{P}$ lying in C_2^3 with the following modification. The next edge in the sorted list of potential edges is chosen by the algorithm if and only if it is not in ST and its addition to ST does not create a cycle in ST. Moreover, when an edge is chosen by the algorithm it is immediately added to ST; see Figure 4 (a) and (b). Let S_2 denote the set of edges that have been added to ST in this iteration.

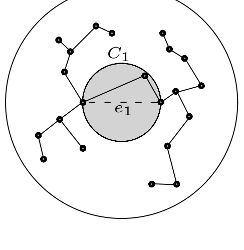
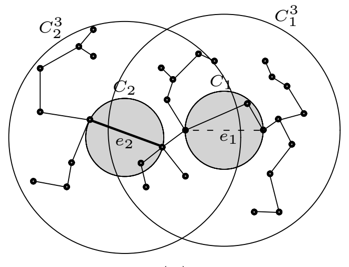
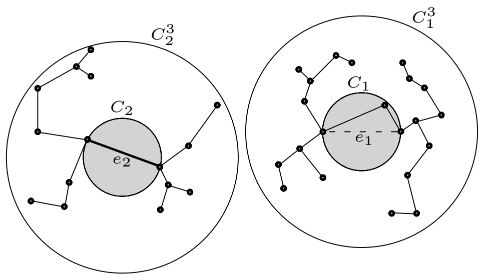


Figure 3: ST after choosing e_1.



(a)



(b)

Figure 4: ST after choosing e_1 and e_2. (a) One of the end points of e_2 is in C_1^3. (b) Both endpoints of e_2 are not in C_1^3.

In the i'th iteration, let e_i be the longest edge in OPT, such that there is no path yet in ST between its endpoints. Draw two concentric circles C_i and C_i^3 around the mid point of e_i, and apply Kruskal's minimum spanning tree algorithm with the modification above to the points of $\mathcal{P}$ lying in C_i^3. Let S_i denote the set of edges that have been added to ST in this iteration. The process ends when for each edge e in OPT there already exists a path in ST between the endpoints of e.

Claim 2 *For each i, S_i is a subset of the edge set of the minimum spanning tree* MST$_i$ *that is obtained by applying Kruskal's algorithm,* without *the modification above, to the points in C_i^3.*

Proof. Let e be an edge that was added to ST during the i'th iteration. If e is not chosen by Kruskal's algorithm (without the modification above), it is only because, when considering e, a path between its two endpoints already exists in MST$_i$. But this implies that e could not have been added to ST, since, any edge already in MST$_i$ was either also added to ST or was not added since there already existed a path in ST between its two endpoints. Thus, when e was considered by the modified algorithm it should have been rejected. We conclude that e must be in MST$_i$. $\square$

Claim 3 ST *is a spanning tree of $\mathcal{P}$.*

Proof. Since only edges that do not create a cycle in ST were added to ST, there are no cycles in ST. Also ST is connected, since otherwise there still exists an edge in OPT that forces another iteration of the construction algorithm. $\square$

Let $\mathcal{C}$ denote the set of the disks $C_1, \ldots, C_k$, and let $\mathcal{C}^3$ denote the set of the disks $C_1^3, \ldots, C_k^3$, where k is the number of iterations in the construction of ST.

Claim 4 *For any pair of disks C_i, C_j in $\mathcal{C}$, $i \neq j$, it holds that $C_i \cap C_j = \emptyset$.*

Proof. Let C_i be any disk in $\mathcal{C}$. We show that for any disk $C_j \in \mathcal{C}$ such that $j > i$, $C_i \cap C_j = \emptyset$. From the construction of ST it follows that $|e_j|$, the diameter of C_j, is smaller or equal to $|e_i|$, the diameter of C_i. Moreover, at least one of the endpoints of e_j lies outside C_i^3 (since if both endpoints of e_j lie in C_i^3, then, by the end of the i'th iteration, a path connecting between these endpoints must already exist in ST). Therefore, C_j whose center coincides with the mid point of e_j, cannot intersect C_i. $\square$

Claim 5 $\sigma_{\mathrm{ST}} = O(\mathrm{area}(\bigcup_{\mathrm{OPT}}))$.

Proof. Recall that $\sigma_{\mathrm{ST}} = \Sigma_i \sigma_{S_i}$, where $\sigma_{S_i} = \Sigma_{e \in S_i} \mathrm{area}(D(e))$. We first show by the sequence of inequalities below that $\sigma_{S_i} = O(\mathrm{area}(C_i))$.

$$\begin{aligned}\sigma_{S_i} &\leq^1 \sigma_{\mathrm{MST}_i} \leq^2 5\,\mathrm{area}(\bigcup_{\mathrm{MST}_i}) =^3 O(\mathrm{area}(C_i^3)) \\ &=^4 O(\mathrm{area}(C_i))\,.\end{aligned}$$

The first inequality follows immediately from Claim 2. The second inequality is true by Lemma 1. Consider Equality 3. Since all edges in MST$_i$ are contained in C_i^3, it holds that $\bigcup_{\mathrm{MST}_i}$ is contained in a disk that is obtained by expanding C_i^3 by some constant factor. It follows that $\mathrm{area}(\bigcup_{\mathrm{MST}_i}) = O(\mathrm{area}(C_i^3)) = O(\mathrm{area}(C_i))$.

Therefore,

$$\sigma_{\mathrm{ST}} = \Sigma_i \sigma_{S_i} = \Sigma_i O(\mathrm{area}(C_i))\,.$$

But according to Claim 4, the latter expression is equal to $O(\mathrm{area}(\bigcup_{\mathcal{C}}))$, and, since $\mathcal{C}$ is a subset of $D(\mathrm{OPT})$, we conclude that $\sigma_{\mathrm{ST}} = O(\mathrm{area}(\bigcup_{\mathrm{OPT}}))$. $\square$

We are now ready to prove the main result of this section.

Theorem 2 MST *is a constant-factor approximation for* MAST, *that is,* $\mathrm{area}(\bigcup_{\mathrm{MST}}) \leq c \cdot \mathrm{area}(\bigcup_{\mathrm{OPT}})$, *for some constant c.*

Proof.

$$\mathrm{area}(\bigcup_{\mathrm{MST}}) \leq^1 \sigma_{\mathrm{MST}} \leq^2 \sigma_{\mathrm{ST}} \leq^3 c \cdot \mathrm{area}(\bigcup_{\mathrm{OPT}})\,.$$

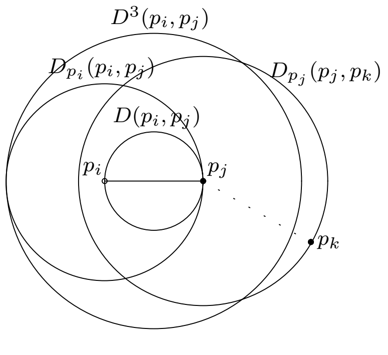


Figure 5: $(p_i,p_j) \in$ MST; $D(p_i,p_j) \in D($MST$)$; $D_{p_i}(p_i,p_j), D_{p_j}(p_j,p_k) \in$ RA; $D^3(p_i,p_j) \in D^3($MST$)$.

The first inequality is trivial. The second inequality holds for any spanning tree of $\mathcal{P}$; that is, for any spanning tree $\mathcal{T}$, $\sigma_{\text{MST}} \leq \sigma_{\mathcal{T}}$. (Since if the lengths $|e|$ of the edges are replaced with weights $\pi|e|^2/2$, we remain with the same minimum spanning tree.) The third inequality is proven in Claim 5. □

3 A constant-factor approximation for MARA

MST induces an assignment of ranges to the points of $\mathcal{P}$. Let $p_i \in \mathcal{P}$ and let r_i be the length of the longest edge in MST that is connected to p_i, then the range that is assigned to p_i is r_i. Put RA $= \{D_{p_1}, \ldots, D_{p_n}\}$, where D_{p_i} is the disk of radius r_i centered at p_i. In this section we apply the main result of the previous section (i.e., MST is a constant-factor approximation for MAST), in order to prove that the range assignment that is induced by MST is a constant-factor approximation for the Minimum Area Range Assignment (MARA) problem. That is, (i) the corresponding (directed) communication graph is strongly connected, and (ii) the area of the union of the disks in RA is bounded by some constant times the area of the union of the transmission disks in an optimal range assignment, i.e., a solution to MARA.

The first requirement above was already proven by Kirousis et al. [6], who showed that the range assignment induced by MST is a 2-approximation for the standard range assignment problem. Let OPT^R denote an optimal range assignment, i.e., a solution to MARA. It remains to prove the second requirement above.

Claim 6 area$(\bigcup_{RA}) \leq 9\,$area$(\bigcup_{\text{MST}})$.

Proof. We define an auxiliary set of disks. For each edge e in MST, draw a disk of diameter $|3e|$ centered at the mid point of e. Let $D^3($MST$)$ denote the set of these $n-1$ disks; see Figure 5. We now observe that area$(\bigcup_{RA}) \leq$ area$(\bigcup_{D^3(\text{MST})})$. This is true since for each $p_i \in \mathcal{P}$, $D_{p_i} = D_{p_i}(p_i,p_j)$ for some point $p_j \in \mathcal{P}$ that is connected to p_i (in MST) by an edge, and $D_{p_i}(p_i,p_j)$ is contained in the disk of $D^3($MST$)$ corresponding to the edge (p_i,p_j). Finally, clearly area$(\bigcup_{D^3(\text{MST})}) \leq 9\,$area$(\bigcup_{\text{MST}})$. □

Theorem 3 *RA is a constant-factor approximation for MARA, that is,* area$(\bigcup_{\text{RA}}) \leq c' \cdot$ area$(\bigcup_{\text{OPT}^R})$*, for some constant c'.*

Proof. The proof is based on the observation that the (directed) communication graph corresponding to OPT^R contains a spanning tree, and on the main result of Section 2. Let p be any point in $\mathcal{P}$. We construct a spanning tree $\mathcal{T}$ of $\mathcal{P}$ as follows. For each point $q \in \mathcal{P}$, $q \neq p$, compute a shortest (in terms of number of hops) directed path from q to p, and add the edges in this path to $\mathcal{T}$. Now make all edges in $\mathcal{T}$ undirected. $\mathcal{T}$ is a spanning tree of $\mathcal{P}$. For each edge (p_i,p_j) in $\mathcal{T}$, the disk $D(p_i,p_j)$ is contained either in the transmission disk of p_i (in OPT^R), or in the transmission disk of p_j (in OPT^R). Hence, $\bigcup_{\mathcal{T}} \subseteq \bigcup_{\text{OPT}^R}$.

The following sequence of inequalities completes the proof. (OPT denotes a solution to MAST.)

$$\begin{aligned}\text{area}(\bigcup_{\text{RA}}) &\leq^1 9\,\text{area}(\bigcup_{\text{MST}}) \leq^2 9c \cdot \text{area}(\bigcup_{\text{OPT}}) \\ &\leq^3 9c \cdot \text{area}(\bigcup_{\mathcal{T}}) \leq^4 9c \cdot \text{area}(\bigcup_{\text{OPT}^R}).\end{aligned}$$

The first inequality follows from Claim 6; the second inequality follows from Theorem 2; the third inequality follows from the definition of OPT; the fourth inequality was shown above. □

References

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. Introduction to Algorithms. In *MIT Press*, Second Edition, 2001.

[2] A. E. F. Clementi, P. Penna, and R. Silvestri. On the Power Assignment Problem in Radio Networks. In *Electronic Colloquium on Computational Complexity*, 2000.

[3] D. Eppstein. Spanning Trees and Spanners. In *Handbook of Computational Geometry*, pages 425–461. J.-R. Sack and J. Urrutia, eds., Elsevier, 1999.

[4] M. C. Golumbic. Algorithmic Graph Theory and Perfect Graphs. In *Academic Press*, New York, 1980.

[5] R. L. Graham and P. Hell. On the History of the Minimum Spanning Tree Problem. In *Annals of the History of Computing*, 7:43–57, 1985.

[6] L.M. Kirousis, E. Kranakis, D. Krizanc, and A. Pelc. Power Consumption in Packet Radio Networks. In *14th Annual Sympos. Theoretical Aspects of Computer Science (STACS)*, pages 363–374. LNCS 1200, 1997.

[7] T.A. McKee and F.R. McMorris. Topics in Intersection Graph Theory. In Volume 2 of *Monographs on Discrete Mathematics and Applications*, SIAM, 1999.

Spanning Trees with Few Crossings in Geometric and Topological Graphs

Christian Knauer[*] Étienne Schramm[†] Andreas Spillner[‡] Alexander Wolff[†]

Abstract

We study the problem of computing a spanning tree in a connected topological graph such that the number of crossings in the spanning tree is minimum. We show it is NP-hard to find a good approximation of the minimum number of crossings even in geometric graphs. On the other hand we show that the problem is fixed-parameter tractable and present a mixed-integer linear program formulation.

1 Introduction

Let $G(V, E)$ be an undirected graph that is embedded in the plane, so that no two edges share an unbounded number of points. We call G a *topological graph.* If all edges are straight-line embedded, then G is called a *geometric* graph.

A *crossing* $\{e, e'\}$ is a pair of edges such that $e \cap e' \not\subseteq V$. We call $\mu_{ee'} = |(e \cap e') \setminus V|$ the *multiplicity* of the crossing $\{e, e'\}$. Let $X \subseteq \binom{E}{2}$ be the set of pairs of crossings in E. Note that c edges intersecting in a single non-endpoint give rise to $\binom{c}{2}$ crossings. We will use n, m, and k as shorthand for the cardinalities of V, E, and X, respectively. We will use the notation $G(V, E, X)$ if we want to emphasize the connection between G and X. We define the *weighted number of crossings* of a subgraph $G(V, E', X')$ of G as $\sum_{\{e,e'\}\in X'} \mu_{ee'}$.

In this paper we study the problem of computing a spanning tree of a given connected topological graph such that the weigted number of crossings in the tree is minimum.

Kratochvil et al. [2] have shown that for topological graphs it is NP-hard to decide whether they contain crossing-free subgraphs of certain types, such as cycles, u–v paths, maximum matchings, or spanning trees. Jansen and Woeginger [1] have strengthened the last result by showing that it is even NP-hard to decide whether or not a *geometric* graph has a crossing-free spanning tree.

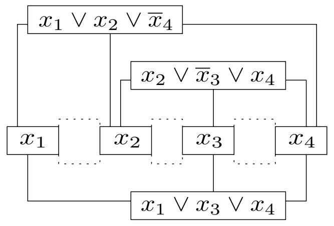

$x_1 \vee x_2 \vee \overline{x}_4$
$x_2 \vee \overline{x}_3 \vee x_4$
x_1
x_2
x_3
x_4
$x_1 \vee x_3 \vee x_4$

Figure 1: Overall structure of G_F.

That result does not rule out the existance of efficient constant-factor approximation algorithms. However, as we will show in Section 2, such algorithms do not exist unless $\mathcal{P} = \mathcal{NP}$. So we turned our attention to other possible ways to attack the problem: in Section 3 we show that the problem under consideration is fixed-parameter tractable with k being the parameter and in Section 4 we present a mixed-integer linear program (MIP) formulation. We conclude in Section 5.

2 Hardness of approximation

Let $\phi(G)$ be 1 plus the minimum number of crossings in a spanning tree of a geometric graph G. The main result of this section is the following theorem.

Theorem 1 *Given geometric graph G, it is NP-hard to approximate $\phi(G)$ within a factor of $k^{1-\varepsilon}$ for any $\varepsilon > 0$.*

Note that it is trivial to give a factor-$(k + 1)$-approximation and since a geometric graph is also a topological graph, theorem 1 obviously remains valid for topological graphs.

We obtain this result by a reduction from planar 3SAT [3]. Our reduction maps a given planar 3SAT formula F to a geometric graph G_F such that G_F has a plane spanning tree iff F has a fulfilling truth assignment. The overall structure of G_F is indicated in Figure 1 for $F = (x_1 \vee x_2 \vee \overline{x}_4) \wedge (x_2 \vee \overline{x}_3 \vee x_4) \wedge (x_1 \vee x_3 \vee x_4)$.

We have a gadget for every variable occuring in F. These gadgets are arranged along a horizontal line ℓ. We further have a gadget for every clause in F which is connected with every variable occuring in the clause. This gadget looks like a three-legged comb.

[*]Institute of Computer Science, Freie Universität Berlin, Germany, christian.knauer@inf.fu-berlin.de

[†]Fakultät für Informatik Universität Karlsruhe, Germany, http://i11www.ira.uka.de/people. Supported by grant WO 758/4-1 of the German Science Foundation (DFG).

[‡]Institute of Computer Science, Universität Jena, Germany, spillner@minet.uni-jena.de

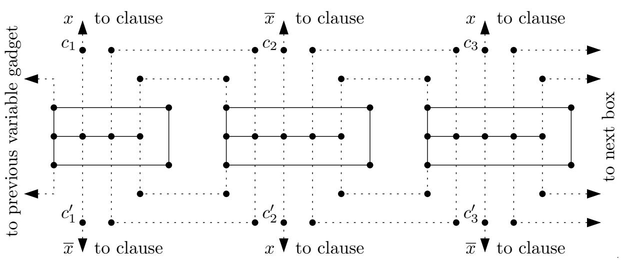


Figure 2: Part of a variable gadget.

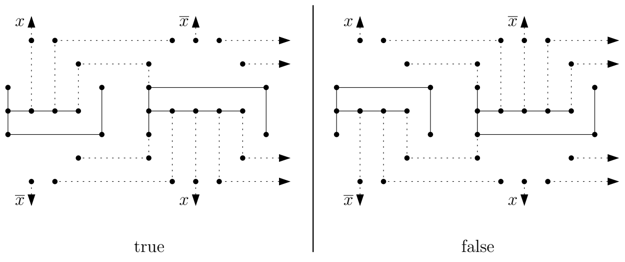


Figure 3: Spanning trees encoding true and false.

Now let's have a closer look at the gadgets. The leftmost part of the gadget for a variable x is shown in Figure 2. The gadget for x consists of at most twice as many boxes as there are clauses in F that contain x. Three of these boxes are drawn with solid edges in Figure 2. The dotted edges that emanate from the boxes fulfill three tasks. First they connect consecutive boxes within one variable gadget. Second they connect the first and last box of variable gadgets that are consecutive on the line ℓ. Third they connect boxes to clause gadgets. Each dotted edge that connects a variable gadget to a clause gadget is associated with a literal. This literal will be true if the dotted edge is part of the spanning tree of G_F.

The intended way of simulating the truth assignment of the variable x is indicated in Figure 3. The Boolean values of x correspond to the two ways in which a crossing-free spanning tree can be chosen among the edges of the gadget of x. Note that only every other box can be connected to a clause gadget above (below) ℓ. This way we ensure that according to the value of x either only the dotted edges associated to positive literals or only the dotted edges associated to negative literals can connect x to clause gadgets. Not all points of type c_i or c'_i in Figure 2 are used—only those where the variable is in fact connected to a clause gadget in G_F. For example, the gadget for x_2 in Figure 1 consists of three boxes, but only the points c_1 and c_3 are used.

A clause gadget is shown in Figure 4(a). The three boxes in the lower part of the figure belong to the gadgets of those variables that form the clause. Note that the clause gadget is connected to the rest of G_F only via the dotted edges that go into these three

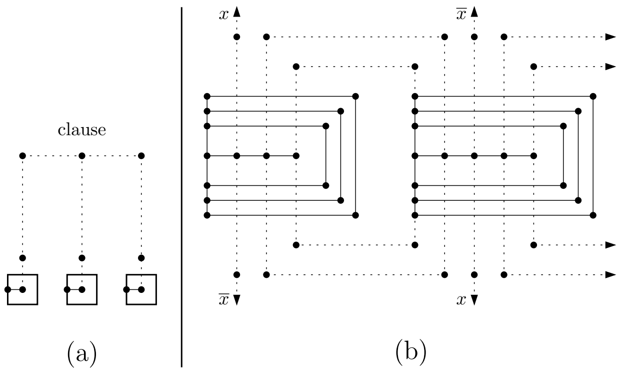


Figure 4: Clause gadget and modified variable gadget.

boxes. Thus G_F has a crossing-free spanning tree only if at least one of the three literals in the clause is true.

Up to this point our construction only proves that deciding whether or not a geometric graph has a crossing-free spanning tree is NP-hard. However, our construction is designed to allow us to apply a simple trick that yields the desired hardness of approximation. We substitute certain solid edges in every box in every variable gadget with z new edges. This is indicated for $z = 3$ in Figure 4(b).

If we set z to a polynomial in the length of F with large enough degree, our reduction remains polynomial and Theorem 1 is an immediate consequence of the following lemma.

Lemma 2 *If F has a satisfying truth assignment then $\phi(G_F) = 1$, else $\phi(G_F) \geq z$.*

Proof. First suppose F has a satisfying truth assignment. Then we select edges for a spanning tree of G_F in every variable gadget as indicated in Figure 3 according to the satisfying truth assignment. Since in every clause of F at least one literal is true under the satisfying truth assignment it is possible to connect every clause gadget in a spanning tree without any crossings. Thus $\phi(G_F) = 1$.

Now suppose F does not have a satisfying truth assignment. Consider a box B after substitution of edges as in Figure 4(b). Then B has z solid horizontal edges above (below) ℓ and at most four dotted vertical edges above (below) ℓ that intersect the former. Let T be a spanning tree of G_F with the minimum number of crossings. Let v_{top} (v_{bot}) be the number of vertical edges in T that are above (below) ℓ and intersect B. Similarly, let h_{top} (h_{bot}) be the number of horizontal edges in T that are above (below) ℓ and belong to B. Since the vertices on the right of the box can only be connected to the rest of the graph via the solid top or bottom edges, we have $h_{\text{top}} + h_{\text{bot}} \geq z$. Now suppose $v_{\text{top}} \leq v_{\text{bot}}$. Then there are $v_{\text{top}}h_{\text{top}} + v_{\text{bot}}h_{\text{bot}} \geq v_{\text{top}}z$ crossings. If we remove all solid bottom edges of B from T and instead use all solid top edges and all solid vertical edges of B, we get a spanning tree T' which does not have more crossings than T. So we may suppose that in every box of G_F all the solid top edges or all the solid bottom edges are in T. We distinguish two cases.

Case 1: There are two consecutive boxes in a variable gadget such that from both boxes all the solid top edges (all the solid bottom edges) are in T. Then there are at least z crossings in T between the solid top edges in the two boxes and the dotted edges that connect these two boxes. Thus $\phi(G_F) \geq z$.

Case 2: There are no two consecutive boxes in a variable gadget such that from both boxes all the solid top edges (all the solid bottom edges) are in T. Then according to Figure 3 we derive a truth assignment β from the structure of T inside every variable gadget. Since F does not have a satisfying truth assignment there is a clause C in F which is false under β. Consider the clause gadget associated with C. This clause gadget must be connected in T with the rest of G_F which leads to at least z crossings. Thus again $\phi(G_F) \geq z$. □

Instead of searching for a spanning tree with a minimum number of crossings one could also ask for a crossing-free spanning forest of a geometric graph G with the minimum number $\phi'(G)$ of trees. But using a similar reduction as the one sketched above we obtain the following theorem.

Theorem 3 *It is NP-hard to approximate $\phi'(G)$ within a factor of $n^{1-\varepsilon}$ for any $\varepsilon > 0$.*

3 Fixed-parameter tractability

We first show that deciding whether or not a topological graph G has a crossing-free spanning tree is fixed-parameter tractable where we use the total number of crossings k as fixed parameter. We set $E_X = \bigcup X$. Thus E_X contains exactly those edges that participate in a crossing. Note that $|E_X| \leq 2k$ and observe that G has a crossing-free spanning tree iff G has a crossing-free connected spanning subgraph. Thus it is sufficient to examine all maximal crossing-free subgraphs of G and check whether one of them is connected. To this end we proceed as follows:

1. Remove all edges in E_X from G. The resulting graph $G' = (V, E')$ is crossing-free.

2. For all maximal crossing-free subsets $H \subseteq E_X$ check whether the graph $G' \cup H = (V, E' \cup H)$ is connected.

Note that in the second step we can actually shrink each connected component of G' to a single vertex. If G' has more than $k+1$ connected components, adding the at most k edges in H cannot make G' connected. Therefore we only have to consider a graph with $O(k)$ vertices.

It can easily be shown by induction on k that there are at most 2^k maximal crossing-free subsets H. Since detecting all the crossings in G can be done in $O(k + m \log m)$ time and generating and checking a subset H can be done in $O(k)$ time, we have the following theorem.

Theorem 4 *We can decide in $O(m \log m + k2^k)$ time whether a topological graph has a crossing-free spanning tree.*

Checking not only the crossing-free subsets of E_X but all subsets we obtain an algorithm that finds a spanning tree with the minimum weighted number of crossings.

Theorem 5 *Given a topological graph we can compute in $O(m \log m + k4^k)$ time a spanning tree with the minimum weighted number of crossings.*

In the remainder of this section we sketch how to speed up the decision algorithm in the special case that the crossings in X are pairwise disjoint.

Suppose G' has at most $k+1$ connected components and we have shrunken each to a single vertex. Our new algorithm distinguishes two cases. Let $\alpha = \frac{4}{5}$ and r be the number of connected components of G'.

Case 1: $r \leq \alpha k$. If G has a connected crossing-free spanning subgraph at all, then there are at least $2^{(1-\alpha)k}$ crossing-free subsets $H \subseteq E_X$ such that $G' \cup H$ is connected. This can be seen as follows: Suppose there is a crossing-free subset $F \subseteq E_X$ that makes G' connected. Then we can choose such an F with $|F| \leq r - 1 < \alpha k$. Let $X_F = \{c \in X \mid c \cap F = \emptyset\}$. Since the elements of X are pairwise disjoint we have $|X_F| > (1 - \alpha)k$. If we select one edge from each crossing in X_F and add these edges to F the resulting set of edges is still crossing-free. There are at least $2^{(1-\alpha)k}$ possible ways to select edges. Thus there are at least $2^{(1-\alpha)k}$ crossing-free subsets $H \subseteq E_X$ such that $G' \cup H$ is connected.

This suggests the following randomized algorithm: From each element of X we randomly select one edge and check if the resulting crossing-free graph is connected. If the given geometric graph G has a crossing-free connected spanning subgraph, the probability of success is at least $2^{(1-\alpha)k}/2^k = 2^{-\alpha k}$. Thus $O(2^{\alpha k})$ iterations suffice with high probability and the expected running time is $O(m \log m + k2^{\alpha k})$ with high probability.

Case 2: $r > \alpha k$. We define the degree of a connected component C of G' as the number of edges in E_X that are incident to a vertex that belongs to C.

We observe that for each component of degree one there is exactly one edge in E_X to connect this component to the rest of G'. Thus this edge must be selected for any connected spanning subgraph of G.

Furthermore we can assume that for every crossing $c \in X$ none of the two edges in c connects vertices in the same component of G', since such an edge could be deleted which would also make the crossing c vanish.

Next we argue that if $r > \alpha k$ there is always at least one component of G' that has degree at most 4. For if all components had degree at least 5, we would get $4k \geq 2|E_X| \geq 5r > 5\alpha k = 4k$.

Thus our algorithm selects a component C of G' such that the degree d of C is at most 4. The key observation is that one of the d edges in E_X incident to a vertex of C must be chosen to connect C to the rest of G'. This means that we do not have to try all 2^d subsets of those d edges but only $2^d - 1$. This observation can be made more precise by a detailed case analysis which we omit in this abstract. The case analysis yields the following recurrence for the running time $T(k)$ of our algorithm:

$$T(k) \leq 15T(k-4) + O(1)$$

This solves to $T(k) \in O(15^{k/4})$. To summerize we state the following theorem.

Theorem 6 *There is a Monte Carlo algorithm with one sided error and expected running time in $O(m \log m + k1.968^k)$ which decides whether a given geometric graph with pairwise disjoint crossings has a crossing-free spanning tree.*

4 A Mixed-Integer Linear Program

In this section we give a MIP formulation for the problem of computing a connected spanning subgraph G' with the minimum weigted number of crossings for a given topological graph $G(V,E)$. We introduce a variable $x_{ee'}$ that is designed to punish a crossing between the edges e and e'.

Our objective function is

$$\min \sum_{\{e,e'\} \in X} \mu_{ee'}\, x_{ee'}. \qquad (1)$$

We have the following constraints. We introduce a variable y_e for each edge $e \in E$ that is 1 iff e will be part of G'. For each pair $\{e,e'\} \in X$ we require:

$$x_{ee'} \geq 0 \quad \text{and} \quad x_{ee'} \geq y_e + y_{e'} - 1 \qquad (2)$$

Given our objective function (1) and the fact that y_e will be forced to lie in $\{0,1\}$, Constraint (2) is equivalent to $x_{ee'} = \min\{y_e, y_{e'}\}$. In other words if we manage to make sure that the graph $G' = (V,E')$ with $E' = \{e \in E \mid y_e = 1\}$ is connected, then the variables of type $x_{ee'}$ in fact count the number of crossings in G'.

We model connectivity by fixing an arbitrary vertex $s \in V$ as sink and then introducing flow between s and each other vertex $t \in V \setminus \{s\}$. The flow is modeled by a 0–1 variable f_e^t for each edge $e \in E$. First we make sure that for each choice of t, the source s and the sink t have exactly one edge with flow:

$$\sum_{e \text{ incident to } s} f_e^t = \sum_{e \text{ incident to } t} f_e^t = 1 \qquad (3)$$

Note that if a graph contains an s–t path, then that graph also contains an s–t path that visits each vertex at most once. So we simply ensure that each vertex $v \in V \setminus \{s,t\}$ has either zero or two incident edges with flow. To do this we need an auxiliary 0–1 variable h_v^t for each v. Now we can model our special kind of flow conservation in each vertex $v \in V \setminus \{s,t\}$.

$$\sum_{e \text{ incident to } v} f_e^t = 2h_v^t \qquad (4)$$

Finally, for each edge $e \in E$ we lower-bound the "global" decision variable y_e (that decides whether e goes into the spanning subgraph G') by the "local" flow f_e^t (that goes through e from s to t):

$$y_e \geq f_e^t \qquad (5)$$

Given our objective function and Constraint (2), this is equivalent to setting $y_e = \max\{f_e^t \mid t \in V \setminus \{s\}\}$. This completes our MIP formulation. It consists of $O(nm + k)$ variables and constraints.

5 Conclusion

We have shown that attacking the problem of finding a spanning tree with few crossings in a given topological graph by approximation algorithms is probably not a good idea, even though it could be an interesting theoretical problem to find approximation algorithms with approximation factor in $o(k)$.

On the other hand we have presented a fixed-parameter algorithm where we used the total number of crossings as fixed parameter. There might be other quantities associated with a geometric graph that could be used as fixed parameter. We are currently investigating this.

Finally we have given a MIP formulation, which we plan to implement in order to evaluate several heuristics.

References

[1] K. Jansen and G. J. Woeginger. The complexity of detecting crossingfree configurations in the plane. *BIT*, 33:580–595, 1993.

[2] J. Kratochvil, A. Lubiw, and J. Nesetril. Noncrossing subgraphs in topological layouts. *SIAM J. Disc. Math.*, 4(2):223–244, 1991.

[3] D. Lichtenstein. Planar formulae and their uses. *SIAM J. Computing*, 11:329–343, 1982.

Bi-Chromatic Minimum Spanning Trees

Magdalene Grantson Henk Meijer David Rappaport[*]

Abstract

Let G be a set of disjoint bi-chromatic straight line segments and H be a set of red and blue points in the plane, no three points are collinear. We give tight upper bounds on the maximum degree of a node in the color conforming minimum weight spanning tree (MST) formed by G and H. We also consider bounds on the total length of the edges of 1) the planar MST and the unrestricted MST, 2) the greedy planar spanning tree and the unrestricted MST, 3) the greedy planar spanning tree and the planar MST.

1 Introduction

Let G be a set of disjoint bi-chromatic straight line segments, and H a set of red and blue points in the plane, no three points are collinear. We obtain a spanning tree T of G (resp. H) by finding a set of $|G|-1$ (resp. $|H|-1$) edges, which connect vertices of different colors ("color conforming") and form an acyclic connected component. (For the spanning tree of G, input edges and non-input edges alternate, non-input edges may intersect, but are not allowed to intersect input edges.) If T must not contain intersections we call it a *planar spanning tree*,[1] otherwise we simply call T an (unrestricted) *spanning tree*.

A color conforming minimum weight spanning tree (MST) of G or H is a spanning tree of minimum total length of the added edges. Variants of this problem include finding the MST of a set of (mono-chromatic) points or line segments in the plane [2, 3, 1]. For these variants a greedy algorithm like Kruskal's [8] is known to yield the optimal solution [4, 3]. In [2, 3, 9] it is shown that in these cases the maximum degree of a node in the MST is bounded by five and seven, respectively.

Little is known about the MSTs of G or H. [6] gives a survey on geometry graphs of H. Recently, it was shown that a color conforming spanning tree of G or H is always obtainable [5]. It was also shown by illustration that the MST of a given G may contain intersections if one uses a greedy algorithm like Kruskal's [4]. (Kruskal's algorithm adds edges in increasing length order, and discards edges creating a cycle in the graph built so far [8].) We show by illustration that Kruskal's algorithm may also introduce intersections, when H is given as input. Modifying Kruskal's algorithm to check for such intersections and eliminating them, leads to a greedy algorithm which we will refer to as greedy planar algorithm. See [1, 7] for different flavors on non-crossing spanning trees.

$\sup_H \left[\frac{P_{gp}}{P_{op}}\right] \geq 2$	$\sup_G \left[\frac{L_{gp}}{L_{op}}\right] = \infty$	$\sup_G \left[\frac{L_{gp}-L_s}{L_{op}-L_s}\right] \geq 2$
$\sup_H \left[\frac{P_{gp}}{P_{uo}}\right] \geq 2$	$\sup_G \left[\frac{L_{gp}}{L_{uo}}\right] = \infty$	$\sup_G \left[\frac{L_{gp}-L_s}{L_{uo}-L_s}\right] \geq 2$
$\sup_H \left[\frac{P_{op}}{P_{uo}}\right] \geq 2$	$\sup_G \left[\frac{L_{op}}{L_{uo}}\right] = \infty$	$\sup_G \left[\frac{L_{op}-L_s}{L_{uo}-L_s}\right] \geq \frac{3}{2}$

Table 1: An overview of some of our results.

Definition 1 *Given a set G (resp. H) as defined above, we denote by: L_{s} the total length of the input straight line edges/segments, L_{ou} (resp. P_{ou}) the total length of the edges of the unrestricted MST, L_{op} (resp. P_{op}) the total length of the edges of the MST and L_{gp} (resp. P_{gp}) the total length of the edges of the greedy planar spanning tree.*

Summary of our results:
1) We show a bound for the maximum node degree in a color conforming MST.
2) We show the given bounds in Table 1.

2 Maximum Node Degree

Lemma 1 *The maximum degree of a node in a color conforming MST of a set G of n bi-chromatic disjoint line segments is not upper bounded by a fixed number, but only by the size of the input.*

Proof. Trivially the maximum degree of a node of the MST of a set of bi-chromatic disjoint line segments is bounded by n, because every vertex of one color can be connected to at most n vertices of the other color. This upper bound is tight, because an example achieving this bound is shown in Figure 1. For the set G of line segments in Figure 1, suppose the vertex v is red in color and the vertices $\{a, b, c, d, e, f, w\}$ are all blue in color and are at a distance r_1 from v. We refer to these vertices as lying on an inner circle C_i

[*]Dept. of Computer Science, Queens University, Kingston, Ontario, Canada {grantson,devor,henk}@cs.queensu.ca

[1]Note that the MST of the set of points or line segments in the plane can not contain intersections [3, 5]. For bi-chromatic straight line segments, however, intersections may occur.

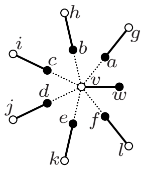


Figure 1: Any spanning tree where the vertex v has a smaller degree than the number of line segments does not have minimum total length.

of radius r_1 with v as its center. Suppose the vertices $\{g,h,i,j,k,l\}$ are red in color and are each at a distance $r_2 = 2r_1$ from v. We refer to these vertices as lying on an outer circle C_o of radius $r_2 = 2r_1$ s.t. v is its center. We make the following observations:
1) There are exactly $n-1$ color conforming edges, each of which has length r_1, namely $\{(v,a),(v,b),(v,c),(v,d),(v,e),(v,f)\}$. Selecting these $n-1$ edges gives a spanning tree.
2) All red vertices visible from an arbitrary blue vertex on the inner circle are $r_1+\epsilon$ away from this vertex, with the exception of v, which is r_1 away.
3) All blue vertices visible from an arbitrary red vertex on the outer circle are $r_1+\epsilon$ away from this vertex.
4) All vertices on the inner circle (outer circle) have the same color, hence it is impossible for an edge to be placed between any pair, since the color conforming characteristics would be violated if we did so.

From the above listed observations, selecting any subset of $n-1$ edges other than the edges from v to the vertices on the inner circle must include at least one edge with a length greater than r_1. Therefore no such subset of edges can be a minimum weight spanning tree. Consequently, the spanning tree formed by the above edges in Figure 1 is of minimum length. □

Lemma 2 *The maximum degree of a node in a color conforming MST of a set of n red and n blue points is not upper bounded by a fixed number, but only by n.*

Proof. Let the set H contain all the vertices in Figure 1, but none of the line segments. Using similar arguments as in the proof of Lemma 1, we observe that there are exactly $2n-1$ color conforming edges having length r_1. Selecting these $2n-1$ edges gives a MST, since all other edges apart from those has length $r_1+\epsilon$, $\epsilon > 0$. □

3 Bounds on Variants of the MST Problem

In [5] it was shown by illustration that the MST of bichromatic line segments may introduce intersections using a greedy algorithm like Kruskal's [4]. We show by illustration in Figure 2 (second diagram) that such intersections may occur when given a set of red and blue points in the plane. Such intersections can however be avoided by modifying, for example, Kruskal's greedy algorithm, so that at each step we rather add the non-crossing edge with the least weight, which

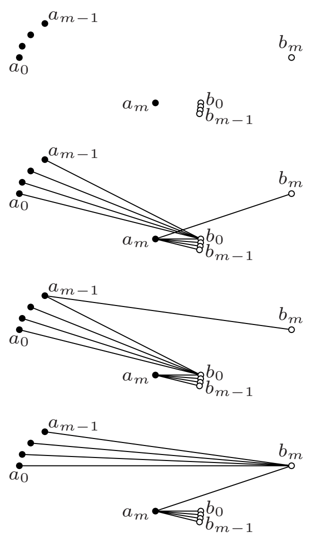


Figure 2: A set of bi-chromatic points for which the greedy spanning tree is by a factor of 2 worse than the planar MST and the unrestricted MST (top: set of points; second: unrestricted MST; third: planar MST; bottom: greedy planar spanning tree).

does not introduce cycles. Examples in this section clearly show that the modified algorithm may not give the optimal solution. The question then is whether there is a bound on the greedy planar solution in the worst case. We investigate such questions and more. Our results are collected in Table 1.

Observation 1 *If the solution may contain edge intersections, then a greedy approach (e.g. Kruskal's algorithm) always yields the optimal solution.*

Given is a graph $G = (V,E)$ (resp. $H = (V,E)$), such that V is the set of bi-chromatic line segment endpoints (resp. red and blue points) and E is the set of lines of sight between red and blue points. The proof that Kruskal's algorithm finds a minimum spanning tree of G (resp. H) follows from the proof [4] that Kruskal's algorithm finds a minimum spanning tree of any weighted, connected, undirected graph.

3.1 Input is a Set of Red and Blue Points

Theorem 3 *Let H be any set of red and blue points in the plane. Then $\sup_H\left[\frac{P_{gp}}{P_{op}}\right] \geq 2$ and $\sup_H\left[\frac{P_{gp}}{P_{ou}}\right] \geq 2$.*

Proof. Choose $m \in \mathbb{N}$, $m \geq 1$, and let $\phi = \arctan\frac{1}{m^2}$. Then define the points (see Figure 2):

red	blue
$a_m = (0,0)$,	$b_m = (m^2,1)$,
$a_k = \left(\frac{k^2}{m^2} - m^2, 1 + \frac{k}{m}\right)$,	$b_k = \left(\cos(\frac{k\phi}{m}), -\sin(\frac{k\phi}{m})\right)$.

$\forall k; 0 \leq k < m$. To show the ratios, let $m \to \infty$. Then

$$\lim_{m\to\infty} \frac{P_{gp}}{P_{op}} \geq \lim_{m\to\infty} \frac{6m^2+m}{3m^2+6m} = 2 \quad \text{and}$$
$$\lim_{m\to\infty} \frac{P_{gp}}{P_{ou}} \geq \lim_{m\to\infty} \frac{6m^2+m}{3m^2+4m} = 2. \qquad □$$

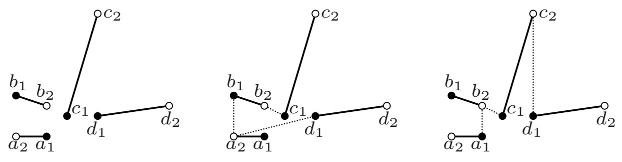


Figure 3: Left: set of bi-chromatic line segments; middle: planar MST; right: greedy planar spanning tree.

Theorem 4 *Let H be any set of red and blue points in the plane. Then* $\sup_H \left[\frac{P_{op}}{P_{ou}}\right] \geq \frac{3}{2}$.

Proof. We consider the set of points

red			blue		
a_{11}	$=$	$(0,0)$,	a_{12}	$=$	$(\epsilon, 0)$,
b_{11}	$=$	$(-1,\epsilon)$,	b_{12}	$=$	$(1,\epsilon)$.

Then

$$\lim_{\epsilon\to 0}\frac{P_{op}}{P_{ou}} = \lim_{\epsilon\to 0}\frac{\epsilon+\sqrt{1+\epsilon^2}+2}{\epsilon+\sqrt{1+\epsilon^2}+\sqrt{\epsilon^2+(1+\epsilon)^2}} = \frac{3}{2}. \quad \square$$

3.2 Input is a Set of Bi-chromatic Line Edges

Theorem 5 *Let G be any set of bi-chromatic straight line segments in the plane. Then* $\sup_G \left[\frac{L_{gp}-L_s}{L_{op}-L_s}\right] = \infty$ *and* $\sup_G \left[\frac{L_{gp}-L_s}{L_{ou}-L_s}\right] = \infty$.

Proof. We consider the set of bi-chromatic line segments in Figure 3 left: $a_1 = (-2,-2)$, $a_2 = (-5,-2)$, $b_1 = (-5,2)$, $b_2 = (-2,1)$, $c_1 = (0,0)$, $c_2 = (3,x)$, $d_1 = (3,0)$, and $d_2 = (x,1)$ with $x \geq 10$. The planar MST consists of the edges (a_2,b_1), (b_2,c_1), and (a_2,d_1) and has a total length of ≈ 13.98, independent of the value of x (see Figure 3 middle).

The modified Kruskal algorithm, however, considers the edge (a_1,b_2) before the edge (a_2,d_1). As a consequence it finds the spanning tree shown in Figure 3 right, which consists of the edges (a_1,b_2), (c_1,b_2), and (d_1,c_2) and thus has a total length of $3+\sqrt{3}+x > 14.73$ for $x \geq 10$. Therefore we have $\frac{L_{gp}-L_s}{L_{op}-L_s} = \frac{3+\sqrt{3}+x}{4+\sqrt{3}+\sqrt{68}}$, which goes to infinity as x goes to infinity. Trivially it follows that $\sup_G \left[\frac{L_{gp}}{L_{ou}}\right] = \infty$, since $L_{op} \geq L_{ou}$. $\square$

Theorem 6 *Let G be any set of bi-chromatic straight line segments in the plane. Then* $\sup_G \left[\frac{L_{op}-L_s}{L_{ou}-L_s}\right] = \infty$.

Proof. We consider the set of bi-chromatic line segments in Figure 4 left. It is $a_1 = (4,1)$, $a_2 = (x,1)$, $b_1 = (x,x)$, $b_2 = (1,3)$, $c_1 = (0,x)$, $c_2 = (0,5)$, $d_1 = (-x,0)$, $d_2 = (-4,0)$, $e_1 = (-x,-x)$, $e_2 = (-1,-2)$, $f_1 = (2,-6)$, $f_2 = (2,-x)$ with $x \geq 15$. The MST consists of the edges (b_2,a_1), (e_2,f_1), (e_2,a_1), (d_2,a_1) and (c_2,f_1) and has a total length of ≈ 33.67, independent of the value of x (see Figure 4 middle). The planar MST, however, consist of the edges (b_2,a_1), (e_2,f_1), (e_2,a_1), (d_2,a_1) and (c_1,b_2), since any other connection leads to an intersection (see Figure 4 right). As a consequence the planar MST has a total length ≥ 34.54 for $x \geq 15$. The ratio of the two total lengths goes to infinity as x goes to infinity. $\square$

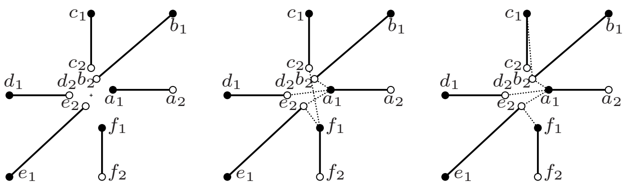


Figure 4: Left: set of bi-chromatic line segments; middle: unrestricted MST; right: planar MST.

Theorem 7 *Let G be any set of bi-chromatic straight line segments in the plane. Then,* $\sup_G \left[\frac{L_{gp}}{L_{ou}}\right] \geq 2$ *and* $\sup_G \left[\frac{L_{op}}{L_{ou}}\right] \geq 2$.

Proof. Choose $m \in \mathbb{N}$, $m \geq 3$, $s \in \mathbb{R}$, $s > 3$, and $r \in [\frac{3}{4}, 1)$ and let $t(k) = \sqrt{1-(\frac{rk}{m})^2}$. Then define the line segments (see Figure 5):

red	blue
$a_{11} = \left(\frac{1}{3m}, 0\right)$,	$a_{12} = (s+1, -1)$,
$a_{21} = \left(-\frac{1}{3m}, 0\right)$,	$a_{22} = (-s-1, 1)$,
$b_{k1} = \left(\frac{rk}{m}, t(k)\right)$,	$b_{k2} = \left(\frac{rk}{m}, s+t(k)\right)$,
$c_{k1} = \left(-\frac{rk}{m}, -1-t(k)\right)$,	$c_{k2} = \left(-\frac{rk}{m}, -t(k)\right)$.

$\forall k; 1 \leq k \leq m$. To show the above ratios, we let $m \to \infty$ and $s \to \infty$. To simplify the double limit $m \to \infty$ and $s \to \infty$, choose $s = m^2$. This yields

$$\lim_{m\to\infty}\frac{L_{gp}}{L_{ou}} = \lim_{m\to\infty}\frac{L_{op}}{L_{ou}} \geq \lim_{m\to\infty}\frac{2m^3}{m^3+13m^2} = 2. \quad \square$$

Theorem 8 *Let G be any set of bi-chromatic straight line segments in the plane. Then* $\sup_G \left[\frac{L_{gp}}{L_{op}}\right] \geq 2$.

Proof. Choose $m \in \mathbb{N}$, $m \geq 3$, $s \in \mathbb{R}$, $s \geq 6$, and let $t(k) = \sqrt{5-(\frac{k}{m})^2}$. Then define the line segments

red	blue
$a_{11} = (1, 1)$,	$a_{12} = (1+s, 1)$,
$a_{21} = (0, 3)$,	$a_{22} = (-s, 3)$,
$b_{11} = (2, 0)$,	$b_{12} = (0, 0)$,
$b_{21} = (-3, 2)$,	$b_{22} = (-1, 2)$,
$c_{k1} = \left(\frac{k}{m}, t(k)\right)$,	$c_{k2} = \left(\frac{k}{m}, s+t(k)\right)$,

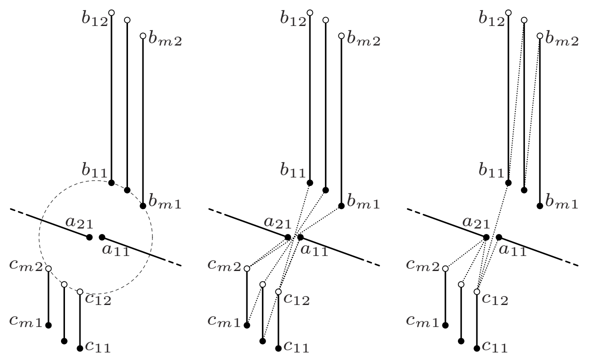


Figure 5: Left: set of bi-chromatic line segments; middle: unrestricted MST; right: planar MST.

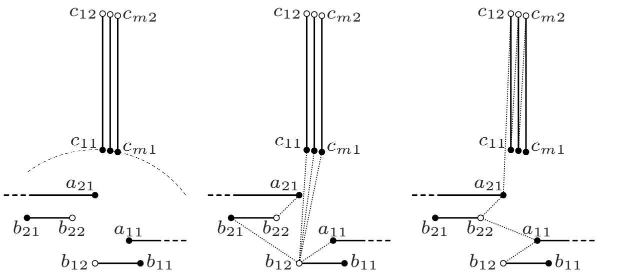


Figure 6: Left: set of bi-chromatic line segments; middle: planar MST; right: greedy planar spanning tree.

$\forall k; 1 \leq k \leq m$. To show the above ratios, we let $m \to \infty$ and $s \to \infty$. To simplify the double limit $m \to \infty$ and $s \to \infty$, choose $s = m^2$. This yields

$$\lim_{m\to\infty} \frac{L_{gp}}{L_{op}} = \lim_{m\to\infty} \frac{L_{gp}}{L_{op}} \geq \lim_{m\to\infty} \frac{2m^3 + 2m^2}{m^3 + 2m^2 + 40} = 2. \qquad \Box$$

3.3 Lower Bounds

Trivially the lower bounds of all the ratios in Table 1 is 1, because the numerator in each ratio is never less than the denominator. Moreover, instances, where the numerator in each ratio equal to the denominator, can be constructed.

3.4 Upper Bounds

Lemma 9 *The upper bound of the ratios* $\frac{P_{gp}}{P_{op}}$, $\frac{P_{gp}}{P_{ou}}$, $\frac{P_{op}}{P_{ou}}$, $\frac{L_{gp}}{L_{op}}$, $\frac{L_{gp}}{L_{ou}}$, *and* $\frac{L_{op}}{L_{ou}}$ *is* n.

Proof. Let d denote the length of the diagonal of the box bounding a given set G of bi-chromatic line segments or H of red and blue points in the plane. Trivially, $L_{ou} \geq d$ and $P_{ou} \geq d$. The total length of edges T_G of any tree of G is $\leq (2n-1)d$. Similarly, the total length of edges T_H of any tree of H is $\leq (n-1)d$ (see Figure 7). Hence we have $\frac{P_{gp}}{P_{op}} \leq n$, $\frac{P_{gp}}{P_{ou}} \leq n$, $\frac{P_{op}}{P_{ou}} \leq n$, $\frac{L_{gp}}{L_{op}} \leq n$, $\frac{L_{gp}}{L_{ou}} \leq n$, and $\frac{L_{op}}{L_{ou}} \leq n$. $\Box$

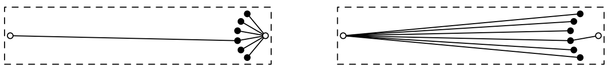


Figure 7: Set of bi-chromatic points for the upper bounds of the ratios.

4 Open Problem

An algorithm to determine a color conforming planar MST of a set of bi-chromatic line segments or red and blue points within a factor k still remains open. Moreover upper bounds other than n on the ratios: $\frac{P_{gp}}{P_{op}}$, $\frac{P_{gp}}{P_{ou}}$, $\frac{P_{op}}{P_{ou}}$, $\frac{L_{gp}}{L_{op}}$, $\frac{L_{gp}}{L_{ou}}$, $\frac{L_{op}}{L_{ou}}$, also remains open.

References

[1] O. Aichholzer, F Aurenhammer and F. Hurtado. Sequences of Spanning Trees and a Fixed Tree Theorem. *Computational Geometry: Theory and Applications* 21(1-2):3–20. Elsevier, Amsterdam, Netherlands 2002.

[2] P. Bose and G. Toussaint. Growing a Tree from its Branches. *Proc. 1st Pacific Conf. on Computer Graphics and Applications (Pacific Graphics '93, Seoul, Korea)*, 91–103. World Scientific Publishing Co., Hackensack, NJ, USA 1993.

[3] P. Bose and G. Toussaint. Growing a Tree from its Branches. *Journal of Algorithms* 19(1):86–103. Elsevier, Amsterdam, Netherlands 1995.

[4] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. Introduction to Algorithms (Second Edition). MIT Press and McGraw-Hill book company, USA 2002.

[5] F. Hurtado, M. Kano, D. Rappaport and C.D. Tòth. Encompassing Colored Crossing-Free Geometric Graphs. *16th Canadian Conference on Computational Geometry.* Montreal, Canada 2004.

[6] A. Kaneko and M. Kano. Discrete Geometry on Red and Blue Points in the Plane — A Survey. Discrete and Computational Geometry, *The Goodman-Pollack Festschrift, With contributions by numerous experts* 551–570. Springer, Heidelberg, Germany 2003.

[7] G. Károlyi, J. Pach, G. Tóth. Ramsey-Type results for geometric graphs. *Discrete and Computational Geometry* 18:247–255. 1997.

[8] J.B. Kruskal. On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. *Proc. of the American Mathematical Society,* 7:48–50. 1956.

[9] C. Monma and S. Suri. Transitions in Geometric Minimum Spanning Tree. *Bell Communications Research Technical Report.* Bell Communications Research, Inc., Livingstone, NJ, USA 1991.

Constructing Interference-Minimal Networks*

Marc Benkert Joachim Gudmundsson[†] Herman Haverkort[‡] Alexander Wolff[§]

Abstract

We consider the problem of producing interference-minimal graphs with additional properties such as connectivity, bounded stretch factor or bounded link diameter. We compute exact interference-minimal graphs and estimated interference-minimal graphs. The latter can be computed faster.

1 Introduction

A wireless communication network is commonly modelled by a graph $G = (V, E)$ that consists of a set of points V in the plane and a set of edges E. The nodes represent hosts, while edges represent communication links between nodes. We will assume that the links are undirected. This is a reasonable assumption as a one-way communication becomes unacceptably problematic. It should be noted however, that the algorithms presented in Section 3 can be extended to work for the directed model.

Burkhart et al. [2] propose a definition of interference in a wireless network, and describe two polynomial-time algorithms to construct an optimal spanning tree and a t-spanner in terms of this definition. Their running times are $\mathcal{O}(n^2 \log n)$ for spanning trees and $\mathcal{O}(n^4)$ for t-spanners, however they assume that all interferences are known in advance. We improve and extend these results. Particularly, our algorithms are output-sensitive.

We investigate three different graph classes and show how to compute interference-minimal graphs in each class. First we consider the problem of computing an interference-minimal connected graph, i.e. a spanning tree. The expected running time of our algorithm is $\mathcal{O}(nk(\log^2 k + \log n))$, where k is the minimum interference for which there is a spanning tree.

In addition to connectivity it is often desired that the resulting topology is a t-spanner for some given constant $t > 1$. Given two vertices u and v of a graph G, we will denote by $d_G(u, v)$ the length of the shortest path between u and v in G, and we use $|uv|$ to denote the Euclidean distance between u and v. A network G is said to be a t-spanner if for every pair of vertices u and v it holds that $d_G(u, v) \leq t \cdot |uv|$. We show that an interference-minimal t-spanner can be computed in $\mathcal{O}(n \log k_t({k_t}^2 + n \log n))$ expected time, where k_t is the minimum interference for which there exists a t-spanner.

The third graph class that we consider is the class of d-hop networks. A graph G is said to be a d-hop network if for every pair of vertices u and v, there is a path using at most d links between u and v. The expected running time of our algorithm for the d-hop network is $\mathcal{O}(n \log k_d({k_d}^2 + n \log n))$, where k_d is the minimum interference for which there exists a d-hop network.

It will turn out that our results are only faster than a method that uses *circular range searching* if $k = \mathcal{O}(n^{5/8})$. For dense graphs we argue in Section 4 that it is not realistic to assume exact data. Hosts on the perimeter of the transmission radius may or may not experience interference. Therefore we propose a model based on realistically accurate data, that is, we treat the sphere of an edge as a fuzzy object. This means that hosts lying close to the boundary of the transmission radius either may or may not be counted. Using this cost model we propose algorithms for the tree and the spanner with running times that beat the circular range searching method for all values of k and k_t. However, the running times depend on how fine the approximation should be.

2 Definitions

We start with the definition of interference. For a point $p \in \mathbb{R}^2$ and a real $r \geq 0$ let $D(p, r)$ be the closed disk that has center p and radius r. Given a communication network $G = (V, E)$ with $V \subset \mathbb{R}^2$, for an edge $\{u, v\} \in G$ and u, the range of u must be at least $|uv|$. Thus, all points in $D(u, |uv|)$ experience interferences. As we consider undirected edges, the following definition follows:

Definition 1 ([2]) *The* sphere *of an edge* $e = \{u, v\}$ *is* $S(e) := D(u, |uv|) \cup D(v, |uv|)$. *We define the* cost function Cov *(coverage) on the edge set* $\binom{V}{2}$ *by* $\mathrm{Cov}(e) := \left|V \cap S(e) \setminus \{u, v\}\right|$.

*This work was supported by grant WO 758/4-1 of the German Science Foundation (DFG) and the European Commission within FET Open Projects DELIS.

[†]Department of Mathematics and Computer Science, TU Eindhoven, h.j.gudmundsson@tue.nl

[‡]Department of Computer Science, University of Århus, herman@haverkort.net

[§]Fakultät für Informatik, Universität Karlsruhe, i11www.ira.uka.de/~awolff,~mbenkert

We define the interference $\operatorname{Int}(G)$ *of a graph* $G = (V, E)$ *by* $\operatorname{Int}(G) := \max_{e \in E} \operatorname{Cov}(e)$.

For given $m \geq 0$ let $G_m = (V, E_m)$ denote the graph where E_m includes all edges e with $\operatorname{Cov}(e) \leq m$.

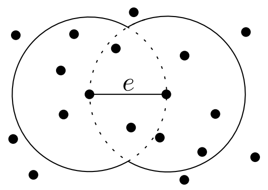


Figure 1: The sphere of e. Here $\operatorname{Cov}(e) = 9$.

3 Computing exact interference graphs

The main idea for computing interference-minimal trees, spanners and d-hop networks is the same. By combining exponential and binary search we determine the minimum graph that has the desired property $\mathcal{P}$, see Algorithm 1.

Algorithm 1: InterferenceNetwork($V, \mathcal{P}$)

```
$E_0$ = ComputeEdgeSet($V, 0$), $G_0 = (V, E_0)$
if FulfillsProperty($G_0, \mathcal{P}$) then
  return $G_0$
$m = 1/2$
repeat
  $m = 2 \cdot m$   {exponential search}
  $E_m$ = ComputeEdgeSet($V, m$), $G_m = (V, E_m)$
until FulfillsProperty($G_m, \mathcal{P}$)
$\ell = \lfloor m/2 \rfloor$, $r = m$
repeat
  $m = \ell + \lceil (r - \ell)/2 \rceil$   {binary search}
  $E_m$ = ComputeEdgeSet($V, m$), $G_m = (V, E_m)$
  if FulfillsProperty($G_m, \mathcal{P}$) then
    $r = m$
  else
    $\ell = m$
until $r = \ell + 1$
return $G_r$
```

The only non-trivial steps are the subroutines *ComputeEdgeSet* and *FulfillsProperty*. We first detail how to implement ComputeEdgeSet efficiently. We will use the *order-m Delaunay graph* and the *order-m Voronoi diagram*. An edge $\{u, v\}$ is an *order-m Delaunay edge* if there exists a circle through u and v that has at most m points of V inside.

Lemma 1 *All edges E_m are order-m Delaunay edges.*

Proof. Let e be an edge with $\operatorname{Cov}(e) \leq m$. By definition 1 the sphere $S(e)$ contains at most m points. Then, the disk that has e as diameter contains at most m points as it is contained in $S(e)$. □

We use the duality between Voronoi diagrams and Delaunay edges. We need precisely that $\{u, v\}$ is an order-m Delaunay edge if and only if there are two incident faces F_1 and F_2 in the order-$(m+1)$ Voronoi diagram such that u is in the set of points that determine F_1 and v is in the set of points that determine F_2. Details can be found in [4]. We need:

Theorem 2 ([4]) *For $m \leq n/2-2$, the order-$(m+1)$ Voronoi diagram can be computed in $\mathcal{O}(nm \log m + n \log n)$ expected time. There are $\mathcal{O}(nm)$ order-m Delaunay edges.*

Hence, by Lemma 1 we obtain the edge set E_m by testing all $\mathcal{O}(nm)$ order-m Delaunay edges. To do this we first compute in $\mathcal{O}(nm \log m + n \log n)$ total time for each point $p \in V$ the $m + 1$ nearest neighbors of p. Then, one edge test requires $O(m)$ time: we have to check how many of the $m + 1$ nearest neighbors of u lie in $D(u, |uv|)$ and in $D(u, |uv|) \cap D(v, |uv|)$ and analogously for v. With these numbers we can decide whether $\operatorname{Cov}(\{u, v\})$ is at most m. Thus, we have:

Theorem 3 *Given n points in the plane, the edge set E_m can be computed in $\mathcal{O}(nm^2 + n \log n)$ expected time for any $m \leq n/2 - 2$.*

Next, we describe how to implement *FulfillsProperty* for each of the three graph classes.

3.1 Spanning trees

To obtain an interference-minimal spanning tree we first run Algorithm 1 with the property connectivity. Let k be the minimum value for which G_k is connected. The exponential and binary search to determine k require $\mathcal{O}(\log k)$ steps in total. As the size of the edge set of each graph G_m is bounded by $\mathcal{O}(nk)$, the answer to *FulfillsProperty*($G_m, \mathcal{P}$) can by given in $\mathcal{O}(nk)$ time by running a breadth-first search on G_m. Hence, we can find the interference-minimal connected graph G_k in $\mathcal{O}(nk^2 \log k + n \log n \log k)$ expected time. If we only want to find a spanning tree $\mathcal{T}$ that minimizes $\operatorname{Int}(\mathcal{T})$, it is enough to run a breadth-first search on G_k. We can also run Prim's algorithm using the values $\operatorname{Cov}(e)$ as edge weights. This requires $\mathcal{O}(nk + n \log n)$ time.

Theorem 4 *We can find an interference-minimal spanning tree $\mathcal{T}$ that minimizes* $\operatorname{Int}(\mathcal{T})$ *and* $\sum_{e \in \mathcal{T}} \operatorname{Cov}(e)$ *in $\mathcal{O}(nk^2 \log k + n \log n \log k)$ expected time, if $k \leq n/2 - 2$.*

3.2 Spanners

Here, *FulfillsProperty*($G_m, \mathcal{P}$) must decide whether the current graph G_m is a t-spanner. We do this by computing shortest paths between all pairs of vertices and by checking for each pair $\{u, v\} \in \binom{V}{2}$ whether $d_{G_m}(u, v) \leq t \cdot |uv|$. Only if the answer is yes for each query, G_m is a t-spanner of V.

For the all-pairs-shortest-path computation we use an algorithm with expected running time $\mathcal{O}(n^2 \log n)$, described by Moffat and Takaoka in [5], i.e., *FulfillsProperty*$(G_m, \mathcal{P})$ can be implemented in time $\mathcal{O}(n^2 \log n)$ if $\mathcal{P}$ is the graph property "t-spanner". Combining this result with Theorem 3 gives:

Theorem 5 *Given $t > 1$, we can find an interference-minimal t-spanner in $\mathcal{O}(n \log k_t({k_t}^2 + n \log n))$ expected time, if $k_t \leq n/2 - 2$.*

3.3 d-hop networks

For testing the d-hop property in G_m we set the weights of all edges in E_m to 1 and run again the all-pairs-shortest-path algorithm. If every shortest path has weight at most d, then G_m is a d-hop network. We obtain time bounds analogous to those in Theorem 5.

Theorem 6 *Given an integer $d > 1$, we can find an interference-minimal d-hop network in $\mathcal{O}(n \log k_d({k_d}^2 + n \log n))$ exp. time, if $k_d \leq n/2 - 2$.*

4 Computing estimated interference graphs

We now assume that exact data is not realistic. Hosts on the perimeter of the transmission radius may or may not experience interference. Therefore we treat the sphere of an edge as a fuzzy object. This means that hosts lying very close to the boundary of the transmission radius either may or may not be counted. We define the estimated interference cost model:

Definition 2 *For fixed $\varepsilon > 0$ the* maximum estimated sphere *of an edge $e = \{u, v\}$ is defined as* $S_{\max}(e, \varepsilon) := D(u, (1 + \varepsilon) \cdot |uv|) \cup D(v, (1 + \varepsilon) \cdot |uv|)$, *the* minimum estimated sphere *of e is defined as* $S_{\min}(e, \varepsilon) := D(u, (1 - \varepsilon) \cdot |uv|) \cup D(v, (1 - \varepsilon) \cdot |uv|)$. *Following this notation we define* $\mathrm{Cov}_{\max}(e, \varepsilon)$ *and* $\mathrm{Cov}_{\min}(e, \varepsilon)$ *correspondingly. We say that $c \in \mathbb{N}$ is an* ε-valid interference estimation *of e if* $\mathrm{Cov}_{\min}(e, \varepsilon) \leq c \leq \mathrm{Cov}_{\max}(e, \varepsilon)$.

We will present algorithms for finding sparse networks with minimum estimated interference. Here, 'minimum' is meant with respect to a fixed assignment $\mathrm{Cov}(\cdot) : \binom{V}{2} \to \mathbb{N}$ of ε-valid estimations for all edges, which we will determine later. Let G_m now denote the graph that contains exactly the edges that have estimated interferences of at most m, w.r.t. $\mathrm{Cov}(\cdot)$. The algorithms follow Algorithm 1, only the subroutines *ComputeEdgeSet* and *FulfillsProperty* are implemented differently. The basic idea to speed them up is to perform all computations on a sparse graph G_m^- which closely approximates G_m. We obtain G_m^- using the *well-separated pair decomposition* (WSPD) of Callahan and Kosaraju. G_n^- will contain only one edge for each well-separated pair and G_m^- will simply be the corresponding subgraph of G_n^-. For details about the WSPD see [3]. We briefly recall the basic terminology. Two subsets $A, B \subset V$ are *well-separated* w.r.t. a separation constant s, if there are two closed disks D_A and D_B that have the same radius r and it holds that $A \subset D_A$, $B \subset D_B$ and the distance of D_A and D_B is at least sr. A WSPD of V is a sequence of subsets $\{A_1, B_1\}, \ldots, \{A_m, B_m\}$ such that each pair $\{A_i, B_i\}$ is well-separated and for any two points $u, v \in V$ there is exactly one pair $\{A_j, B_j\}$ with $u \in A_j$ and $v \in B_j$ or $v \in A_j$ and $u \in B_j$. Callahan and Kosaraju [3] showed that a WSPD of size $m = \mathcal{O}(s^2 n)$ can be computed in $\mathcal{O}(s^2 n + n \log n)$ time.

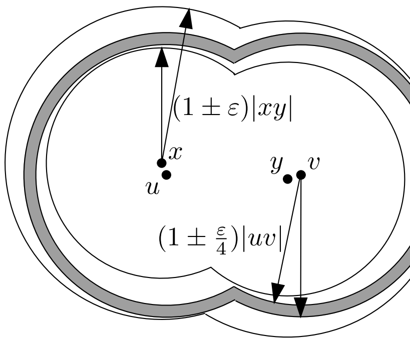

$(1 \pm \varepsilon)|xy|$
x
u
y
v
$(1 \pm \frac{\varepsilon}{4})|uv|$

Figure 2: Illustration for Lemma 7.

For an well-separated pair $\{A_i, B_i\}$ let E^i denote the set $E^i = \{\{u, v\} \mid u \in A_i, v \in B_i\}$. We obtain G_n^- by computing the WSPD and adding one edge of each E^i. We then compute an $(\varepsilon/4)$-valid interference estimation for the choosen edge of E^i. The next lemma shows that the resulting value is an ε-valid interference estimation for ell edges of E^i. This approach also yields the fixed assignment $\mathrm{Cov}(\cdot)$ of ε-valid interference estimations with respect to which we will compute the estimated interference-minimal graphs.

Lemma 7 *Let $\{A_i, B_i\}$ be a well-separated pair. Let $e_1 = \{u, v\}$ and $e_2 = \{x, y\}$ be two edges of E^i. It holds that an $(\varepsilon/4)$-valid interference estimation for e_1 is an ε-valid interference estimation for e_2, assuming $s \geq (8 + 5\varepsilon)/\varepsilon$.*

If we set the separation constant s to $\mathcal{O}(1/\varepsilon)$, the graph G_n^- has $\mathcal{O}(n/\varepsilon^2)$ edges, and so has G_m^-. The dilation of a graph G is the infimum of the set $\{t \mid G \text{ is a } t\text{-spanner}\}$. We can show that:

Theorem 8 *(i) G_m is connected if and only if G_m^- is connected,*

(ii) if G_m^- has dilation t then G_m has dilation in the range $[t/(1 + \varepsilon), t]$.

Next, we describe how to compute the valid estimations of edge costs efficiently. The idea is to use extended query ranges. For a given ε and a query range Q, the extended query range Q_ε is the set of points lying at L_∞-distance at most εw from Q, where w is the diameter of Q. In our application, the query range will be the union of two disks of radius r, thus we can easily set the values such that Q_ε is the set of points lying at L_∞-distance at most εr from Q.

We will perform ε-approximate counting queries. A simple modification of the BBD-tree by Arya and Mount [1] gives the following theorem:

Theorem 9 *Given a set V of n points in the plane, a constant-complexity query range Q and some $\varepsilon > 0$, ε-approximate range counting queries can be answered in $\mathcal{O}(1/\varepsilon + \log n)$ time using $\mathcal{O}(n \log n)$ preprocessing and $\mathcal{O}(n)$ space.*

Proposition 10 *Given a real value $\varepsilon > 0$ and a constant-complexity range Q, it holds that an ε-approximate range counting query returns an integer N such that $|\{V \cap Q\}| \leq N \leq |\{V \cap Q_\varepsilon\}|$.*

Given an edge e it is straight-forward to see that performing an ε-approximate counting query will give an ε-valid interference estimation for e.

In summary, the graph G_n^- can be computed in $\mathcal{O}(n/\varepsilon^2(1/\varepsilon + \log n))$ deterministic time. Note that we compute G_n^- in a preprocessing step and the graphs G_m^- needed in the exponential and binary search of the algorithms are obtained as subgraphs from G_n^- in $\mathcal{O}(n/\varepsilon^2)$ time. We now show how the sparse graphs G_m^- help to implement the subroutine *FulfillsProperty(G_m, V)* efficiently.

4.1 Spanning trees

According to Theorem 8 (i), we can perform the testing whether G_m is connected by simply testing G_m^-. A breadth-first search in G_m^- requires $\mathcal{O}(n/\varepsilon^2)$ time. Thus, we require $\mathcal{O}(n/\varepsilon^2 \log k)$ time in total for connectivity testing (exponential and binary search), where k is the minimum value for which there is a connected graph. This means that G_k can be computed in $\mathcal{O}(n/\varepsilon^2(1/\varepsilon + \log n))$ time. We then simply run a breadth-first search in G_k^- to obtain an estimated interference-minimal spanning tree.

Theorem 11 *We can find an estimated interference-minimal spanning tree $\mathcal{T}$ in $\mathcal{O}(n/\varepsilon^2(\log n + 1/\varepsilon))$ time.*

By running Prim's algorithm in G_k we could find an interference-minimal spanning tree $\mathcal{T}$ that also minimizes $\sum_{e \in \mathcal{T}} \mathrm{Cov}(e, \varepsilon)$. However, this could require $\Theta(n^2)$ time since G_k can have up to $\Theta(n^2)$ edges.

4.2 Spanners

Let $t^* > 0$ be the given constant for which we want to compute a t^*-spanner. To decide whether G_m has dilation at most t^* one needs to perform a shortest-path query for every pair of points in V. However, since we are only looking for an approximate solution we can use an approximation of t^*. We call t a (c_1, c_2)-*approximate stretch factor* of a graph with dilation t^*, if it holds that $t/c_1 \leq t^* \leq c_2 t$. We apply a result of Narasimhan and Smid [6] that says that a $(1+\varepsilon, 1+\delta)$-approximate stretch factor can be computed in $\mathcal{O}(n\delta^{-2}(\log n + T(n)))$ time for any $\delta > 0$, where $T(n)$ is the time required for a $(1+\varepsilon)$-approximate shortest path query. From Theorem 8 (ii) it follows that $d_{G_m}(u,v) \leq d_{G_m^-}(u,v) \leq (1+\varepsilon) \cdot d_{G_m}(u,v)$ for any two points $u, v \in V$. Thus, performing an exact shortest path query in G_m^- yields a $(1+\varepsilon)$-approximate shortest path for G_m. The running time of such a query is $T(n) = O(n\varepsilon^{-2} \log n)$ if we use Dijkstra's algorithm on the linear-size graph G_m^-. This means computing an approximate stretch factor t of G_m requires $\mathcal{O}(n^2\varepsilon^{-2}\delta^{-2} \log n)$ time in total. The subroutine *FulfillsProperty(G_m, V)* will answer yes if $t/(1+\varepsilon) \leq t^*$. We then know that the output is a graph G_{k_t} for which k_t is at most Int^*, the minimum estimated interference value for any t^*-spanner of V. Setting $\delta = \varepsilon$ and summing up the running times yields the following theorem.

Theorem 12 *Given two real numbers $t^* > 1$ and $\varepsilon > 0$, we can find a $(1+\varepsilon)t^*$-spanner in time $\mathcal{O}(n^2\varepsilon^{-4} \log n \log k_{t^*})$ with estimated interference value at most* Int^*.

References

[1] S. Arya and D. Mount. Approximate Range Searching. Computational Geometry: Theory & Applications, 17(3–4): 135–152, 2000.

[2] M. Burkhart, P. von Rickenbach, R. Wattenhofer and A. Zollinger. Does topology control reduce interference? Proc. 5th ACM International Symp. on Mobile Ad Hoc Networking and Computing (MOBIHOC), 2004.

[3] P. B. Callahan and S. R. Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. Journal of the ACM, 42:67–90, 1995.

[4] J. Gudmundsson, M. Hammar and M. van Kreveld. Higher order Delaunay triangulations. Computational Geometry: Theory & Applications, 23(1):85–98, 2002.

[5] A. Moffat and T. Takaoka An all pairs shortest path algorithm with expected time $\mathrm{O}(n^2 \log n)$. SIAM Journal on Comp., 16(6):1023–1031, 1987.

[6] G. Narasimhan and M. Smid. Approximating the Stretch Factor of Euclidean Graphs. SIAM Journal on Comp., 30(3):978–989, 2000.

A Note on Simultaneous Embedding of Planar Graphs (Abstract)[*]

Emilio Di Giacomo[†] Giuseppe Liotta[†]

1 Introduction

Let G_1 and G_2 be a pair of planar graphs such that $V(G_1) = V(G_2) = V$. A *simultaneous embedding* [6] $\Psi = (\Gamma_1, \Gamma_2)$ of G_1 and G_2 is a pair of crossing-free drawings Γ_1 and Γ_2 of G_1 and G_2, respectively, such that for every vertex $v \in V$ we have $\Gamma_1(v) = \Gamma_2(v)$. If every edge $e \in E(G_1) \cap E(G_2)$ is represented with the same simple open Jordan curve both in Γ_1 and in Γ_2 we say that Ψ is a *simultaneous embedding with fixed edges*. If the edges of G_1 and G_2 are represented with straight-line segments in Γ_1 and Γ_2 we say that Ψ is a *simultaneous geometric embedding*. The existence of simultaneous geometric embeddings for pairs of paths, cycles, and caterpillars is shown in [2], where also counter-examples for pairs of general planar graphs, pairs of outerplanar graphs, and triples of paths are presented.

Concerning the the computation of (non-geometric) simultaneous embeddings, Erten and Kobourov [6] presented an $\mathcal{O}(n)$-time algorithm to simultaneously embed any pair of planar graphs on the $\mathcal{O}(n^2) \times \mathcal{O}(n^2)$ grid with at most three bends per edge, where n is the number of vertices of G_1 and G_2. If the two graphs are trees then the number of bends per edge can be reduced to one. Furthermore, in [6] an $\mathcal{O}(n)$-time algorithm to compute a simultaneous embedding with fixed edges of a tree and a path on the $\mathcal{O}(n) \times \mathcal{O}(n^2)$ grid with no bends on the path-edges and at most one bend per edge on the tree-edges is described.

In this note we revisit the elegant technique of Erten and Kobourov [6] to present some new results on simultaneous embeddings with fixed edges. We prove that the pairs outerplanar graph - path and outerplanar graph - cycle admit a simultaneous embedding with fixed edges and at most one bend per edge. For the pair outerplanar graph - path, the edges of the path are straight-line segments. We also present some extensions of the results in [6] about simultaneous embeddings of planar graphs that are immediate consequence of existing literature. For reasons of space some proof are sketched or omitted.

[*]Research partially supported by MIUR, under Project "ALGO-NEXT(Algorithms for the Next Generation Internet and Web: Methodologies, Design and Experiments)".

[†]Dipartimento di Ingegneria Elettronica e dell'Informazione, Università degli Studi di Perugia, {digiacomo, liotta}@diei.unipg.it

2 Preliminaries

The algorithms to compute a simultaneous embedding of a pair of planar graphs and of a pair of trees presented in [6] are an elegant combination of the technique by Kaufmann and Wiese [7] for point-set embedding and of the simultaneous embedding strategy for two paths by Brass et al. [2]. In this section, we first recall the main ideas of Erten and Kobourov for simultaneous embedding of two planar graphs G_1 and G_2 and then make a couple of observations on how to combine these ideas with known literature in order to extend some of the results in [6].

Suppose first that both G_1 and G_2 are Hamiltonian. Let C_1 and C_2 be two Hamiltonian cycles of G_1 and G_2, respectively. For each cycle, one arbitrarily chosen edge (called *closing edge* in the following) is removed in order to obtain two Hamiltonian paths P_1 and P_2 for G_1 and G_2, respectively. Paths P_1 and P_2 are simultaneously embedded by the algorithm of Brass et al. [2]. Erten and Kobourov add the remaining edges of G_1 and of G_2 to the drawing by using the technique of Kaufmann and Wiese [7]. Namely, let δ be the maximum slope of a segment of the path defined by P_1. The closing edge for cycle C_1 is drawn as a polyline with two segments whose slopes are δ' and $-\delta'$, where $\delta' = \delta + \epsilon$ for an arbitrary small *epsilon* > 0. The remaining edges of G_1 are divided into edges that are inside C_1 and edges that are outside C_1. The edges that are inside (outside) C_1 are drawn inside (outside) C_1 as polylines each consisting of two segments having slopes δ' and $-\delta'$, respectively. Possible overlaps between segments corresponding to different edges can be removed by a simple rotation technique described in [7]. The drawing of G_2 is computed with an analogous procedure starting form the drawing of path P_2. Erten and Kobourov show that the overall time complexity of the procedure is $\mathcal{O}(n)$ where n is the number of vertices in G_1 and in G_2; also, if the bends may not be at integer grid points, the size of the grid is $\mathcal{O}(n^2) \times \mathcal{O}(n^2)$ ($\mathcal{O}(n^3) \times \mathcal{O}(n^3)$ else).

If G_1 and G_2 are not sub-Hamiltonian, then Erten and Kobourov use the $\mathcal{O}(n)$ algorithm by Chiba and Nishizeki [3] to augment the graphs in order to make them Hamiltonian. The augmentation is done by adding dummy edges and by splitting each edge with at most one dummy vertex. A simultaneous embedding of the augmented graphs can now be computed in linear time be the technique described above. After such an embedding is computed the dummy edges are removed and the dummy vertices are treated as bend points. As a result, every edge (u, v) that is split by a dummy vertex w ends up having at most three bends, one between u and w, one at w and one between w and v. Observe that the bend at w can be avoided if the two segments of (u, w) and (w, v) incident on w have the same slope. In [7] it is described how to rotate the segments incident on w so to avoid the third bend. However this rotation increases the area of the drawing, and Erten and Kobourov do not apply the rotation. As a result, they prove the existence of a simultaneous embedding of two planar graphs in an integer grid of size $\mathcal{O}(n^2) \times \mathcal{O}(n^2)$ and with at most three bends per edge (or $\mathcal{O}(n^3) \times \mathcal{O}(n^3)$ if bends are at integer grid points). In [5] it is showed a variant of the point-set embedding algorithm of Kaufmann and Wiese [7] that makes it possible to never have a third bend on the split edges and thus it does not require any rotation of the edges. A combination of the results in [5] and of the technique in [6] leads therefore to the following improvement of the result by Erten and Kobourov.

Theorem 1 *Let G_1 and G_2 be two planar graphs such that $V(G_1) = V(G_2) = V$. G_1 and G_2 can be simultaneously embedded in $\mathcal{O}(n)$ time, using at most two bend per edge, on an integer grid of size $\mathcal{O}(n^2) \times \mathcal{O}(n^2)$, where $n = |V|$.*

The approach of Erten and Kobourov is such that if the two graphs can be augmented without adding dummy vertices, then we have a simultaneous embedding with at most one bend per edge. In the special case of trees, they augment the graphs to become Hamiltonian without using dummy vertices. We recall that every sub-Hamiltonian graph has the property that it can be augmented with only edge addition to become Hamiltonian. Although it is $\mathcal{NP}$-hard to recognize the sub-Hamiltonian graphs, there are some families of graphs that are known to be sub-Hamiltonian and for which the edge augmentation can be found in time proportional to the number of the vertices. Among such families, we mention here outerplanar graphs and series-parallel graphs (see, e.g., [1, 4]). We can therefore extend Theorem 3 of [6] to families of graphs other than trees.

Theorem 2 *Let G_1 and G_2 be two graphs such that $V(G_1) = V(G_2) = V$ and G_i $(i = 1, 2)$ is either a series-parallel graph or an outerplanar graph. G_1 and G_2 can be simultaneously embedded in $\mathcal{O}(n)$ time, using at most one bend per edge, on an integer grid of size $\mathcal{O}(n^2) \times \mathcal{O}(n^2)$, where $n = |V|$.*

3 Simultaneous Embedding with Fixed Edges

In [6] the following result is proved.

Theorem 3 *[6] Let T be a tree and let P be a simple path such that $V(T) = V(P) = V$. T and P can be simultaneously embedded with fixed edges in $\mathcal{O}(n)$ time, using at most one bend for each edge of T and zero bends for each edge of P, on an integer grid of size $\mathcal{O}(n) \times \mathcal{O}(n^2)$, where $n = |V|$.*

The proof of Erten and Kobourov behind Theorem 3 exploits the technique described in Section 2. Namely, they present a linear-time recursive strategy for computing a Hamiltonian path P_T of the tree that contains all edges shared by P and T and then use such a Hamiltonian path and path P itself to compute a simultaneous embedding by the technique described Section 1. Since P and P_T are drawn with straight-line segments and the remaining edges of T have at most one bend, the theorem follows. It is worth remarking that the key idea in the proof of Erten and Kobourov for Theorem 3 is reducing the tree-path simultaneous embedding problem with fixed edges to the combinatorial question of finding an augmented Hamiltonian cycle in the tree with the additional constraint that the cycle must contain all edges of P. In this section we use the same approach of Erten and Kobourov to extend Theorem 3.

We need some more definitions. A k-pages book embedding $\phi(G)$ of a graph G is a crossing-free drawing of G such that the vertices of G are drawn as points along a straight line l called *spine*, and each edge is drawn as a simple Jordan curve on one among k half-planes, called *pages*, having l as common boundary. The minimum number of pages over all book embeddings of a graph G is called the *page number* of G. A graph has page number one if and only if it is outerplanar and that a graph has page number two if and only if it is sub-Hamiltonian [1].

Let $v_0, v_1, \ldots, v_{n-1}$ be the vertices of G in the order they are encountered along the spine from left to right. We say that $v_0, v_1, \ldots, v_{n-1}$ is the *linear ordering induced by* $\phi(G)$. An edge $e_1 = (v_i, v_j) \in E(G)$ is said to be *nested inside* another edge $e_2 = (u_h, v_l) \in E(G)$ in $\phi(G)$ if e_1 and e_2 are drawn on the same page and $h < i < j < l$. A r*-rainbow* is a set of r edges $e_0, e_1, \ldots, e_{r-1} \in E(G)$ such that e_{i+1} is nested inside e_i $(0 \leq i < r - 1)$. Edges e_0 and e_1 are called the *top edge* and the *second edge* of the r-rainbow, respectively.

In what follows we are interested in book embeddings on at most two pages. In this case the book embedding is a planar drawing and the two pages are the two half-planes defined by the straight line representing the spine. We assume that the spine is drawn as an horizontal straight line and we refer to the two pages as the *top page* and as the *bottom page*.

Also, when we consider a 1-page book embedding we shall assume that the only page is the top page. The next two lemmas use 1-page book embeddings to show some combinatorial properties of outerplanar graphs.

Lemma 4 *Let G be an outerplanar graph, let $\phi(G)$ be a 1-page book embedding of G, and let $v_0, v_1, \ldots, v_{n-1}$ the linear ordering induced by $\phi(G)$. Let $E' \subseteq E(G)$ be a set of disjoint edges and let $e^* = (v_i, v_j)$, $0 < i < j < n-1$ be an edge of E'. It is possible to add edges to G in such a way that the augmented graph G' is planar and contains a path π with the following properties: (i) π starts at v_i and ends at v_{j+1}; (ii) π contains e^*, all the edges of E' nested inside e^*, and all vertices v_l with $i < l < j+1$.*

Sketch of Proof. Let ϕ' be the 1-page book embedding obtained by deleting from $\phi(G)$ all edges of $E(G)$ that do not belong to E'. For each edge $e \in E'$, the *weight* of e is the maximum r such that there exists a r-rainbow in ϕ' having e as its top edge; if e is not the top edge of any rainbow, the weight of e is 0.

We prove the statement by induction on the weight w of e^*. If $w = 0$, i.e. no edge of E' is nested inside e^*, we proceed as illustrated in Figure 1: if there is no vertex between v_i and v_j along the spine, we choose $\pi = v_i, v_j, v_{j+1}$; otherwise, π is obtained by concatenating the path v_i, v_j with the path $v_j, v_{j-1}, \ldots, v_{i+1}$, and with edge (v_{i+1}, v_{j+1}). The edges of π that are not in $\phi(G)$ are augmenting edges. We draw all the augmenting edges on the bottom page. Let $\phi(G')$ be the augmented drawing and let G' be the corresponding augmented graph. We prove that $\phi(G')$ is a 2-page book embedding and therefore G' is planar. The edges in the top page do not cross each other because they are all edges of G and do not cross in $\phi(G)$. All edges in the bottom page connect pairs of consecutive vertices except, possibly, edge (v_{i+1}, v_{j+1}). It follows that there cannot be any crossing in the bottom page and hence G' is planar.

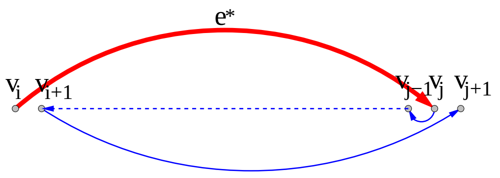


Figure 1: The base case of Lemma 4

Suppose now that $w = k \geq 1$ and that the statement is true for $w < k$. Let ϕ' be the 1-page book embedding obtained by deleting from $\phi(G)$ all edges of $E(G)$ that do not belong to E'. Let $e_0 = (v_{i_0}, v_{j_0}), e_1 = (v_{i_1}, v_{j_1}), \ldots e_h = (v_{i_{h-1}}, v_{j_{h-1}})$ $(i_0 < i_1 < \cdots < i_{h-1})$ be the second edges of the rainbows that have e as their top edge. The weight of each edge e_m $(0 \leq m \leq h-1)$ is at most $k-1$ and by inductive hypothesis there exists a path π_m from v_{i_m} to v_{j_m+1} that contains e_m, all edges of E' that are nested inside e_m, and all vertices between v_{i_m} and v_{j_m+1}. Also, since the edges of E' are disjoint we have that $i_0 < j_0 < i_1 < j_1 < \cdots < i_{h-1} < j_{h-1}$. Denote as $\overline{\pi_m}$ the path with the same edges as π_m that starts at v_{j_m+1} and ends at v_{i_m} (that is, $\overline{\pi_m}$ is the "reverse" of π_m). We choose π as depicted in Figure 2, i.e. $\pi = v_i, v_j, v_{j-1}, \ldots, \overline{\pi_{h-1}}, \ldots, \overline{\pi_{h-2}}, \ldots, \overline{\pi_0}, \ldots, v_{i+1}, v_{j+1}$, where the edges of π that are not in G are augmenting edges. Path π starts at v_i, ends at v_{j+1}, it contains e^*, and by induction it contains all vertices between v_i and v_{j+1} and all edges of E' nested inside e.

Let $\phi(G')$ be the augmented drawing and let G' be the corresponding augmented graph. We prove that $\phi(G')$ is a 2-page book embedding and therefore G' is planar. The edges in the top page do not cross each other because they are all edges of G and do not cross in $\phi(G)$. Let d_1 and d_2 be two edges in the bottom page both connecting two vertices that are non-consecutive along the spine. If d_1 and d_2 are both edges of a path π_m $(0 \leq m \leq h-1)$, then they do not cross by induction. If d_1 and d_2 are edges of two different paths π_{m_1} and π_{m_2} $(0 \leq m_1 < m_2 \leq h-1)$, then the vertices of π_{m_1} are before those of π_{m_2} and d_1 and d_2 do not cross. The only edge in the bottom page that connects vertices that are non-consecutive along the spine and that does not belong to $\cup_{m=0}^{h-1} \pi_m$ is (v_{i+1}, v_{j+1}). Let d_1 be the edge (v_{i+1}, v_{j+1}) and let d_2 be an edge of a path π_m $(0 \leq m \leq h-1)$; we have $v_{i+1} < v_{i_m} < v_{j_m} < v_{j+1}$, which implies that the two edges do not cross and hence G' is planar. $\square$

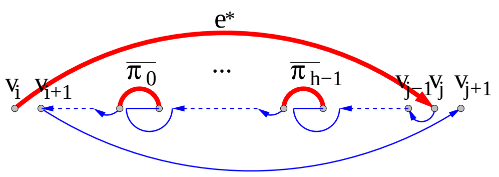


Figure 2: The inductive case of Lemma 4

Lemma 5 *Let G be an outerplanar graph and let $E' \subseteq E(G)$ be a set of disjoint edges. It is possible to add edges to G in such a way that the augmented graph G' is planar and has a Hamiltonian cycle containing all edges of E'.*

Sketch of Proof. Since G is outerplanar it admits a 1-page book embedding $\phi(G)$. Let $v_0, v_1, \ldots, v_{n-1}$ the linear ordering induced by $\phi(G)$. It can be proved [1] that one can always compute $\phi(G)$ such that v_0 and v_{n-1} are not connected by an edge of $E(G) \cap E(P)$. Let $e_0 = (v_{i_0}, v_{j_0}), e_1 = (v_{i_1}, v_{j_1}), \ldots e_{h-1} = (v_{i_{h-1}}, v_{j_{h-1}})$ $(0 < i_0 < i_1 < \cdots < i_{h-1} < n-1)$

be the edges of E' not nested inside any other edge of E'. Since the edges of E' are disjoint then $i_0 < j_0 < i_1 < j_1 < \cdots < i_{h-1} < j_{h-1}$. Also, by Lemma 4 for each edge e_m $(0 \leq m \leq h-1)$ there exists a path π_m from v_{i_m} to v_{j_m+1} that contains edge e_m, all edges of E' nested inside e_m, and all vertices between v_{i_m} and v_{j_m+1}. We choose a cycle $C = v_0, v_1, \ldots, \pi_0, \ldots, \pi_1, \ldots, \pi_{h-1}, v_{n-2}, v_{n-1}, v_0$, where the edges of C that are not in G are augmenting edges. By Lemma 4, C contains all vertices of G and all edges of E'. □

Lemma 6 *Let G be an outerplanar graph and let P be a simple path such that $V(G) = V(P) = V$. It is possible to add edges to G in such a way that the augmented graph G' is planar and has a Hamiltonian cycle containing all edges of $E(G) \cap E(P)$.*

Sketch of Proof. The subgraph of G induced by the edges of $E(G) \cap E(P)$ is a forest of paths; we denote by $\pi_0, \pi_1, \ldots \pi_{h-1}$ the paths of this forest. Let $\phi(G)$ be a 1-page book embedding of G and let $v_0, v_1, \ldots, v_{n-1}$ the linear ordering induced by $\phi(G)$. As in Lemma 5, we can assume that v_0 and v_{n-1} are not connected by an edge of $E(G) \cap E(P)$. Let H be the graph defined as follows. The vertices of H are all vertices of G except those having two edges of $E(G) \cap E(P)$ incident on them; for each path π_m with endvertices v_i and v_j, H has the edge $e_m = (v_i, v_j)$ $(0 \leq m \leq h-1)$.

Graph H is a set of disjoint edges and we can apply Lemma 5 to H with $E' = E(H)$. We obtain an augmented graph H' that is planar and has a Hamiltonian cycle D containing all edges of H (actually, cycle D coincides with H' since E' coincides with $E(H)$). The technique in the proof of Lemma 5 defines H' by computing a 2-page book embedding $\phi(H')$ of H'. $\phi(H')$ is computed by adding edges to a given 1-page book embedding $\phi(H)$ of H; we assume here that $\phi(H)$ is obtained from $\phi(G)$ by deleting the vertices and edges of G that are not in H and by adding in the top page the edges of H that replace each π_m $(0 \leq m \leq h-1)$. Let G' be the graph obtained by adding to G the augmenting edges of H' and let C be the cycle obtained by replacing each edge e_m of H in H' with the corresponding path π_m $(0 \leq m \leq h-1)$. Cycle C is a simple cycle of G' and by construction it contains all vertices of G and all edges of $E(G) \cap E(P)$.

Consider the drawing $\phi(G')$ of G' defined as follows. The vertices of G' are drawn as in $\phi(G)$; the edges of $E(G') \cap E(G)$ are drawn as in $\phi(G)$, i.e. they are drawn on the top page; the edges of $E(G') \cap E(H')$ are drawn as in $\phi(H')$, i.e. they are drawn on the bottom page. We have that the edges above the spine do not cross since they do not cross in $\phi(G)$ and the edges below the spine do not cross since they do not cross in $\phi(H')$. It follows that $\phi(G')$ is a 2-page book embedding and therefore G' is planar. □

Theorem 7 *Let G be an outerplanar graph and let P be a simple path such that $V(G) = V(P) = V$. G and P can be simultaneously embedded with fixed edges in $\mathcal{O}(n)$ time, using at most one bend for each edge of G and zero bends for each edge of P, on an integer grid of size $\mathcal{O}(n) \times \mathcal{O}(n^2)$, where $n = |V|$.*

Theorem 7 can be extended to the pair outerplanar graph - cycle.

Theorem 8 *Let G be an outerplanar graph and let C be a simple cycle such that $V(G) = V(C) = V$. G and C can be simultaneously embedded with fixed edges in $\mathcal{O}(n)$ time, using at most one bend per edge, on an integer grid of size $\mathcal{O}(n^2) \times \mathcal{O}(n^2)$, where $n = |V|$.*

Theorems 7 and 8 can be extended to the more general case in which the two graphs have only a subset of their vertices in common. Let G_1 and G_2 be a pair of planar graphs such that $V(G_1) \cap V(G_2) = V$. A *simultaneous embedding with fixed edges* of G_1 and of G_2 is a a pair of crossing-free drawings Γ_1 and Γ_2 of G_1 and G_2, respectively, such that for every vertex $v \in V$ we have $\Gamma_1(v) = \Gamma_2(v)$ and for every edge $e \in E(G_1) \cap E(G_2)$ we have $\Gamma_1(e) = \Gamma_2(e)$.

Theorem 9 *Let G_1 be an outerplanar graph and let G_2 be either a simple path or a simple cycle such that $V(G_1) \cap V(G_2) = V$. G_1 and G_2 can be simultaneously embedded with fixed edges in $\mathcal{O}(n)$ time, using at most one bend per edge, on an integer grid of size $\mathcal{O}(n^2) \times \mathcal{O}(n^2)$, where $n = |V(G_1) \cup V(G_2)|$. If G_2 is a simple path its edges are drawn as straight-line segments and the grid size is $\mathcal{O}(n) \times \mathcal{O}(n^2)$.*

References

[1] F. Bernhart and P. C. Kainen. The book thickness of a graph. *J. Combin. Theory*, Ser. B 27:320–331, 1979.

[2] P. Brass, E. W. Cenek, C. A. Duncan, A. Efrat, C. Erten, D. P. Ismailescu, S. G. Kobourov, A. Lubiw, and J. S. B. Mitchell. On simultaneous planar graph embeddings. In *Proc. WADS 2003*, pp. 243–255, 2003.

[3] N. Chiba and T. Nishizeki. The hamiltonian cycle problem is linear-time solvable for 4-connected planar graphs. *J. of Algorithms*, 10:189–211, 1989.

[4] E. Di Giacomo, W. Didimo, G. Liotta, and S. K. Wismath. Book embeddings and point-set embeddings of series-parallel digraphs. In *Proc. GD 2002*, pp. 162–173, 2002.

[5] E. Di Giacomo, W. Didimo, G. Liotta, and S. K. Wismath. Curve-constrained drawings of planar graphs. *CGTA*, 30:1–23, 2005.

[6] C. Erten and S. G. Kobourov. Simultaneous embedding of planar graphs with few bends. In *Proc. GD 2004*, to appear.

[7] M. Kaufmann and R. Wiese. Embedding vertices at points: Few bends suffice for planar graphs. *JGAA*, 6(1):115–129, 2002.

Author Index