



22nd Annual Fall Workshop on Computational Geometry

Workshop Proceedings

November 9-10 2012

Dept. of Computer Science

College Park, MD

Forward

The 22nd Fall Workshop on Computational Geometry was held at the University of Maryland, College Park MD, on November 9-10, 2012. The workshop was supported by generous gifts from:

- University of Maryland Institute for Advanced Computer Studies (UMIACS)
- University of Maryland Department of Computer Science
- University of Maryland Department of Mathematics
- Google

The Fall Workshop followed the tradition of being an annual forum for the presentation, discussion, and dissemination of new results, as well as free exchange of questions and research challenges.

These proceedings contain abstracts of the technical presentations accepted at the Fall Workshop. The 30 contributed talks given at the Fall Workshop covered a broad range of topics, from theoretical to practical aspects of Computational Geometry and related fields.

David M. Mount
University of Maryland

Joseph S. B. Mitchell
Stony Brook University

Program Committee of FWCG'12

- F. Betül Atalay (Saint Joseph's University)
- Esther Ezra (Courant Institute of Mathematical Science)
- Joseph S.B. Mitchell (co-chair, Stony Brook University)
- David Mount (co-chair, University of Maryland)
- Thomas J. Peters (University of Connecticut)
- Jeff M. Phillips (University of Utah)
- Jack Snoeyink (University of North Carolina)
- Ileana Streinu (Smith College)

Local Organization

- Eunhui Park (University of Maryland)
- David Mount (University of Maryland)

Workshop Program

Friday Nov, 9, 2012

1st Session

- 9:30 Paul Accisano and Alper Üngör
On the Curve/Point Set Matching Problem
- 9:50 S. Sankararaman, P. K. Agarwal, T. Mølhave and A. P. Boedihardjo
Identifying Common Portions Between Two Trajectories
- 10:10 Jack Snoeyink and Vishal Verma
Transformations to Critical Arcgons: Progress on Tighter Bounds for Delaunay Stretch
- 10:30 Jack Snoeyink and Clinton Freeman
A Geometric Workbench for Degree-Driven Algorithm Design
- 10:50 Break

2nd Session

- 11:10 John Iacono and Kostyantyn Mazur
Tactix on an S-shaped Board
- 11:30 Sarah R. Allen and John Iacono
Packing Identical Simple Polygons is NP-hard
- 11:50 Rainer Penninger and Ivo Vigan
Point Set Isolation Using Disks is NP-complete
- 12:10 Andrew Winslow
Inapproximability of the Smallest Superpolyomino Problem
- 12:30 Lunch break

3rd Session

- 2:00 Bahman Kalantari
A Characterization Theorem and an Algorithm for a Convex Hull Problem
- 2:20 Michael Biro, Justine Bonanno, Roozbeh Ebrahimi and Lynda Montgomery
Approximation Algorithms for Outlier Removal in Convex Hulls
- 2:40 Rezaul Chowdhury, Pramod Ganapathi and Yuan Tang
The Range 1 Query (R1Q) Problem
- 3:00 Christopher Vo and Jyh-Ming Lien
Group Following in Monotonic Tracking Regions
- 3:20 Break

4th Session

- 3:40 Asish Mukhopadhyay, Chris Drouillard and Godfried Toussaint
Guarding Simple Polygons with Semi-open Edge Guards
- 4:00 Akitoshi Kawamura and Yusuke Kobayashi
Fence Patrolling by Mobile Agents with Distinct Speeds
- 4:20 Michael Biro and Justin Iwerks
Watchman Paths in Disk Grids
- 4:40 Jonathan Lenchner and Eli Packer
Visibility Problems Concerning One-Sided Segments
- 5:10 **Open Problem Session**
- 6:00 Adjourn

Saturday Nov. 10, 2012

5th Session

- 9:00 Hu Ding and Jinhui Xu
Chromatic Clustering in High Dimensional Space
- 9:20 Jeff M. Phillips and Bei Wang
Kernel Distance for Geometric Inference
- 9:40 Jiemin Zeng and Jie Gao
A Linear Time Euclidean Spanner on Imprecise Points
- 10:00 Break

6th Session

- 10:20 Ning Xu, Peter Braß and Ivo Vigan
An improved Algorithm in Shortest Path Planning for a Tethered Robot
- Kan Huang, Chien-Chun Ni, Rik Sarkar, Jie Gao and Joseph S. B. Mitchell
- 10:40 *Bounded Stretch Homotopic Routing Using Hyperbolic Embedding of Sensor Networks*
- 11:00 J. Li and T. J. Peters
Isotopy Convergence Theorem
- 11:20 Break

7th session: Invited Talk

- 11:30 **William Goldman** (University of Maryland)
The Experimental Geometry Lab at the University of Maryland
- 12:30 Lunch break

8th Session

- 2:00 Erin W. Chambers, Tao Ju and David Letscher
Medial Residue On a Piecewise Linear Surface
- 2:20 Erin W. Chambers, Tao Ju, David Letscher and Lu Liu
Burning the Medial Axes of 3D Shapes
- 2:40 Gary L. Miller and Donald R. Sheehy
A New Approach to Output-Sensitive Voronoi Diagrams and Delaunay Triangulations
- 3:00 Xiaotian Yin, Wei Han, Xianfeng Gu, Shing-Tung Yau
Solving the Cutting Flow Problem for Prismatic Mesh Subdivision
- 3:20 Break

9th Session

- 3:40 Adrian Dumitrescu and Csaba D. Tóth
Packing Disks that Touch the Boundary of a Square
- 4:00 O. Aichholzer, S. R. Allen, G. Aloupis, L. Barba, P. Bose, J.-L. De Carufel, J. Iacono, S. Langerman and D. L. Souvaine
Sum of Squared Edges for MST of a Point Set in a Unit Square
- 4:20 Stefan Langerman, Mudassir Shabbir and William Steiger
Computing Small Hitting Sets for Convex Ranges
- 4:40 Boris Aronov, John Iacono, Özgür Özkan and Mark Yagnatinsky
Finding the Thinnest V-shape with Few Outliers
- 5:00 Break
- 5:10 **Open Problem Follow-up and Closing Remarks**
- 6:00 End of Workshop

On the Curve/Point Set Matching Problem

Paul Accisano*

Alper Üngör *

Abstract

Let P be a polygonal curve in \mathbb{R}^d of length n , and S be a point set of size k . We consider the problem of finding a polygonal curve Q on S such that all points in S are visited and the Fréchet distance between Q and P is at most a given ε . We show that this problem is NP-complete, regardless of whether or not the points of S are allowed to be visited more than once.

1 Introduction

Measuring the similarity between two geometric objects is a fundamental problem in many fields of science and engineering. However, to perform such comparisons, a good metric is required to formalize the intuitive concept of “similarity.” Among the many metrics that have been considered, Fréchet distance has emerged as a popular and powerful choice, especially when the geometric objects are curves. Shape matching with Fréchet distance has been applied in many different fields, including handwriting recognition [7], protein structure alignment [5], and vehicle tracking [3].

In this abstract, we consider the basic problem of measuring the similarity of two polygonal curves. However, in our problem, the input is only partially defined. Instead of being given both curves, we are given only one polygonal curve P as well as a point set S . Our problem is to complete this partial input by constructing a polygonal curve Q that best matches the given curve, under the restriction that the constructed curve’s vertices are exactly S . We show that, under the Fréchet distance metric, this problem is NP-complete. Figure 1 shows an example problem instance and its solution.

2 Previous Work and New Results

Given two curves $P, Q : [0, 1] \rightarrow \mathbb{R}^d$, the *Fréchet distance* between P and Q is defined as $\delta_F(P, Q) = \inf_{\sigma, \tau} \max_{t \in [0, 1]} \|P(\sigma(t)), Q(\tau(t))\|$, where $\sigma, \tau : [0, 1] \rightarrow [0, 1]$ range over all continuous non-decreasing surjective functions [4].

The decision version of the Fréchet distance problem asks, given two geometric objects and a real number $\varepsilon > 0$, is the Fréchet distance between the two objects less

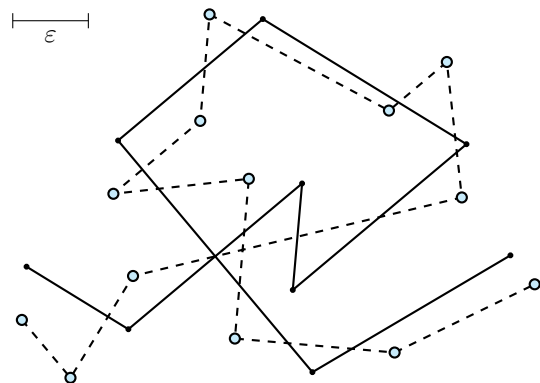


Figure 1: A problem instance and its solution. The input is the solid line and the circle points, and the solution is the dotted line.

than or equal to ε ? Alt and Godau [1] showed that, when the objects in question are polygonal curves of length n and m , this problem can be solved in $O(nm)$ time. They also showed that finding the exact Fréchet distance between the two curves can be done in $O(nm \log(nm))$ time using parametric search.

Maheshwari et al. [6] examined the following variant of the Fréchet distance problem, which we refer to as the Curve/Point Set Matching (CPSM) problem. Given a polygonal curve P of length n , a point set S of size k , and a number $\varepsilon > 0$, determine whether there exists a polygonal curve Q on a subset of the points of S such that $\delta_F(P, Q) \leq \varepsilon$. They gave an algorithm that decides this problem in time $O(nk^2)$. They also showed that the curve of minimal Fréchet distance can be computed in $O(nk^2 \log(nk))$ time using parametric search.

Wylie and Zhu [8] also explored the CPSM problem from the perspective of discrete Fréchet distance. In contrast to the continuous Fréchet distance defined above, the discrete Fréchet distance only takes into account the distance at the vertices along the paths. They formulated four versions of the CPSM problem depending on whether or not points in S were allowed to be visited more than once (Unique vs. Non-unique) and whether or not Q was required to visit all points in S at least once (All-Points vs. Subset). They showed that, under the discrete Fréchet distance metric, both non-unique versions were solvable in $O(nk)$ time, and both unique versions were NP-complete.

In this abstract, we show that the Continuous All-

*Dept. of Computer & Info. Sci. & Eng., University of Florida, {accisano, ungor}@cise.ufl.edu

Points versions of the CPSM problem, both Unique and Non-unique, are NP-complete. Table 1 shows the eight versions of the problem, with our results highlighted.

		Discrete	Continuous
Subset	Unique	NP-C [8]	Open
	Non-Unique	P [8]	P [6]
All-Pts	Unique	NP-C [8]	NP-C*
	Non-Unique	P [8]	NP-C*

Table 1: Eight versions of the CPSM problem and their complexity classes. New results starred.

3 Reduction Outline

The well-known 3SAT problem takes as input a Boolean formula with clauses of size 3, and asks whether there exists an assignment to the variables that makes the formula evaluate to TRUE. If we restrict the input to formulas in which each literal occurs exactly twice, the problem becomes the (3,B2)-SAT problem. This may seem to be a rather extreme restriction, and, indeed, formulas of this type with less than 20 clauses are always satisfiable. However, despite this restriction, the problem was shown to be NP-complete in [2], and an example of an unsatisfiable formula with 20 clauses was presented.

Let Φ be a formula given as input to the (3,B2)-SAT problem. We construct a polygonal curve P and a point set S such that Φ is satisfiable if and only if there exists polygonal curve Q whose vertices are exactly S with Fréchet distance at most ε from P . Our construction is somewhat lengthy and involves the construction of a complex gadget, as well as a number proofs about its properties. For this reason, we provide only a brief summary of the construction in this abstract.

First, we construct a gadget consisting of components of P and S that will force any algorithm to choose between two possible polygonal path constructions. The gadget is constructed in such a way that these two choices are the only possible polygonal paths along the gadget's component of S with Fréchet distance at most ε from P . These two path possibilities will correspond to TRUE and FALSE assignments for a given variable.

Then, we create a series of points in S to represent the clauses in Φ , one point for each clause. For each variable, a gadget will be placed so that the pair clause points representing the clauses in which the variable's positive instances occur are only reachable along one of the two curve possibilities, and likewise for the negative instances. Once this has been done for each variable in Φ , any polygonal curve Q whose vertices are exactly S with Fréchet distance at most ε from P will correspond to an assignment to the variables of Φ in which every clause is satisfied, thus making the formula evaluate to

TRUE. Furthermore, if no such curve exists, then there can be no such satisfying assignment for Φ .

Given that the problem of determining the Fréchet distance between two given polygonal curves is in P, the CPSM problem is clearly in NP. This leads to our main result.

Theorem 1 *The All-points Continuous CPSM Problem is NP-complete.*

There are still a number of details that have been omitted for the sake of brevity, including the proof that placing the gadgets in the necessary positions is always geometrically possible. These will be included in our full paper. We also plan to explore various generalizations of this problem. For example, the given geometric object could perhaps be a tree or graph, or the point set could be given as imprecise points or regions.

References

- [1] H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5(01n02):75–91, 1995.
- [2] Piotr Berman, Marek Karpinski, and Alex D. Scott. Approximation hardness of short symmetric instances of MAX-3SAT. *Electronic Colloquium on Computational Complexity (ECCC)*, (049), 2003.
- [3] Sotiris Brakatsoulas, Dieter Pfoser, Randall Salas, and Carola Wenk. On map-matching vehicle tracking data. In *Proceedings of the 31st international conference on Very large data bases, VLDB '05*, pages 853–864. VLDB Endowment, 2005.
- [4] George Ewing. *Calculus of variations with applications*. Dover Publications, New York, 1985.
- [5] M. Jiang, X. Ying, and B. Zhu. Protein structure–structure alignment with discrete frechet distance. *Journal of bioinformatics and computational biology*, 6(01):51–64, 2008.
- [6] Anil Maheshwari, Jörg-Rüdiger Sack, Kaveh Shahbaz, and Hamid Zarrabi-Zadeh. Staying close to a curve. In *CCCG*, 2011.
- [7] R. Sriraghavendra, K. Karthik, and C. Bhat-tacharyya. Fréchet distance based approach for searching online handwritten documents. In *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, volume 1, pages 461–465. IEEE, 2007.
- [8] T. Wylie and B. Zhu. Discretely following a curve (short abstract). In *Computational Geometry: Young Researchers Forum (CG:YRF)*, 2012.

Identifying Common Portions between Two Trajectories

Swaminathan Sankararaman* Pankaj K. Agarwal* Thomas Mølhave* Arnold P. Boedihardjo†

*Dept. of Computer Science, Duke University. email: {swami, pankaj, thomasm}@cs.duke.edu

†U.S. Army Corps of Engineers. email: arnold.p.boedihardjo@erdc.dren.mil

1 Introduction

Trajectories are functions from a time domain—an interval on the real line—to \mathbb{R}^d with $d > 1$, and observed as sequences of points sampled from them. A fundamental problem in analyzing this data is that of identifying common patterns between pairs or among groups of trajectories observed as sequences of sampled points.

Let $P = \langle p_1, \dots, p_m \rangle$ and $Q = \langle q_1, \dots, q_n \rangle$ be two sequences of points in \mathbb{R}^d , sampled from two trajectories γ_1 and γ_2 defined over the time interval $[0, 1]$. For simplicity, we assume that P and Q are points sampled from the images of the trajectories and we ignore the temporal component.¹ Since, in practice, the underlying continuous trajectories γ_1 and γ_2 are not known but we observe only the sampled points P and Q , we will work in the discrete setting where we are only concerned with these sample points. In this abstract, we will refer to the discrete sample points P , Q as the input trajectories.

We wish to compute correspondences between points belonging to similar portions of these trajectories while distinguishing these portions from the dissimilar ones. The following issues with trajectory sampling must be taken into account when identifying similarity: (i) significantly different sampling rates, (ii) presence of noise/outliers which must be distinguished from dissimilarities, and (iii) presence of significant unobserved portions on the trajectories with no sample points.

Background. A common choice for measuring trajectory similarity is the Fréchet distance [1] defined as follows. A *reparameterization* is a continuous non-decreasing surjection $\alpha : [0, 1] \rightarrow [0, 1]$, such that $\alpha(0) = 0$ and $\alpha(1) = 1$. The Fréchet distance $\text{Fr}(\gamma_1, \gamma_2)$ is given by:

$$\text{Fr}(\gamma_1, \gamma_2) = \inf_{\alpha, \beta} \max_{t \in [0, 1]} \|\gamma_1(\alpha(t)) - \gamma_2(\beta(t))\|,$$

where $\|\cdot\|$ is the underlying norm (typically the Euclidean norm), and α and β are reparameterizations of

¹Strictly speaking, a trajectory is the graph of the underlying function, and what we have are the curves traced by the two trajectories, but we will not distinguish between the two.

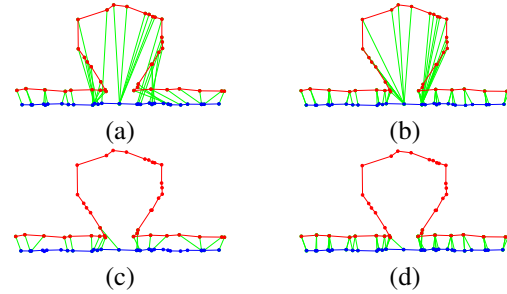


Figure 1. Comparison of measures: (a) Fréchet distance, (b) average Fréchet distance, (c) sequence alignment based method, (d) our model. Green edges indicate correspondences.

$[0, 1]$. Since we only observe a finite set of sample points, we may define a discrete version of the Fréchet distance where the reparameterizations are discrete functions restricted to the sampled points P and Q .

A set of correspondences yielding the optimal Fréchet distance is not necessarily a good indicator of similarity due to the large number of such correspondences; see Fig. 1(a). The *average Fréchet distance* which minimizes the average distance of the correspondences rather than the maximum distance provides a better set of correspondences. However, if there are significant dissimilar portions, possibly due to actual deviations rather than outliers, the results are not meaningful due to the requirement of correspondences for all points; see Fig. 1(b).

In computational biology, the technique of pairwise sequence alignment [2, cf. Chapter 2] is designed to distinguish similar and dissimilar portions between biological sequences. Given two sequences A and B , their alignment is expressed by writing them in two rows such that similar characters are placed in the same column. Characters in one sequence with no similar character in the other sequence are aligned with a blank character. A maximal contiguous sequence of blank characters is termed a *gap*. The goal is to optimize a scoring function which assigns a score for aligning two characters (incentive or penalty depending on their similarity) and a penalty for gaps.

We may extend the sequence-alignment model to the

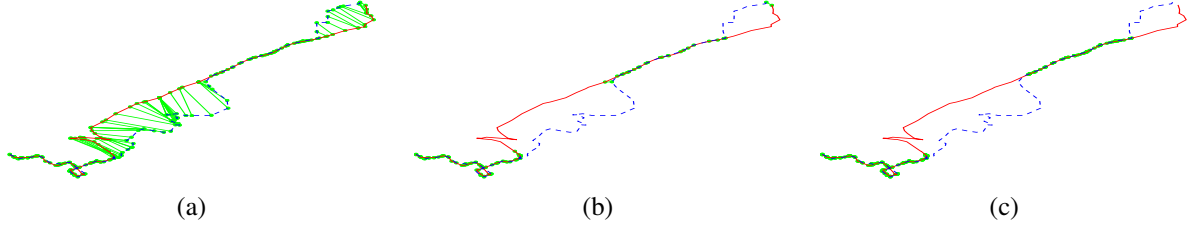


Figure 2. Real data: (a) average Fréchet distance, (b) sequence alignment based approach, (c) our model. Green edges indicate correspondences.

alignment of trajectories with the choice of an appropriate scoring function. However, as Fig. 1(c) shows, non-uniform sampling rates cause similar portions to be designated as gaps since correspondences are one-to-one.

2 Model

As noted above, the average Fréchet distance yields good correspondences for similar portions even with different sampling rates while sequence alignment identifies dissimilarities accurately. We capture the advantages of these two methods under a unifying notion of assignments.

Definition 2.1. An *assignment* for P and Q is a pair of functions $\alpha : P \rightarrow Q \cup \{\perp\}$ and $\beta : Q \rightarrow P \cup \{\perp\}$ for the points of P and Q respectively. If $\alpha(p_i) = \perp$ (or $\beta(q_j) = \perp$), then p_i (or q_j) is called a *gap point*. A maximal contiguous sequence of gap points in P or Q is called a *gap*. An assignment is *monotone* if it satisfies the following conditions: (i) if $\alpha(p_i) = q_j$ implies that for all $i' > i$, $\alpha(p_{i'}) \in \{\perp\} \cup \{q_{j+1}, \dots, q_n\}$, (ii) $\beta(q_j) = p_i$ implies that for all $j' > j$, $\beta(q_{j'}) \in \{\perp\} \cup \{p_{i+1}, \dots, p_m\}$.

Intuitively, if a point $p_i \in P$ lies on a similar portion of the two trajectories then $\alpha(p_i)$ defines the point on Q to which p_i corresponds, and p_i is a gap point otherwise. A similar interpretation holds for $\beta(\cdot)$. Let $\Gamma(\alpha, \beta)$ denote the set of gaps in P and Q for the assignment α, β . We define the *score* of α, β , denoted by $\sigma(P, Q; \alpha, \beta)$, as

$$\sigma(P, Q; \alpha, \beta) = \sum_{\substack{p_i \in P \\ \alpha(i) \neq \perp}} \frac{1}{c + \|p_i - \alpha(p_i)\|^2} + \sum_{\substack{q_j \in Q \\ \beta(j) \neq \perp}} \frac{1}{c + \|q_j - \beta(q_j)\|^2} + \sum_{g \in \Gamma(\alpha, \beta)} (a + \Delta \cdot |g|),$$

where a, Δ and c are carefully chosen parameters, $\|\cdot\|$ is the L_2 -norm and $|g|$ is the length of a gap g . For a pair of points $p_i \in P$ and $q_j \in Q$, the difference in values $1/(c + \|p_i - q_j\|^2)$ versus Δ dictates the choice of whether to assign $\alpha(p_i) = q_j$ or $\beta(q_j) = p_i$ versus assigning one or both as gap points. Thus, Δ is chosen based on a distance threshold for similarity. The parameter a is used to

avoid extremely short gaps (of length less than l for some $l > 0$) which may be due to outliers rather than actual deviations and is set to $-l\Delta$.

A monotone assignment α, β which maximizes $\sigma(P, Q; \alpha, \beta)$ may be found by a dynamic programming algorithm in time $O(mn)$. This is essentially a more complicated version of the algorithm for sequence alignment. Fig. 1(d) shows the results which perform similarly to the average Fréchet correspondences in the similar portions while distinguishing the dissimilar portions as accurately as the sequence alignment based approach.

3 Discussion

Our framework is not limited to the scoring function described. For example, the sequence-alignment based approach, average Fréchet distance or other measures such as adaptations of edit-distance are easily incorporated into our model. Further, we may extend the dynamic programming algorithm to compute locally similar portions instead of global trajectory similarity in a manner similar to local alignment of sequences.

We have conducted experiments comparing the average Fréchet distance, sequence alignment and our model on a dataset of 145 trajectories of school buses in Athens, Greece [3]. Fig. 2 shows the comparison for a pair of trajectories from this set. As is clearly seen, sequence alignment “finds” the dissimilar portions accurately but the on close examination, we note that there are gaps even in the similar portions. This is rectified by our model which performs similarly to average Fréchet distance in the similar portions while avoiding dissimilar portions accurately.

References

- [1] H. Alt and M. Godau. Computing the Fréchet Distance between Two Polygonal Curves. *Int. J. Comput. Geometry Appl.*, 5:75–91, 1995.
- [2] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.
- [3] E. Frentzos, K. Gratsias, N. Pelekis, and Y. Theodoridis. Nearest neighbor search on moving object trajectories. In *SSTD 2005*, pages 328–345. 2005.

Transformations to critical arcgons: progress on tighter bounds for Delaunay stretch

Jack Snoeyink[†]

Vishal Verma[†]

Abstract

Chew observed that the stretch factor of L_2 Delaunay triangulations was at least $\pi/2$ in the paper that also established the L_1 Delaunay as the first geometric spanner; recent examples have raised the lower bound to 1.5932. The upper bound of 2.45 established in 1992 has also recently come down to below 2 by Xia. We outline a program that brings both upper and lower bounds closer to 1.6 by characterizing the properties of examples that are maximal under local transformations. Here we describe the characterization and our plan of bounding the stretch factor of L_2 Delaunay triangulations.

1 Introduction

The *Delaunay stretch* of a point set S is the maximum, for all pairs $p, q \in S$, of the ratio of the length shortest path from p to q in the Delaunay triangulation of S over $|pq|$. It is known that the Delaunay stretch for any point set is upper bounded by a constant [3, 4]. The best bounds published [5, 6] until now are [1.5932, 1.998]. Our project aims to narrow this interval, by transforming examples into canonical form without decreasing stretch. Our plan is sketched in three sections: In Sec. 2 we transform any given point set to an *arcgon*. In Sec. 3 we transform any arcgon to a *max arcgon*. In Sec. 4 we bound the stretch factors of max arcgons, thus bounding the Delaunay stretch.

2 Counterexamples and Arcgons

Fig. 1(a,b) shows example point sets with high Delaunay stretch. The points lie densely along the boundary of a union of discs. By controlled perturbation into general position, Delaunay edges in the disc interiors can be directed to preserve Delaunay stretch. This construction as a sequence of discs helps find lower bounds, but also helps compute upper bounds.

An *arcgon* is defined as an embedded graph whose

- edges are either circular arcs or straight lines.

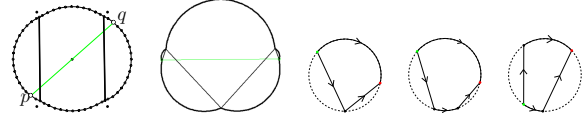


Figure 1: Point sets with high Delaunay stretch (a) $\kappa = 1.5846$ from [1], (b) our $\kappa = 1.59324$. Balanced (equal length) paths around three face types: (c) wedge, (d) anti-parallel stump, (e) parallel stump

- faces f are convex subsets of a disc c with vertices on the boundary of c in a sequence. The ends are circle segments, with special vertices p and q . Interior faces are *wedge* with 3 edges (Fig. 1(c)) or *stump* with 4 (Fig. 1(d,e)).
- outer boundary contains only circular arc edges. The edges in the interior, called *diagonals*, are straight line edges that connect the points of intersections of two neighboring discs.

Arcgons must satisfy two empty circle properties:

Local Delaunay: Vertices of face f lie on or outside of circles of neighboring faces of f .

pq -Delaunay: p, q on or outside circles of all faces.

In *realizable arcgons*, all diagonals intersect \overline{pq} .

From a given point set S we can extract a realizable arcgon without decreasing Delaunay stretch, κ .

Lemma 1 *Let $p, q \in S$ attain the maximum Delaunay stretch κ in the Delaunay triangulation of the points S . There is a realizable arcgon for which \overline{pq} has a stretch factor $\geq \kappa$.*

Proof Sketch: From the Delaunay triangulation of S , take the sequence of triangles that intersect the interior of segment \overline{pq} . Then in every triangle of this sequence, replace the edges that do not intersect the \overline{pq} with the corresponding arcs of the circumcircle. Similar transformations have been used in [2, 4, 5]. ■

3 Max Arcgons

Define the *complexity* of an arcgon to be the number of its faces. A *max arcgon* has stretch factor greater than all realizable arcgons of lower complexity, and

[†]Department of Computer Science, University of North Carolina at Chapel Hill. Email: {snoeyink,verma}@cs.unc.edu

not less than all realizable arcgons of equal complexity. Thus, Delaunay stretch is bounded by the stretch factor of max arcgons.

We now characterize max arcgons by studying local transformations

Lemma 2 *In a max arcgon A , the segment \overline{pq} does not pass through the endpoints of any of the diagonals*

We say that an edge e of an arcgon is *critical* if some shortest path between p and q passes through e . We can show that in max arcgons all edges are critical. Here we sketch the proof for one case.

Lemma 3 *In a max arcgon A , the circular arc edges of every stump face f are critical*

Proof Sketch: Let L be the part of the arcgon to the left of the face f and R be the part to the right. If an arc-edge of f , say e , is not critical then we can increase the stretch factor κ of A by rotating R about the center, o , of the circle associated with the face f . We parameterize this rotation on the angle θ subtended by the arc-edge e at o . Let $(')$ be represent the derivative with respect to θ . We show that $\kappa'' > 0$ whenever $\kappa' = 0$ i.e. κ can be increased by increasing or decreasing θ . ■

Using similar first derivative and second derivative analysis we can prove the following lemma:

Lemma 4 *Every max arcgon is realizable and*

- *The arc-edge of any wedge face is critical*
- *Diagonals adjacent to any stump face are critical*
- *Circular arc edges of two neighboring wedge faces cannot coincide.*

We are currently working to complete the proof that any diagonal adjacent to two wedge faces is critical. Together with Lem. 4, this would imply:

Claim 5 (To be established) *All the edges of a max arcgon are critical*

This claim constrains max arcgons to a class we call *critical arcgons*.

4 Upper bound on critical arcgons

In a *critical arcgon* interior faces are of three types:

- balanced wedge:** the length of the arc edge equals the sum of the two diagonals (Fig. 1(c)).
- balanced anti-parallel stump:** the difference in lengths of the two circular arc edges equals the sum of the two diagonals (Fig. 1(d)).
- balanced parallel stump:** the difference in lengths of the circular arc edges equals the difference in the diagonals (Fig. 1(e)).

Claim 6 *Let A be a critical arcgon, $\ell(A)$ be the length of the shortest path between the special vertices p, q of A . Let $d(A)$ be the length of the shortest geodesic path between p and q that passes through the faces of A . Let $L(A)$ be half of the perimeter of the arcgon A minus the end faces. Then,*

$$g(A) = \ell(A) - \frac{\pi}{2}d(A) - 0.04L(A) \leq 0$$

Note that if the critical arcgon is also realizable (as is the case with max arcgons) then $d(A)$ equals $|pq|$. Moreover $L(A) \leq \ell(A)$. Thus the above lemma immediately implies,

Theorem 7 *For a realizable critical arcgon A , the stretch factor*

$$\ell(A)/|pq| \leq \frac{\pi}{2(1-0.04)} \leq 1.636245$$

A construction of a critical arcgon raises the lower bound as well, so the maximum stretch factor of Delaunay triangulations would lie in $[1.59324, 1.63625]$.

Proof Sketch for Claim 6: The proof is by induction on the number of faces of the critical arcgon. The proof divides into three cases depending on whether the penultimate face (q is on the last face) is a balanced wedge, parallel, or anti-parallel stump. We can show that $g(A)$ attains a maximum when p lies on this penultimate face. The final inequality is established numerically. ■

References

- [1] P. Bose, L. Devroye, M. Löffler, J. Snoeyink, and V. Verma. Almost all Delaunay triangulations have stretch factor greater than $\pi/2$. *Comput. Geom.*, 44(2):121–127, 2011.
- [2] L. P. Chew. There are planar graphs almost as good as the complete graph. *Journal of Computer and System Sciences*, 39(2):205 – 219, 1989.
- [3] D. P. Dobkin, S. J. Friedman, and K. J. Supowit. Delaunay graphs are almost as good as complete graphs. *Discrete Comput. Geom.*, 5(4):399–407, May 1990.
- [4] J. Keil and C. Gutwin. Classes of graphs which approximate the complete Euclidean graph. *Discrete & Computational Geometry*, 7:13–28, 1992.
- [5] G. Xia. Improved upper bound on the stretch factor of Delaunay triangulations. In *Symposium on Computational Geometry*, pages 264–273, 2011.
- [6] G. Xia and L. Zhang. Toward the tight bound of the stretch factor of Delaunay triangulations. In *CCCG*, pages 175–180, 2011.

A Geometric Workbench for Degree-Driven Algorithm Design

Jack Snoeyink *

Clinton Freeman *

Abstract. We recall the design and implementation of a geometric algorithm workbench for implementing and presenting degree-driven geometric algorithms.

1 Introduction

Two and three dimensional geometric algorithms are often difficult to implement and convey to others. Algorithm *implementers* need to correct programming errors and ensure that degenerate situations are handled correctly. Traditional debuggers provide only textual or numerical representations of geometric data structures, and generating degenerate geometric input is a nontrivial task for which there is often little recourse. Algorithm *presenters* need to convey their ideas to audiences of researchers and students. Many presenters tend to use static depictions with verbal explication of algorithm mechanics. This type of presentation does not fully capture the dynamic nature of algorithms, and can be difficult for the audience to follow.

A *geometric algorithm workbench* aids algorithm implementers and presenters by providing facilities to dynamically visualize geometric algorithms. Implementers can visually inspect geometric relationships and properties of data structures, enabling quick recognition of erroneous computations. Presenters can produce animations of their algorithms, affording a clearer means of conveying essential ideas to their audience. Both types of geometers can interactively control the flow of execution and easily generate or visually specify degenerate input data.

Degree-driven algorithm design encourages robust geometric computing by minimizing an algorithm’s arithmetic precision with its running time and space [5]. Millman built a C++ library (DDAD) to facilitate the implementation of these algorithms; our aim is to build a workbench to support users of DDAD. The creation of this workbench mostly requires the application of techniques developed in previous software visualization research, but the addition of precision

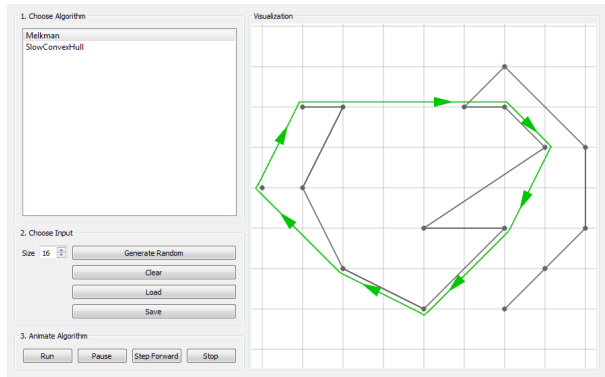


Figure 1: A user watches as Melkman’s algorithm handles a degenerate situation.

as a constraint provides for new avenues of exploration.

We begin by briefly reviewing previous workbench systems. Next, we explain how our current system’s facilities satisfy each user type and significant decisions we made during their implementation. Finally, we conclude with a discussion of how we can extend our system moving forward.

2 Previous Work

Initially, we spent some time reviewing existing software visualization systems to see if any might help us implement and present our algorithms. In 1997, Dobkin and Hausner [4] reviewed four geometric visualization systems: Workbench, XYZ GeoBench, Geomview, and GASP. Unfortunately, while these systems presented different ways of solving foundational challenges, support has long been discontinued. We also considered the Geometry Center’s GeoLab [2] and Stasko’s [7] more general algorithm animation software such as POLKA, SAMBA, and XTANGO, and found them similarly unsupported. Lacking a working geometric workbench, we decided to build our own system.

3 User Facilities

Two simple 2D convex hull algorithms, **SlowConvexHull** [1] and **Melkman’s algorithm** [6], provided

*Department of Computer Science, University of North Carolina at Chapel Hill. Email: {snoeyink, freeman}@cs.unc.edu

the initial target inputs for the system. For each algorithm, we first programmed an implementation, then recorded animations of them running on example input data to produce a corresponding Youtube video ¹. This development process placed us in the position of both implementer and presenter, and led us to develop facilities appropriate for both user types.

As implementers, we desired a system with four major capabilities. First, we needed a means of manipulating input data into degenerate situations to test that special cases were handled correctly. Second, we needed to run the algorithm and visually display the results of final and intermediate calculations. Third, upon discovering an incorrect result, we needed to single step the algorithm from the beginning on the same input data in order to see where the algorithm went awry. Finally, we needed visualization code to minimally invade our implementation code.

In response, our system provides four corresponding facilities. First, our system randomly generates either a random point set or simple polyline, and clicking and dragging moves vertices into different configurations. Second, our system maintains display lists which are updated as *interesting events* [3] occur. Third, our system uses threading to control the speed of execution and provides UI controls for starting, pausing, single stepping, and resetting the algorithm. Finally, our system embeds low level visualization functions in higher level geometric types to maintain code readability and ensure visualization consistency.

As presenters, implementer facilities already satisfied many of our needs, producing animations that captured the essential characteristics of each algorithm. However, we desired two additional capabilities: we needed to convey information not necessary for implementation purposes, and view the same algorithm in different ways. In response, our system provides two corresponding facilities: visualizations of arbitrary primitives that aren't directly used by the algorithm, and a passive *model-view-controller* architecture that can be extended with custom views.

4 Engineering Decisions

Two engineering decisions are of particular interest. First, we wanted to use model-view-controller to structure our design, but needed the traditional model concept to encompass a geometric algorithm.

¹See: <http://cs.unc.edu/~freeman/GAV/>

Extracting visual representations of operations and data structures without user guidance is a difficult, if not impossible, task. We decided to maintain a separate visual model of display lists, which the algorithm implicitly updates. Second, we needed to track visual semantics on geometric primitives so previous states could be restored (e.g. a hull segment is invalidated). We decided to store a semantic stack for each primitive; low level visualization functions push and pop new states as the algorithm executes.

5 Conclusion

Implementing a geometric algorithm workbench is a challenging task with a rich set of problems encompassing a variety of disciplines. While we continue extending our system by animating more algorithms, two questions provide opportunities for further exploration. First, how can we extend our workbench to highlight precision as a resource? An answer will help thematically differentiate the project from previous work. Second, given that so many past systems fell into disuse, how can we build our workbench to have better longevity? An answer will help solidify a foundation for future work.

Our workbench continues to grow and has not yet reached a state suitable for distribution to the geometry community. As the underlying design and outward interface stabilize, we will release a version for download ¹.

References

- [1] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. 3rd edition, 2008.
- [2] P. de Rezende and W. Jacometti. Animation of geometric algorithms using GeoLab. In *Proceedings of the ninth annual symposium on Computational geometry*, SCG '93, pages 401–402, 1993.
- [3] C. Demetrescu, I. Finocchi, and J. Stasko. Specifying Algorithm Visualizations: Interesting Events or State Mapping? In *Revised Lectures on Software Visualization, International Seminar*, pages 16–30, 2002.
- [4] A. Hausner and D. P. Dobkin. Making Geometry Visible: An Introduction to the Animation of Geometric Algorithms, 1997.
- [5] G. Liotta, F. P. Preparata, and R. Tamassia. Robust proximity queries: an illustration of degree-driven algorithm design. In *Proceedings of the thirteenth annual symposium on Computational geometry*, SCG '97, pages 156–165, 1997.
- [6] A. Melkman. On-line construction of the convex hull of a simple polyline. *Inf. Process. Lett.*, 25(1):11–12, 1987.
- [7] J. Stasko. Software Visualization research at GVV, 2011. <http://www.cc.gatech.edu/gvu/ii/softvis/>.

Tactix on an S-shaped Board

John Iacono and Kostyantyn Mazur

Polytechnic Institute of New York University

Abstract

TACTIX is a geometric variant of the classic game NIM. A full characterization of a class of instances of TACTIX is presented where it is shown how to efficiently compute which player has a winning strategy.

Nim. The classic game NIM is traditionally played with some distinct piles of objects such as coins. There are two players which take turns alternately. Each player may take any number of objects from any pile (from one object to the entire pile), but may only take from the one pile that the player chooses for that turn. After each turn, the piles grow smaller, and eventually, all the objects are gone. There are two versions of NIM that differ on what happens then: NIM with the *normal play convention* has the player that takes the last counter win, while *misère* NIM has this player lose. NIM is a decidedly one-dimensional game; in this paper we consider a natural two-dimensional geometric variant of NIM.

Tactix. TACTIX, a game invented by Piet Hein, is a two-dimensional version of NIM. TACTIX is also a two-player game. There is a 4×4 grid of counters, and each player is allowed to take any horizontal or vertical sequence of consecutive counters. TACTIX is played with the *misère* rule, meaning that the player that takes the last counter loses. TACTIX has been solved; the second player has a winning strategy [1].

If the normal play convention were used, then this would be more obvious; the second player could make the 180° rotation of whatever move that the first player chose, and repeat until the last counter. This strategy would also work on a square grid of even size, or on a rectangular grid with both dimensions even. On a rectangular grid with one odd dimension, the first player has a winning strategy: take the entire middle column (or row, if there is an odd number of rows), which leaves two rectangular boards of equal size, and copy whichever move the second player chooses to make on one board on the other board. However, these strategies do not work in TACTIX, where the *misère* convention is used.

Tactix. TACTIX is a variation of TACTIX (note lower case *t*), played with the normal play convention (the rules regarding legal moves are the same). Since requiring a rectangular starting position makes this game trivially solved, any subset of a rectangular grid is an allowable starting position in Tactix.

Impartial Games. All three games (NIM, TACTIX, and TACTIX) share the property of the allowable moves depending only on the game position and not on which of the two players is on move. Such games are called *impartial games*. This condition negates the requirement of stating who is on move; each position has a specific result with reference to whoever is on move. Most games are not impartial; for instance, GO is not impartial, since a player can only play a stone of the color assigned to the player.

Finite Games. All three games also share the property of being finite. A finite game is one where, if it is played starting in any given position, a final result is always reached after no more than a number of moves that only depends on the starting position. In any of these three games, each move removes at least one counter (or object in the case of NIM), and the game is decided when there are no more counters (or objects), so the number of moves played is at most the number of counters. One way for a game not to be finite is if a position can be repeated by a sequence of moves without a rule governing what happens then. CHESS without the threefold repetition or 50 move rules is an example of a game that is not finite. However, with the 50-move or threefold repetition rules, the game of CHESS is finite.

Sprague-Grundy. TACTIX is equivalent to NIM, as are all impartial games with the normal play convention, by the Sprague-Grundy theorem [3, 4]. This gives each starting position a so-called NIM value, also known as the *number* or *Grundy value*. The NIM value of a position is given as the lowest nonnegative integer that is not the NIM value of any resulting position after any move (and it is zero for a position with no legal moves). There is an efficient way to calculate the NIM value of a disjoint combination of two or more of these games (where the games are played simultaneously and a legal move is selecting one game and making a legal move there), namely that the NIM value of the combination is the bitwise

XOR of the NIM values of the individual games. A NIM value gives a determination of whether or not the first player has a winning strategy: the first player has a winning strategy unless the NIM value is zero. This gives a recursive way to compute the NIM value of a position, but it can easily be exponential to compute for general positions in certain games, even with dynamic programming.

Thus the main question one can ask in the study of a particular game is, how efficiently can you determine which player has a winning strategy? For an impartial and finite game with the normal play convention, this can be done by deriving an algorithm to compute the NIM value of the game.

Monotonic boards. For TACTIX boards of the form, which are shaped like a staircase and which we call *monotonic*:

×	...	×						
		×	...	×				
					
						×	...	×

where \times represents a counter, there is a polynomial-time dynamic programming algorithm to compute the NIM value, similar to the solution to LINEAR CRAM described in [2]. Removing any counters leaves two disjoint groups of counters, henceforth each move must be entirely in one group or the other. The NIM value of the resulting combination is the bitwise XOR of the NIM values of the two groups left. In the direct solution generated by the definition of the NIM value, there are only $O(n^2)$ (where n is the number of counters) possible connected groups, one per start and end point, the calculation of the NIM value of each only requires a polynomial number of lookups of NIM-values of smaller connected groups (as there are only $O(n^2)$ legal moves in any position). When such a NIM value is obtained, it is memoized, or stored in a table to be looked up later. This ensures that the NIM value of any given connected group is calculated only once. This solution thus runs in polynomial time, $O(n^4)$ if $\log n$ is word-sized.

If the vertical connections were entirely disallowed, then there is an easy solution, namely calculating the bitwise XOR of the number of counters on each row, since this is nothing more than NIM. This method does not work for monotonic TACTIX, be-

cause of the possibility of a vertical move (taking two vertically-adjacent counters). This raises a question: is there a method of computing the NIM value that is faster than the dynamic programming algorithm? Our main result is that if the number of lines is limited to two, then the answer is yes. This is the type of starting position we call a *S-shaped board*:

×	...	×		
		×	...	×

Result. The NIM value of a TACTIX game on an S-shaped board with a counters on the top and b counters on the bottom is:

$$r(a, b) = \begin{cases} r(a', b'), & r(a', b') < a' + b' \\ a' + b' + 1, & r(a', b') = a' + b' \\ & \text{and } a' + b' < m - 1 \\ a + b, & r(a', b') = a' + b' \\ & \text{and } a' + b' \geq m - 1 \end{cases}$$

where m is the lowest power of 2 below or at a , $a' = a - m$, and $b' = b - m$. This only applies if the lowest power of 2 below or equal to b is also m ; if not, then the NIM value is simply the number of counters.

This formula can be evaluated in time $O(\log n)$, where $n = a + b$.

The proof of this result can be found as Lemma 23 in the draft which was submitted with this paper, the proof of which occupies pages 23-42. Efficiently determining which side has a winning strategy for more general versions of TACTIX remains open.

References

- [1] M. Beeler. Taxtix. *HACKMEM*, 74, 1972.
- [2] Erik D. Demaine. Playing games with algorithms: Algorithmic combinatorial game theory. *CoRR*, cs.CC/0106019, 2001.
- [3] P. M. Grundy. Mathematics and games. *Eureka*, 1:6–8, 1939.
- [4] R. P. Sprague. Uber mathematische kampfspiele. *Tohoku Mathematical Journal*, 41:438–444, 1935-36.

Packing identical simple polygons is NP-hard

Sarah R. Allen and John Iacono

Polytechnic Institute of New York University

Abstract

Given a small polygon S , a big simple polygon B and a positive integer k , it is shown to be NP-hard to determine whether k copies of the small polygon (allowing translation and rotation) can be placed in the big polygon without overlap. Previous NP-hardness results were only known in the case where the big polygon is allowed to be non-simple. A novel reduction from PLANAR-CIRCUIT-SAT is presented where a small polygon is constructed to encode the entire circuit.

Introduction.

Packing is a fundamental problem in computational geometry. In this paper we study the problem of packing multiple copies of a small object inside a big object:

Simple Polygon Packing. *Given a small simple polygon S , a big simple polygon B , and a positive integer k , is it possible to place k copies (allowing translation and rotation) of the small polygon inside the big polygon without overlap?*

This problem was heretofore neither known to be in P, nor in NP, nor to be NP-hard. Here we show it is NP-hard.

Our result is the first to establish the hardness of packing of multiple copies of a simple polygon inside another simple polygon. Previous reductions for related problems fall into two categories. In the case of multiple small polygons, a reduction from KNAPSACK or PARTITION is easy. In the case of having a nonsimple big polygon, the reduction in [3] is from PLANAR-CIRCUIT-SAT. Such a reduction creates a big polygon which is essentially a drawing of the circuit, where the interior of the big polygon represents the wires and the gates, but where there are holes between all of the wires. Without the ability to literally create a big polygon that uses holes to create a circuit drawing, nothing was known. Our construction is also a reduction from PLANAR-CIRCUIT-SAT, but in a completely different manner. Previously the circuit was encoded in the big polygon; our big polygon is

independent of all aspects of the circuit, other than the circuit size, while the circuit is encoded entirely in the small polygon.

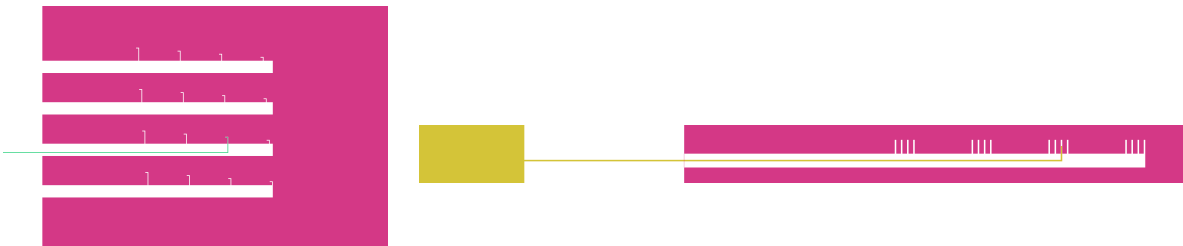
However, because our construction creates a small polygon which is nonconvex and polynomial in size, there remains a range of open problems relating to the packing of identical polygons. The simplest such variation (most likely to be in P) would be: given as the big polygon an orthogonally convex simple polygon drawn on a unit grid, how many grid-aligned 2×2 unit squares can be packed? (This is a slightly easier variant of problem 56 on the *Open Problems Project* [1], which was shown to be in NP in [2]).

Reduction overview. We give here a very high level description of our reduction. We reduce from a variant of circuit-sat where the circuit is drawn in a planar embedding on an $n \times n$ grid. The idea is to have our small polygons be unit-square-like where n^2 of them can only pack into a slightly larger than $n \times n$ -sized square-like large polygon if the packing is done according to a grid. The polygons deviate from being perfect squares by having several inclusions and exclusions; these force a small polygon to have certain interactions with the neighboring small polygon if a packing of n^2 small polygons is to be achieved. Each polygon will have a number of allowable vertical shifts, which among other things, represent a series of truth values. The construction is very carefully constructed such that the position of a small polygon relative to its neighbor will encode its position in the grid packing, its truth value, as well as the truth value of two neighboring small polygons. Given that this relative positioning encodes all this information, inclusions can be made to allow or deny shifts encoding configurations that are consistent with the given planar circuit on a grid.

- [1] Erik D. Demaine, Joseph S. B. Mitchell, and Joseph O'Rourke. The open problems project. <http://cs.smith.edu/~orourke/TOPP/Welcome.html>.
- [2] Dania El-Khechen, Muriel Dulieu, John Iacono, and Nikolaj van Omme. Packing 22 unit squares into grid polygons is NP-complete". In *CCCG*, pages 33–36, 2009.
- [3] Robert J. Fowler, Mike Paterson, and Steven L. Tanimoto. Optimal packing and covering in the plane are NP-complete. *Inf. Process. Lett.*, 12(3):133–137, 1981.



Figure 1: An illustration of our reduction. The 9 colored small polygons are all identical, and the inner white-black boundary defines the large polygon. Note that for ease of viewing the whitespace and other visual elements have been exaggerated dramatically, and certain elements of the construction have been simplified. The inclusions and exclusions on the large polygon form an arithmetic progression. To the lower left is a closeup of what we call the *nailer*; this forces a unique positioning depending on a small polygon's location in the grid. To the bottom-right is a closeup of a horizontal protrusion and inclusion; these are designed so that the notch occupied is a unique function of the position. By removing some notches, some configurations can be allowed or forbidden. The vertical groups of three are meant to illustrate states (such as true and false). For reasons described in the full paper, 64 such vertical groups are needed instead of the three illustrated.



Point Set Isolation Using Unit Disks is NP-complete

Rainer Penninger¹ and Ivo Vigan^{*2}

¹Dept. of Computer Science I, University of Bonn

²Dept. of Computer Science, City University of New York,
The Graduate Center

ABSTRACT. We consider the situation where one is given a set S of points in the plane and a collection \mathcal{D} of unit disks embedded in the plane. We show that finding a minimum cardinality subset of \mathcal{D} such that any path between any two points in S is intersected by at least one disk is NP-complete. This settles an open problem raised in [1]. Using a similar reduction, we show that the Multiterminal Cut Problem introduced in [4] remains NP-complete when restricted to unit disk graphs.

1. INTRODUCTION AND MAIN RESULT

In this note we show that the Point Set Isolation Problem defined below in Problem 1 is NP-complete. This problem was introduced in [1] where a polynomial-time constant-factor approximation algorithm was presented, but the problem complexity was stated as an open problem. As a motivation for studying this problem, its relevance to trap coverage in sensor networks is mentioned, where one wants to detect certain spacial transitions among the observed objects (see for example [3]).

In order to show NP-completeness of the Point Set Isolation Problem we are going to reduce the Planar Subdivision Problem defined in Problem 2 to it. This problem is NP-complete by Proposition 3.

Problem 1 (Point Set Isolation Problem [1]). *Given a set S of k points in the plane and a collection \mathcal{D} of n unit disks embedded in the plane, no disk containing a point of S . The goal is to find a minimum cardinality subset $\mathcal{D}' \subseteq \mathcal{D}$, s.t. every path between two points in S is intersected by at least one disk in \mathcal{D}' .*

Problem 2 (Planar Subdivision Problem). *Given a simple unweighted planar graph $G = (V, E)$ embedded in the plane and a set S of k points properly contained in the faces of G with no face containing more than one point, find the minimum cardinality set $E' \subseteq E$ such that in the embedding of the reduced graph $G' = (V, E')$, no two points are contained in the same face.*

^{*}Research supported by NSF grant 1017539

Proposition 3. *The Planar Subdivision Problem is NP-complete if k is not fixed.*

Theorem 4. *The Point Set Isolation Problem is NP-complete if k is not fixed.*

Problem 5 (Multiterminal Cut Problem [4]). *Given a simple graph $G = (V, E)$ and a set $S \subseteq V$ of k terminals, the task is to find the minimum cardinality set $E' \subseteq E$ such that in $G' = (V, E \setminus E')$ there is no path between any two nodes in S .*

Theorem 6. *The Multiterminal Cut Problem remains NP-complete on unit disk graphs if k is not fixed.*

For a high level description of the proof of Theorem 4, we reduce an instance $I_2 = (G_2, S_2)$ of the Planar Subdivision Problem in polynomial time to an instance $I_1 = (\mathcal{D}, S_1)$ of the Point Set Isolation Problem. We do this by first transforming the embedding of G_2 to an "equivalent" straight line embedding on an integer grid. Each embedded edge then gets replaced by an edge gadget which consists of a path of unit disks constructed in such a way that every edge gadget contains the same amount of unit disks, regardless of the length of the embedded edge. The dimensions of each edge gadget is chosen such that no two unit disks of different edge gadgets intersect. Furthermore, we replace each embedded vertex v by a vertex gadget which consists of a cycle of unit disks which is circularly arranged around v . Each edge gadget of edges incident to v will intersect a small number of disks contained in the vertex gadget. The main task of the reduction is to choose the radius of the disks and the dimension of the gadgets such that every edge gadget consists of the same amount of disks and so that non-incident edge gadgets are disjoint. For the proof of Theorem 6 similar gadgets are used.

REFERENCES

- [1] Gibson, Matt and Kanade, Gaurav and Varadarajan, Kasturi, On Isolating Points using Disks, In ESA'11 2011.
- [2] Dahlhaus, E. and Johnson, D. S. and Papadimitriou, C. H. and Seymour, P. D. and Yannakakis, M., *The Complexity of Multiterminal Cuts*, SIAM J. Comput. 1994.
- [3] Paul Balister, Zizhan Zheng, Santosh Kumar, and Prasun Sinha. Trap coverage: Allowing coverage holes of bounded diameter in wireless sensor networks. In *In Proc. of IEEE INFOCOM, Rio de Janeiro*, 2009.
- [4] Dahlhaus, E. and Johnson, D. S. and Papadimitriou, C. H. and Seymour, P. D. and Yannakakis, M. The Complexity of Multiterminal Cuts. *SIAM J. Comput.*, 1994.

Inapproximability of the Smallest Superpolyomino Problem*

Andrew Winslow[†]
awinslow@cs.tufts.edu

1 Introduction

We consider the *smallest superpolyomino problem*: given a set of colored polyominoes, find the smallest superpolyomino containing each input polyomino as a subpolyomino. Alternatively, find an overlapping arrangement of the polyominoes such that all overlapping cells have matching colors and the union of the polyominoes is as small as possible.

In one dimension, this problem is equivalent to the *smallest superstring problem* and admits a greedy constant-factor approximation algorithm [1]. Charikar et al. [2] use this to develop a straightforward $O(\log^3 n)$ -approximation algorithm for finding the smallest context-free grammar encoding a string.

One motivation for investigating the smallest superpolyomino problem is the possibility of extending the Charikar algorithm to higher dimensions, yielding good grammar-based image and shape compression algorithms. Here we show that such an extension is unlikely to exist by proving that the smallest superpolyomino problem is NP-hard to approximate within a $O(n^{1/3-\varepsilon})$ -factor for any $\varepsilon > 0$ by a reduction from chromatic number.

2 Definitions

A polyomino $P = (S, L)$ is defined by a connected set of points S on the square lattice (called *cells*) containing $(0, 0)$, and a coloring of the cells, e.g. cell $(3, 1)$ is red, cell $(3, 2)$ is gray, etc. We denote the color of the cell (x, y) as $P(x, y)$, and $|P|$ denotes the number of cells in P , i.e. the *size* of P . Two polyominoes $P_u = (S_u, L_u)$ and $P_v = (S_v, L_v)$ at some translation (δ_x, δ_y) are *compatible* if for each (x, y) , either $P_v(x, y)$ or $P_u(x, y)$ is empty or $P_v(x, y) = P_u(x + \delta_x, y + \delta_y)$. Similarly, a polyomino $P = (S, L)$ is a *superpolyomino* of $P' = (S', L')$ if there exists a translation (δ_x, δ_y) such that for each (x, y) , either $(x, y) \notin S'$ or $P'(x + \delta_x, y + \delta_y) = P(x, y)$, i.e. there is a translation of P' such that P' is compatible with P and lies entirely in P .

3 Reduction

Given a graph $G = (V, E)$, each vertex $v \in V$ is converted into a polyomino $P_v = (S_v, L_v)$ that encodes v and the neighbors of v in G (see Figure 1). Each P_v is a rectangular $2|V| \times |V|$ polyomino with up to $|V| - 1$ single squares removed and lower-left corner at $(0, 0)$. The four corners of all P_v have a common set of four colors: green, blue, purple, and orange. Cells at locations $\{(2i + 1, 1) \mid 0 \leq i < |V|\}$ are colored black if $v_i = v$, red if $(v, v_i) \in E$, or are empty locations if v_i is not v or a neighbor of v . All remaining cells have a common gray color.

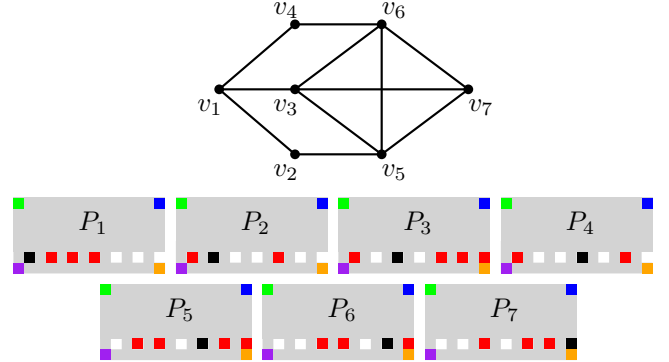


Figure 1: An example of the set of polyominoes generated from an input graph by the reduction.

Consider how two polyominoes P_u and P_v can overlap, depending upon the relationship of u and v . Because of the four distinct corner colors, P_u and P_v can only overlap when these four locations in P_v are translated to the same locations in P_u . In this translation, the cells at location $(2i + 1, 1)$ in P_u and P_v are compatible exactly when $(u, v) \notin E$, i.e. u and v are not neighbors. All other cells are colored gray and thus compatible.

The superpolyomino formed by a pair of compatible P_u and P_v in this translation has the common set of four colored corner cells and many gray cells, and has two black cells and a number of red cells corresponding to the combined neighborhoods of u and v . Then by induction, any set of polyominoes can overlap if and only if they form an independent set. Moreover, if they overlap, they overlap using a set of translations in which

*A full version of this paper is available at <http://arxiv.org/abs/1210.3877>.

[†]Department of Computer Science, Tufts University. Research supported in part by NSF grants CCF-0830734 and CBET-0941538.

the four corners of all polyominoes are placed at four common locations.

Because the polyominoes can only overlap in this constrained way, any superpolyomino of the polyominoes $\{P_v \mid v \in V\}$ consists of a number of *decks* of superimposed polyominoes corresponding to independent sets of vertices in G arranged disjointly to form a single connected polyomino (see Figure 2).

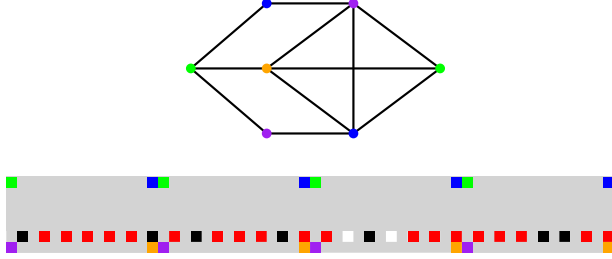


Figure 2: An example of a corresponding 4-deck superpolyomino and 4-colored graph.

Recall that each P_v is a $2|V| \times |V|$ rectangle with $|V|$ cells colored black, red, or are not present. The size of P_v is then between $2|V|^2 - |V| + 1$ and $2|V|^2$ depending upon the number of neighbors of v , and each deck of polyominoes also has size in this range.

Lemma 3.1 *For a graph $G = (V, E)$, there exists a superpolyomino of size at most $2k|V|^2$ for polyominoes $\{P_v \mid v \in V\}$ if and only if the vertices of V can be k -colored.*

Proof First, consider extreme sizes of superpolyominoes consisting of k and $k - 1$ decks. For any V and k with $1 \leq k \leq |V|$, $(k - 1)(2|V|^2) = 2k|V|^2 - 2|V|^2 < 2k|V|^2 - k|V| = k(2|V|^2 - |V|)$, i.e. the size of any superpolyomino of $k - 1$ decks is smaller than the size of any superpolyomino of k decks.

We now prove both implications of the lemma. First, assume the superpolyomino of size at most $2k|V|^2$ exists. Then the superpolyomino must consist of at most k decks. Each deck is the superposition of a set of polyominoes forming an independent set, so G can be k -colored.

Next, assume that G can be k -colored. Then the polyominoes $\{P_v \mid v \in V\}$ can be translated to form k decks, one for each color, each with size at most $2|V|^2$. Placing these decks adjacent to each other yields a superpolyomino of size at most $2k|V|^2$. \square

Note that only $|V|$ cells of each P_v are distinct and depend on v , while the other $2|V|^2 - |V|$ are held constant. The extra cells are needed for the first inequality in Lemma 3.1, and they effectively “drown out” the difference in sizes of various decks due to the number of cells not present in each deck.

Theorem 3.2 *The smallest superpolyomino problem is NP-hard to approximate within a factor of $O(n^{1/3-\epsilon})$ for any $\epsilon > 0$.*

Proof Consider the smallest superpolyomino problem for the polyominoes generated from a graph $G = (V, E)$ with chromatic number k . There are $|V|$ of these polyominoes, each of size $\Theta(|V|^2)$, so the polyominoes have total size $n = \Theta(|V|^3)$. By Lemma 3.1, a superpolyomino of size between $(2|V|^2 - |V|)k'$ and $2|V|^2k'$ exists if and only if there exists a k' -coloring of G . Then by Zuckerman [3], finding a superpolyomino such that $(2|V|^2 - |V|)k' / (2|V|^2)k = O(|V|^{1-\epsilon}) = \Theta(n^{1/3-\epsilon})$ is NP-hard.

As seen in Figure 3, the result also holds when constrained to sets of polyominoes using at most two colors by converting each cell into a unique 8×8 macro-cell.

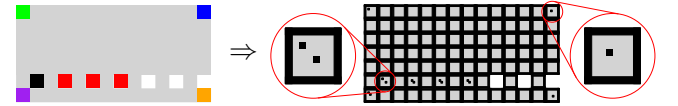


Figure 3: Converting a reduction polyomino (left) to a two-color reduction polyomino (right).

We mention (but do not prove here) that the problem constrained to single-color sets of polyominoes is NP-hard by a reduction from set cover. An example of a polyomino set used in the reduction is seen in Figure 4.



Figure 4: The set of polyominoes produced from the reduction from minimum set cover to smallest superpolyomino for the set $\{\{1, 2\}, \{1, 4\}, \{2, 3, 4\}, \{2, 4\}\}$.

References

- [1] A. Blum, T. Jiang, M. Li, J. Tromp, M. Yannakakis, Linear approximation of shortest superstrings, *Journal of the ACM*, 41(4) (1994), 630–647.
- [2] M. Charikar, E. Lehman, A. Lehman, D. Liu, R. Panigrahy, M. Prabhakaran, A. Sahai, a. shelat, The smallest grammar problem, *IEEE Transactions on Information Theory*, 51(7) (2005), 2554–2576.
- [3] D. Zuckerman, Linear degree extractors and the inapproximability of max clique and chromatic number, *Theory of Computation*, 3 (2007), 103–128.

A Characterization Theorem and an Algorithm for a Convex Hull Problem

Bahman Kalantari

Extended Abstract. Given a set $S = \{v_1, \dots, v_n\} \subset \mathbb{R}^m$ and a point $p \in \mathbb{R}^m$, testing if $p \in \text{conv}(S)$, the convex hull of S , is a fundamental problem in computational geometry and linear programming. Denoting the Euclidean distance between $u, w \in \mathbb{R}^m$ by $d(u, w) = \sqrt{\sum_{i=1}^m (u_i - w_i)^2}$, first we prove a *distance duality*:

Distance Duality

Precisely one of the two conditions is satisfied:

- (i): For each $p' \in \text{conv}(S) \setminus \{p\}$, there exists $v_j \in S$ such that $d(p', v_j) \geq d(p, v_j)$;
- (ii): There exists $p' \in \text{conv}(S)$ such that $d(p', v_i) < d(p, v_i)$, for all $i = 1, \dots, n$.

Condition (i) is valid if and only if $p \in \text{conv}(S)$, and condition (ii) if and only if $p \notin \text{conv}(S)$. Utilizing this duality, we describe a simple fully polynomial time approximation scheme, called the *Triangle Algorithm*:

Triangle Algorithm ($S = \{v_1, \dots, v_n\}, p$)

- **Step 1.** Given $p' \in \text{conv}(S) \setminus \{p\}$, check if there exists $v_j \in S$ such that $d(p', v_j) \geq d(p, v_j)$. If no such v_j exists, stop, $p \notin \text{conv}(S)$.
- **Step 2.** Otherwise, on the line segment joining p' to v_j compute the point nearest to p . Denote this by p'' . Replace p' with p'' , go to Step 1.

We refer to p' in Step 1 as *iterate* and v_j as *pivot point*. Given $\epsilon \in (0, 1)$, the Triangle Algorithm in at most $48mn\epsilon^{-2} = O(mn\epsilon^{-2})$ arithmetic operations computes a point $p' \in \text{conv}(S)$ such that either

$$d(p', p) \leq \epsilon d(p, v_j), \quad \text{for some } j; \text{ or} \quad (1)$$

$$d(p', v_i) < d(p, v_i), \quad \forall i = 1, \dots, n. \quad (2)$$

We refer to the point p' satisfying (1) as an ϵ -*approximate solution*. Clearly, approximation to a prescribed absolute error is also possible. We refer to a point p' satisfying (2) as *witness*. This condition holds if and only if $p \notin \text{conv}(S)$. This is because in this case we can prove the Voronoi cell of p' with respect to the two point set $\{p, p'\}$ contains $\text{conv}(S)$ (see Figure 1). Equivalently, the orthogonal bisector of the line segment pp' separates p from $\text{conv}(S)$.

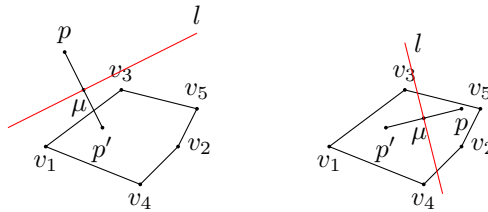


Figure 1: Example of cases where orthogonal bisector of pp' does and does not separate p from $\text{conv}(S)$.

The set W_p of all such witnesses is the intersection of $\text{conv}(S)$ and the open balls, $B_i = \{x \in \mathbb{R}^m : d(x, v_i) < d(p, v_i)\}$, $i = 1, \dots, n$. W_p is a convex open set in the relative interior of $\text{conv}(S)$ (see Figure 2).

By squaring the distances, $d(p', v_j) \geq d(p, v_j) \iff d(p', 0)^2 - d(p, 0)^2 \geq 2v_j^T(p' - p)$. Thus Step 1 does not require taking square-roots. Also, the computation of p'' in Step 2 requires no square-root operations.

Given a point $p' \in \text{conv}(S)$ that is not a witness, having $d(p, p')$ as the current *gap*, the Triangle Algorithm moves to a new point $p'' \in \text{conv}(S)$ where the new gap $d(p, p'')$ is reduced. We will prove that when $p \in \text{conv}(S)$, the number of iterations K_ϵ , needed to get an approximate solution p' satisfying (1) is bounded above by $48\epsilon^{-2} = O(\epsilon^{-2})$. In the worst-case each iteration of Step 1 requires $O(mn)$ arithmetic operations.

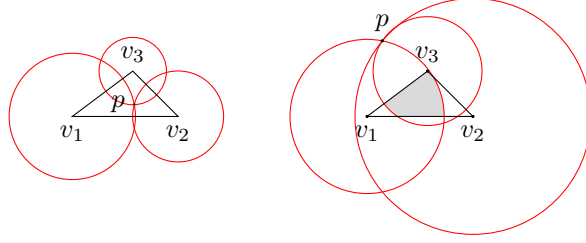


Figure 2: Examples of empty W_p ($p \in \text{conv}(S)$) and nonempty W_p ($p \notin \text{conv}(S)$), gray area.

However, it may also take only $O(m)$ operations. The number of arithmetic operations in each iteration of Step 2 is only $O(m)$. Thus the complexity for computing an ϵ -approximate solution is $O(mn\epsilon^{-2})$ arithmetic operation. In particular, for fixed ϵ the complexity of the algorithm is only $O(mn)$.

When $p \notin \text{conv}(S)$, the Triangle Algorithm does not attempt to compute the closest point to p , say $p_* \in \text{conv}(S)$, rather a separating hyperplane. However, by virtue of the fact that it finds a hyperplane orthogonally bisecting the line pp' , it in the process computes an approximation to $d(p, p_*)$ to within a factor of two. More precisely, any witness p' satisfies the inequality

$$.5d(p, p') \leq d(p, p_*) \leq d(p, p'). \quad (3)$$

Not only this approximation is useful for the convex hull problem, but for computing the distance between two convex hulls, the *polytope distance* problem. As is well known the Minkowski difference of two convex hulls is a polytope whose shortest vector has norm equal to the distance between the two polytopes.

The justification in the name of the algorithm lies in the fact that in each iteration the algorithm searches for a triangle $\triangle pp'v_j$ where $v_j \in S$, $p' \in \text{conv}(S) \setminus \{p\}$, such that $d(p', v_j) \geq d(p, v_j)$. Given that such triangle exists, it uses v_j as a pivot point to “pull” the current iterate p' closer to p to get a new iterate $p'' \in \text{conv}(S)$.

We also show how to solve general LP via the Triangle Algorithm and give a corresponding complexity analysis. In particular, we prove a sensitivity theorem that converts LP feasibility with bounded domain into a convex hull problem, then gives the necessary accuracy for computing an ϵ -approximate solution. We also contrast the theoretical performance of the Triangle Algorithm with the *sparse greedy approximation* (equivalent to Frank-Wolfe and Gilbert algorithms) for the minimization of a convex quadratic over a simplex, a problem arising in machine learning, approximation theory, and statistics. The bibliography contains sample references from the main article.

References

- [1] K. L. Clarkson. Coresets, Sparse Greedy Approximation, and the Frank-Wolfe algorithm. In SODA '08: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms, 922 - 931. Society for Industrial and Applied Mathematics, 2008.
- [2] M. Frank and P. Wolfe, An algorithm for quadratic programming, *Naval Res. Logist. Quart.*, 3 (1956), 95 - 110.
- [3] B. Gärtner and M. Jaggi, Coresets for polytope distance, Symposium on Computational Geometry (2009), 33 - 42.
- [4] E. G. Gilbert, An iterative procedure for computing the minimum of a quadratic form on a convex set, *SIAM Journal on Control* Volume 4 (1966), 61 - 80.
- [5] B. Kalantari, Finding a lost treasure in convex hull of points from known distances. In the Proceedings of the 24th Canadian Journal of Computational Geometry (2012), 271 - 276.
- [6] T. Zhang, Sequential greedy approximation for certain convex optimization problems, IEEE Trans. Information Theory, 49 (2003), 682 - 691.

Department of Computer Science, Rutgers University, Piscataway, NJ 08854
kalantari@cs.rutgers.edu

Approximation Algorithms for Outlier Removal in Convex Hulls

Michael Biro ^{*} Justine Bonanno^{*} Roozbeh Ebrahimi [†] Lynda Montgomery^{*}

Abstract

Given n points in \mathbb{R}^2 , we give approximation algorithms to find a subset of k of the points that has minimum-area or minimum-perimeter convex hull. We give algorithms that, for each k , yield a constant-factor approximation to the minimum-perimeter problem in linear time. We also show a 2-approximation for the minimum-area problem in time $O(\min(n^3 \log n, n^2 \log n + kn(n-k)(n-k + \log k)))$, as well as a heuristic for both problems that appears to work well in practice.

1 Introduction

The problem of finding a subset of size k from a point set of size n that has the least-perimeter convex hull was considered in several papers, going back to 1983, with results improving from the original $O(k^2 n \log n + k^5 n)$ in Dobkin et al. [8] to $O(n \log n + k^3 n)$ in Datta et al. [7] and Eppstein et al. [4]. Finding the subset of k points with the minimum-area convex hull was considered in Eppstein [3], and Eppstein et al. [5], where they give $O(kn^3)$ and $O(n^2 \log n + k^3 n^2)$ exact algorithms. These algorithms give the exact solution, but their runtimes can be $\Omega(n^4)$ and $\Omega(n^5)$, respectively for perimeter and area, and for large k . Recently, for $k = n - c$, Atanassov, et al, [2] gave exact $O(n \log n + \binom{4c}{2c}(3c)^{c+1}n)$ algorithms for both problems, however this still leaves a difficulty of finding exact solutions for k sufficiently far from n . We give a linear-time, constant-factor approximation to the minimum-perimeter problem, as well as a 2-approximation algorithm for the minimum-area problem that runs in $O(\min(n^3 \log n, n^2 \log n + kn(n-k)(n-k + \log k)))$ time. In addition, we describe a heuristic for choosing the outliers that seems to work well in practice.

2 Minimum-Perimeter Convex Hull

We approximate the k -outlier minimum-perimeter convex hull by approximating the shape of the convex hull as either a rectangle or circle.

Lemma 2.1. *Let P be a convex set in \mathbb{R}^2 . Then the perimeter of P is at most a factor $\sqrt{2}$ away from the perimeter of the minimum-perimeter axis-parallel rectangle containing P .*

Lemma 2.2. *Let P be a convex set in \mathbb{R}^2 . Then the perimeter of P is at most a factor of $\frac{\pi}{2}$ away from the perimeter of the minimum disc enclosing P .*

We now use recent results of Ahn et al. [1], that finds the minimum-perimeter axis-parallel rectangle in time $O(n + k^3)$, and results of Har-Peled et al. [9] that finds a $(1 + \epsilon)$ -approximation to the minimum enclosing disk in time $O(n + n \cdot \min(\frac{1}{k\epsilon^2} \log^2(\frac{1}{\epsilon}), k))$. Then we have,

Theorem 2.3. *The k -outlier minimum-perimeter convex hull problem can be approximated by a factor of $\sqrt{2}$ in time $O(n + k^3)$, and a factor of $\frac{\pi}{2}(1 + \epsilon)$ in time $O(n + n \cdot \min(\frac{1}{k\epsilon^2} \log^2(\frac{1}{\epsilon}), k))$.*

These algorithms give constant-factor approximations to the minimum-perimeter problem for a variety of values of k . If $k = O(n^{1/3})$ then the rectangle algorithm yields a $\sqrt{2}$ approximation in linear time, and if $k = \Omega(n^{1/3})$, then the disk algorithm gives a $\frac{\pi}{2}(1 + \epsilon)$ -approximation in linear time (for constant values of ϵ). Therefore, for each value of k , we give constant-factor approximations to the k -outlier minimum-perimeter problem that run in linear time.

Corollary 2.4. *For each k , the k -outlier minimum-perimeter convex hull problem can be approximated by a constant factor in linear time.*

3 Minimum-Area Convex Hull

We approximate the k -outlier minimum-area convex hull problem by approximating the shape of the convex hull as a rectangle with arbitrary orientation.

^{*}Dept. Applied Mathematics and Statistics, Stony Brook University, mbiro@ams.stonybrook.edu, justine.bonanno@stonybrook.edu, lynda.montgomery@stonybrook.edu

[†]Dept. Computer Science, Stony Brook University, rebrahimi@cs.stonybrook.edu

Lemma 3.1. *Let P be a convex set in \mathbb{R}^2 . Then the area of P is at most a factor of 2 away from the area of the minimum-area rectangle enclosing P .*

Proof. Take the longest diagonal D of P , and construct a minimal enclosing rectangle R with two sides parallel to D . Take R along with the D and the two points defining the perpendicular edges to D of R . The area of P is at least the area of the two triangles thus defined, and the two triangles take up exactly half of R . Therefore, R has area at most twice the area of P , and as the minimum-area rectangle has area at most the area of R , it has area at most twice the area of P . [10] \square

Using the idea in the above proof, we construct an algorithm that checks every possible longest diagonal D of the point set P , then computes the minimum-area rectangle containing at least k points among all such diagonals, in time $O(n^3 \log n)$.

1. Examine every pair of points (p, q) and look at the strip defined by lines perpendicular to \overline{pq} through p and q respectively. Find the points of P that lie in the strip.
2. Sort the points in the strip by distance of \overline{pq} , and for every point in the strip find the corresponding point so that the rectangle defined by the four points contains k points of P .
3. Take the minimum-area rectangle among all rectangles constructed.

We combine this with the recent result of Das et al. [6], that finds a minimum-area rectangle in time $O(n^2 \log n + kn(n-k)(n-k + \log k))$ time.

Theorem 3.2. *The k -outlier minimum-area convex hull problem can be 2-approximated in time $O(\min(n^3 \log n, n^2 \log n + kn(n-k)(n-k + \log k)))$.*

This approximation is useful if k is $\Theta(n)$, so the optimal solution in [3] runs in time $\Omega(n^5)$, but not $n - c$ for constant c , as the optimal solution in [2] runs in time exponential in c .

4 Heuristic

In this section, we describe a heuristic for the minimum-area convex hull problem. It runs in time $O(n(n-k) \log n)$ in the worst case, and while there are cases where it gives arbitrarily bad approximations, in practice it has yielded good results.

1. Find the diameter d of the points, say between points a and b . Define a lune L by intersecting disks of radius d centered at a and b , respectively.

2. Let O be the midpoint of \overline{ab} . Divide L evenly into 4 equal sectors around O and associate the points with their respective sector. Sort the points with respect to distance from O .
3. If any sector contains fewer than $n - k$ points, remove all points in that sector.
4. Remove points in order of decreasing distance from O until either $n - k$ points are removed, or one of a, b is removed. If the latter, begin again.

5 Conclusion

For future work, we would like to find a PTAS for minimum-perimeter that runs in linear or near-linear time, as well as improve the running time of the approximation for minimum-area convex hulls.

References

- [1] H. Ahn, S. Bae, E. Demaine, M. Demaine, S. Kim, M. Korman, I. Reinbacher, W. Son. Covering points by disjoint boxes with outliers. *Computational Geometry*, 44(3):178-190, 2011.
- [2] R. Atanassov, P. Bose, M. Couture, A. Maheshwari, P. Morin, M. Paquette, M. Smid, S. Wührer. Algorithms for optimal outlier removal. *J. Discrete Algorithms*, 7(2):239-248 2009.
- [3] D. Eppstein. New algorithms for minimum area k -gons. *Proceedings of the 3rd ACM-SIAM Symp. Discrete Algorithms*, pages 83-88, 1992.
- [4] D. Eppstein, J. Erickson. Iterated nearest neighbors and finding minimal polytopes. *Discrete & Computational Geometry*, 11(1):321-350, 1994.
- [5] D. Eppstein, M. Overmars, G. Rote, G. Woeginger. Finding minimum area k -gons. *Discrete & Computational Geometry*, 7:45-58, 1992.
- [6] S. Das, P. Goswami, S. Nandy. Smallest k -point enclosing rectangle and square of arbitrary orientation. *Information Processing Letters*, 94(6):259-266, 2005.
- [7] A. Datta, H. P. Lenhof, C. Schwarz, M. Smid. Static and Dynamic Algorithms for k -Point Clustering Problems. *J. Algorithms*, 19(3):474-503, 1995.
- [8] D. Dobkin, R. Drysdale, L. Guibas. Finding smallest polygons. *Computational Geometry*, 1:181-214, 1983.
- [9] S. Har-Peled, S. Mazumdar. Fast algorithms for computing the smallest k -enclosing disc. *Proceedings of the 11th Annual European Symposium on Algorithms*, pages 278-288, 2003.
- [10] A. King (mathoverflow.net/users/4580), Area ratio of a minimum bounding rectangle of a convex polygon, <http://mathoverflow.net/questions/93528> (version: 2012-04-08)

The Range 1 Query (R1Q) Problem

Rezaul Chowdhury*, Pramod Ganapathi†, Yuan Tang‡

Abstract: Given a bit array of size n and two indices i and j , find efficiently if there is a 1 in the subarray $[i, j]$. We call this problem the "Range 1 Query" (R1Q) problem. The problem can be easily generalized to higher dimensions. We give exact and approximation algorithms to solve the problem in 1D and higher dimensions. We can also answer queries for right triangles, isosceles triangles and in general, polygons with certain constraints. The applications include the Pochoir stencil compiler [1], where we have to answer octagonal queries to select optimized code clones.

Exact Algorithm: The 1D algorithm is as follows. We divide the input array into blocks of size 2^p for different values of p greater than some threshold. We preprocess these blocks so that any query of length exactly 2^p that crosses a block boundary or lies completely inside a block can be answered in constant time. To answer intra-block queries, we use Four Russians trick.

To answer the query $\text{R1Q}(i, j)$, we find at most two possibly overlapping blocks of size $2^k \leq j - i + 1$ (for some positive integer k) that completely cover the range, and then compute the answer from the two queries. For input size of n bits, preprocessing takes $o(n)$ bits of space and $O(n)$ time and query execution takes $O(1)$ time. The algorithm can be extended to higher dimensions.

Approximation Algorithm: As in the exact algorithm, we preprocess the blocks of size 2^p for different values of p , and this time we use Cormode-Muthukrishnan's Count Min (CM) sketch data structure [2] to store the preprocessed data. We create separate CM sketches for different block sizes so that queries of different sizes can be answered approximately in sublinear space and constant time.

By setting the value of the error rate, the space usage of the approximation algorithm can be made arbitrarily small compared to the exact algorithm. The query execution is similar to the exact algorithm.

Other Shapes: We answer right triangular queries as follows. We preprocess all the right triangles with horizontal base in eight orientations, with either base

*Stony Brook University, U.S.A.; rezaul@cs.stonybrook.edu

†Stony Brook University, U.S.A.; pganapathi@cs.stonybrook.edu

‡Fudan University, China; yuantang@fudan.edu.cn

length or height a power of two. Now we split a general horizontal base right triangle into two overlapping right triangles with base length or height a power of two, and possibly a rectangle. We find the answers to these triangles and rectangle using the preprocessed data and hence answer the query triangle.

We answer polygonal queries as follows. The polygon is split into a collection of right triangles and rectangles that completely lie inside the given polygon and completely cover it, and each of which can be answered using preprocessed data. The results of these right triangular and rectangular queries are combined to answer the input polygonal query.

References

- [1] Y. Tang, R. Chowdhury, B. Kuszmaul, C. K. Luk, C. Leiserson. *The Pochoir Parallel Stencil Compiler*. Proceedings of the SPAA, pp. 117-128, 2011.
- [2] G. Cormode, S. Muthukrishnan. *An Improved Data Stream Summary: The Count-Min Sketch and Its Applications*. Journal of Algorithms, vol. 55, pp. 58-75, 2005.

Group Following in Monotonic Tracking Regions *

Christopher Vo

Jyh-Ming Lien †

Abstract

For a 2-d pursuit-evasion game called group following, we study a data structure called *monotonic tracking regions* (MTR). An MTR has a support path so that the points along the path can collectively see every point in the MTR. An MTR can be considered as a generalization of a star-shaped region. Using an MTR, we can plan online the path of a bounded-speed camera tracking a group of n agents to a planning horizon h in time $O(hn^2)$.

1 Introduction

Monotonic tracking regions (MTRs) [Vo and Lien 2010] are a data structure to guide a single camera following multiple coherent targets in a 2D workspace. Intuitively, in an MTR, the camera can maintain visibility of targets by moving along a trajectory. This MTR data structure allows us to generate the camera's motion online by solving a *linear programming problem*.

We assume that the camera C has a bounded linear velocity v_C^{max} . The exact configuration of this view at time t , denoted as $\mathcal{V}_C(t)$, is defined by the camera's view direction $\theta_C(t)$ and location $x_C(t)$. The position of the camera is simply: $x_C(t + \Delta t) = x_C(t) + \Delta t \cdot v_C(t)$, where $v_C(t)$ is the camera's velocity at time t . The target T comprises a group of coherent *non-adversarial* members, whose trajectories are not known in advance. We also assume that the size of T and T 's maximum (linear) velocity v_T^{max} are known (by the camera). The position of $x_T(t)$ a target $\tau \in T$ at time t is known only if τ is visible by the camera. We attempt to maximize the number of visible targets:

$$\arg \max_{v_C(t)} \left(\sum_t \text{card}(\{T' \subset T \mid X_{T'}(t) \subset \mathcal{V}_C(t)\}) \right),$$

where $\text{card}(\mathcal{X})$ is the number of elements in \mathcal{X} .

2 Monotonic Tracking Regions (MTRs)

We let a 2D region \mathcal{M}_π be a generalized cylinder defined w.r.t a *supporting path* π . We say π is a supporting path of \mathcal{M}_π if every point $x \in \mathcal{M}_\pi$ can see a subset of π . Consequently, the visibility of π spans \mathcal{M}_π .

Definition 2.1. $\mathcal{M}_\pi \subset \mathcal{F}$ is a region supported by a path π if $\mathcal{M}_\pi = \{x \mid \exists y \in \pi \text{ s.t. } \overline{xy} \subset \mathcal{F}\}$, where \overline{xy} is an open line segment between x and y , and \mathcal{F} is the free space (i.e., the area without obstacles).

Furthermore, we define the subset of π visible by x as: $\mathcal{V}_\pi(x) = \{y \in \pi \mid \overline{xy} \subset \mathcal{F}\}$. Finally, we define MTR.

*This work is supported in part by NSF IIS-096053, NSF EFRI-1240459, AFOSR FA9550-12-1-0238.

†Both authors are with Department of Computer Science, George Mason University, Fairfax, VA 22030 USA.

Definition 2.2. A region $\mathcal{M}_\pi \in \mathcal{F}$ is an MTR supported by π if $|\mathcal{V}_\pi(x)| = 1, \forall x \in \mathcal{M}_\pi$, where $|\mathcal{X}|$ is the number of connect components in a set \mathcal{X} .

Because each $x \in \mathcal{M}_\pi$ can see only an interval of π , we can compactly represent the visible region (called visibility interval) of x as a tuple $\mathcal{V}_\pi(x) = (s, t), 0 \leq s \leq t \leq 1$, if we parameterize π from 0 to 1.

2.1 Follow a single target

Let $x_\tau(t)$ be the position of the target τ at time t . Since we know the maximum speed of the target, we can estimate the positions $x_\tau(t + \Delta t)$ in the next time step, i.e., the intersection of \mathcal{F} and a disc with radius $\Delta t \cdot v_T^{max}$. In order to keep the target in the view, the camera's next position $x_C(t + \Delta t)$ must be:

$$x_C(t + \Delta t) \in \mathcal{V}_\pi(x_\tau(t + \Delta t)) = \bigcap_{x \in x_\tau(t + \Delta t)} \mathcal{V}_\pi(x).$$

Let $I_i = \mathcal{V}_\pi(x_\tau(t + i \cdot \Delta t)) = (s_i, t_i)$. Here i is an integer from 1 to h , where h is the user-defined time horizon. Both s_i and t_i are parameters on the parameterized path π . In order to follow the target for h steps, the planner needs find a sequence of parameterized camera locations x_i from a sequence of intervals such that every point x_i is in its corresponding interval I_i . In addition, one may desire to minimize the distance travelled by the camera. Taking all these into consideration, this problem can be formulated as an h -dimensional linear programming (LP) problem:

$$\begin{aligned} \min \quad & x_h - x_1 \\ \text{s.t.} \quad & s_i \leq x_i \leq t_i, \forall i \in \{1, 2, \dots, h\} \end{aligned}$$

We call the above LP problem the *canonical following problem*. Solving a canonical following problem can be done efficiently since h is usually small (≤ 20).

2.2 Follow multiple targets

Now, we will extend the canonical following problem to handle multiple targets T . Let $x_T(t)$ be the current positions of the targets T . Similar to the case of a single target, we estimate the positions $x_T(t + \Delta t)$ in the next time step. In order to see a least one target, the camera must move so that

$$x_C(t + \Delta t) \in \bigcup_{\tau \in T} \mathcal{V}_\pi(x_\tau(t + \Delta t)) = \bigcup_{\tau \in T} \left(\bigcap_{x \in x_\tau(t + \Delta t)} \mathcal{V}_\pi(x) \right).$$

To simplify our notation, let $I_i = \bigcup_{\tau \in T} \mathcal{V}_\pi(x_\tau(t) + i \cdot \Delta t) = (s_i, t_i)$. By placing the camera in I_i , we can guarantee that at least one target is visible. However, our goal is to maximize the number of visible targets, at least over the planning horizon. To do so, we segment I_i into subintervals I_i^j , each of which can see n_i^j

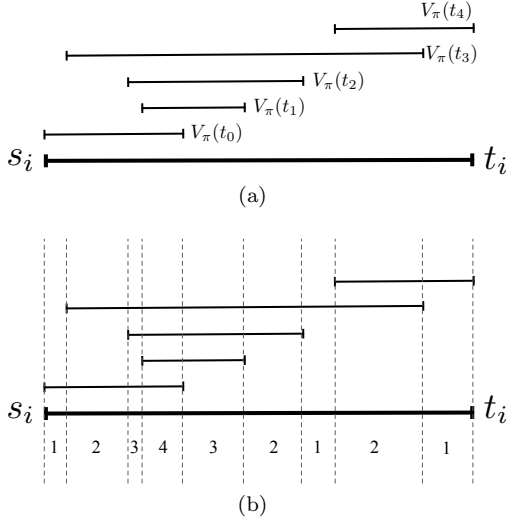


Figure 1: (a) The interval $I_i = (s_i, t_i) = \cup_{\tau \in T} \mathcal{V}_\pi(x_\tau)$. (b) The interval I_i is segmented into 9 subintervals, each of which is a set of points in π that can see the same number of targets, which is shown below each interval.

targets. Fig. 1(a) shows an example of I_i defined as the union of all the visibility intervals $\mathcal{V}_\pi(x_\tau)$ of the targets τ . Note that I_i may contain multiple connected components. Fig. 1(b) shows the subdivision of I_i (i.e., subintervals I_i^j) bounded by the end points of $\mathcal{V}_\pi(x_\tau)$. Each I_i^j is associated with the number of visible targets n_i^j . When the velocity of the camera is unlimited, then the optimal strategy is to pick the subinterval I_i^j with the largest n_i^j in each I_i , i.e., I_i is shrunk to I_i^j . Thus, instead of solving the following problem using I_i , the subintervals I_i^j will be used. See Fig. 2.

From Fig. 2, one can also see that the distance that the camera has to travel from x_2 to x_3 is quite long, thus the camera will need to move very fast to maintain the maximum visibility. When the camera speed is bounded, this may not always be possible. Therefore, we need a way to select a subinterval from each I_i so that the total number of visible targets is maximized while still maintaining the constraint that the minimum distance between $I_i^j \subset I_i$ and $I_{i+1}^k \subset I_{i+1}$ is smaller than $\Delta t \cdot v_T^{max}$. More specifically, we would like to find a solution to the following problem:

$$\arg \max_{\{j_i\}} \left(\sum_{i=1}^h n_i^{j_i} \right) \text{ s.t. } \text{dist}(I_i^{j_i}, I_{i+1}^{j_{i+1}}) \leq \Delta t \cdot v_T^{max}, \forall i,$$

where j_i is the index of the j_i -th subinterval in interval I_i , and $\text{dist}(x, y)$ is the closest distance between two subintervals x and y . Although, at the first glance, this problem seems to be another LP problem, fortunately, Lemma 2.3 shows that the optimal subintervals can be found in $O(hn^2)$ time, where n is the number of targets and h is the time horizon.

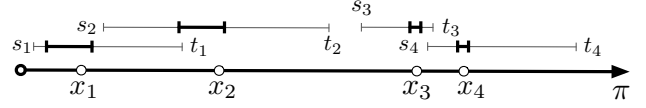


Figure 2: Make predictions in subintervals with maximum targets visibility for the next $h = 4$ future steps.

Lemma 2.3. Finding all $I_i^{j_i}$ will take $O(hn^2)$ time for n targets and h planning time horizon.

Proof. The main idea is to construct a directed graph from these subintervals and the current position of the camera, and show that this graph must be a DAG with $O(hn)$ vertices and $O(hn^2)$ edges. Then the problem of finding a sequence of optimal subintervals becomes the longest path search problem in the DAG, which can be solved in time linear to the size of the graph.

To construct such a graph, we first define the idea of *reachability*. Given two subintervals u and v from two consecutive intervals, the reachable interval $r(u, v) \in v$ is a set of points in v that the camera can reach from u in one step without violating the speed constraint. If $r(u, v)$ is not empty, then we say v is reachable from u . Note that the reachability can be nested, i.e., given three subintervals u , v , and w , we say that w is reachable from u if $r(u, w) = r(r(u, v), w)$ is not empty. In the graph that we will construct, we ensure that every node in the graph is reachable from the current position x_C of the camera. Finally, we say that a subinterval v is reachable by the camera if $r(x_C, v)$ is not empty.

Specifically, we let the current position x_C of the camera be the source of the graph and let the subintervals be the rest of the nodes in the graph. The source are then connected to the subintervals in I_1 that are reachable by the camera. The each reachable subinterval in I_1 are connected to the subintervals in I_2 that are reachable by the camera. The process repeats until the reachable intervals in I_h are connected by those in I_{h-1} . Note that since we only need to pick one subinterval from each interval, the subintervals within each interval are not connected. Finally, we let the edge weight be the number of visible targets in the destination node. The graph constructed this way must be a DAG since there is no back edge. Any path that connects the source to a sink will contain a sequence of valid subintervals. Thus, finding the maximum number of targets visible from these subintervals is equivalent to finding the longest path in the DAG, which can be solved in linear time using topological sort. Since each interval will have $\Theta(2n)$ subintervals and two consecutive intervals will have $4n^2$ edges, this DAG has $O(hn)$ vertices and $O(hn^2)$ edges. \square

References

VO, C., AND LIEN, J.-M. 2010. Following a large unpredictable group of targets among obstacles. In *The Third International Conference on Motion in Games*.

Guarding simple polygons with semi-open edge guards ^{*}

Asish Mukhopadhyay [†]

Chris Drouillard [†]

Godfried Toussaint [‡]

1 Introduction

Let $bd(P)$ denote the boundary of a simple polygon P . Points p and q of P are mutually visible if segment \overline{pq} lies entirely inside P . This notion of visibility gave birth to an extensive literature on art-gallery problems [3], concerned with guarding the floor of a polygonal art-gallery. Chvatal [2] showed that $\lfloor n/3 \rfloor$ point guards are always sufficient and sometimes necessary. Allowing guards to move on an edge gives rise to the class of edge guard problems. An edge guard is closed (open) if the end-points of the edge are included (excluded), semi-open if only one end-point is included. Shermer [5] established an upper bound of $\lfloor 3n/10 \rfloor + 1$ on the number of closed edge guards needed, while Toussaint [3] showed that $\lfloor n/4 \rfloor$ guards are sometimes necessary. A *guard edge* is one that guards all of P . In [6], Toth et al. have shown that a non star-shaped simple polygon has at most one open guard edge. Park et al. [4] showed that such a polygon can have at most 3 closed guard edges.

Thus it is interesting to explore the scenario in which the edge guards are semi-open. A semi-open edge guard includes exactly one of the end-points. For clarity and focus, in this paper the included end-point is always the end that is met first in a clockwise traversal of P . We show that a non star-shaped polygon has at most 3 semi-open guard edges and propose an $O(n)$ algorithm to find all semi-open guard edges of a polygon.

2 Semi-open guard edges

Lemma 1 *Let $e = (u, v]$ be a semi-open edge of a polygon P and p a point interior to it. Then p is visible from e iff the set of common vertices of the paths $p \rightsquigarrow u$ and $p \rightsquigarrow v$ is either $\{p\}$ or $\{p, v\}$.*

Figure 1 shows that a non star-shaped polygon can have 3 semi-open guard edges. In fact, the following result holds.

Theorem 1 *Every non star-shaped simple polygon has at most three semi-open guard edges.*

^{*}The full content of this abstract together with the proofs of all the results will be published in the Proceedings of the Third International Conference on Digital Information Processing and Communications, Islamic Azad University (IAU), Dubai, United Arab Emirates, Jan. 30, 2013 - Feb. 1, 2013.

[†]School of Computer Science, University of Windsor, {asish.mukerji, beaner.drouillard}@gmail.com

[‡]New York University, Abu Dhabi, gt42@nyu.edu

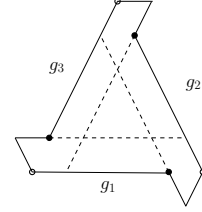


Figure 1: A non-starshaped polygon that has three semi-open guard edges : g_1, g_2 and g_3

3 Characterizing semi-open guard edges

Let r be a reflex vertex of P . With respect to a counter-clockwise order of $bd(P)$, let r^- be the vertex that precedes r on $bd(P)$, and r^+ the one that succeeds it. Let p^- be the intersection with $bd(P)$ of a ray shot from r in the direction $\overrightarrow{r^-r}$, while p^+ is the intersection with a ray shot from r in the direction $\overrightarrow{r^+r}$.

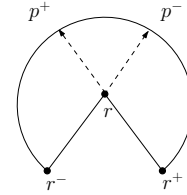


Figure 2: Two subpolygons defined by a reflex vertex r

These rays define two polygons: a left polygon $C_{left}(r)$ bounded by the chord rp^- and the part of $bd(P)$ from p^- to r in the counter-clockwise order; and a right polygon $C_{right}(r)$ bounded by the chord rp^+ and the part of $bd(P)$ from r to p^+ in the counter-clockwise order.

The left (respectively, right) kernel, $K_{left}(P)$ (respectively, $K_{right}(P)$) is the intersection of all the left (respectively, right) polygons $C_{left}(r)$ (respectively, $C_{right}(r)$), while the kernel of P is the intersection of $K_{left}(P)$ and $K_{right}(P)$.

Fact 1 *The kernels $K_{left}(P)$ and $K_{right}(P)$ are both convex.*

Toth et al. [6] showed that the left and right kernels can be used to define left-kernel and right-kernel decomposi-

tions of $\text{int}(P)$. These decompositions were used to prove the following theorem.

Theorem 2 *A open edge $e = (a, b)$ of a simple polygon P is a guard edge iff e intersects both the left and right kernels of P .*

Theorem 3 *A semi-open edge $e = (a, b]$ is a guard edge iff e has a non-empty intersection with $C_{\text{left}}(r) \cap C_{\text{right}}(r)$ for every reflex vertex, r .*

4 Algorithm

In [1], Bhattacharya et al. proposed a linear time algorithm for computing a shortest internal line segment l from which a polygon P is weakly internally visible. Central to their algorithm is the notion of a non-redundant component. Both $C_{\text{left}}(r)$ and $C_{\text{right}}(r)$, as defined in this paper, are components of P . A component is non-redundant if it does not properly contain any other component. They show how to compute all non-redundant components in linear time.

Let the ends of each non-redundant component that lie on $bd(P)$ be marked blue and red in counter-clockwise order in an initial counter-clockwise traversal of $bd(P)$.

To calculate the number of non-redundant components that an edge intersects, we obtain this value for the previous edge, subtract the number of red marks it contains, and then add the number of blue marks the current edge contains. To initialize the process, we find the number of non-redundant components for the first edge. This requires an extra pass over $bd(P)$ with a counter initialized to 0; every red mark passed over decrements the counter, and every blue mark increments it. This finds all closed guard edges. To narrow down to just semi-open guard edges, we remove all edges with both end points reflex.

At the end of the second round, we declare those edges e as semi-open guard edges whose intersection count $\text{edge-Count}(e)$ is equal to the number of non-redundant components.

5 Polygons with holes

A polygon P with holes can have guard edges that lie on the outer boundary or on the boundaries of the holes. Park et al. [4] have shown that to establish upper bounds it is enough to consider polygons with only one convex hole, indeed just one triangular hole. It is quite obvious that no semi-open edge of this triangular hole can be an guard edge as it cannot see all the points on its own boundary. As for guard edges on the outer boundary, the following theorem of [4] for closed guard edges carries over when the guard edges are semi-open.

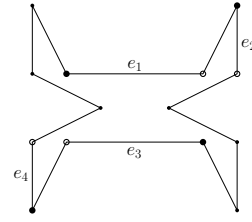


Figure 3: Semi open edges $e_1 - e_4$ guard this polygon

Theorem 4 *For a polygon P with a convex hole H , the number of guard edges is at most 3.*

6 Conclusion

By considering semi-open guard edges, we are led to some interesting conclusions. The upper bound on the number of semi-open guard edges is the same as for closed guard edges. A more careful characterization is needed for a semi-open guard edge as one or both the kernels can be empty. It would also be interesting to find tight upper and lower bounds on the number of semi-open edge guards needed to guard a polygon P . The classes of polygons in Figures 3, 4 seem to suggest a lower bound of $2n/7$ semi-open edge guards.

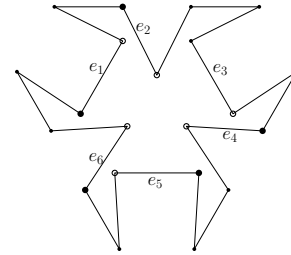


Figure 4: Semi open edges $e_1 - e_6$ guard this polygon

References

- [1] B. K. Bhattacharya, G. Das, A. Mukhopadhyay, and G. Narasimhan. Optimally computing a shortest weakly visible line segment inside a simple polygon. *Comput. Geom.*, 23(1):1–29, 2002.
- [2] V. Chvátal. A combinatorial theorem in plane geometry. *Journal of Combinatorial Theory, Series B*, 18(1):38–41, 1975.
- [3] J. O’Rourke. *Art Gallery Theorems and Algorithms*, volume 31. Oxford University Press, 1987.
- [4] J.-H. Park, S. Y. Shin, K.-Y. Chwa, and T. C. Woo. On the number of guard edges of a polygon. *Discrete & Computational Geometry*, 10:447–462, 1993.
- [5] T. Shermer. Recent results in art galleries. *Proceedings of IEEE*, 80:1384–1399, 1992.
- [6] C. Toth, G. Toussaint, and A. Winslow. Open guard edges and edge guards in simple polygons. In *Canadian Conference on Computational Geometry, Toronto, ON, Canada, Aug 10-12, 2011*.

Fence patrolling by mobile agents with distinct speeds

Akitoshi Kawamura

University of Tokyo

Yusuke Kobayashi

University of Tokyo

This is based on a paper with the same title [2] to be presented at ISAAC 2012.

Fence patrolling and the partition-based strategy

Suppose we want to patrol a fence (line segment) using k mobile agents with given speeds v_1, \dots, v_k so that every point on the fence is visited by an agent at least once in every unit time period, indefinitely.

Czyzowicz et al. [1] conjectured that the maximum length of the fence that can be patrolled is $(v_1 + \dots + v_k)/2$, which is achieved by the simple *partition-based strategy* where each agent i keeps moving back and forth in a sub-segment of length $v_i/2$.

The partition-based strategy is not always optimal

We [2] disproved the conjecture by the counterexample in the figure below, where six agents with speeds 1, 1, 1, 1, $7/3$, $1/2$ patrol a fence of length $7/2$, beating the partition-based strategy attaining $41/12$. In the figure, time flows upwards and the agents (four in the left diagram, one in the middle, one in the right) move along the solid lines. The regions that have been visited in the past unit time are shaded. Observe that the regions together cover the whole strip. The dotted lines delimit the regions already covered in previous diagram(s).

Theorem 1 *There is a setting of six agents' speeds for which the partition-based strategy does not patrol the longest possible fence.*

We also showed (see the full version) that the conjecture is true for three agents.

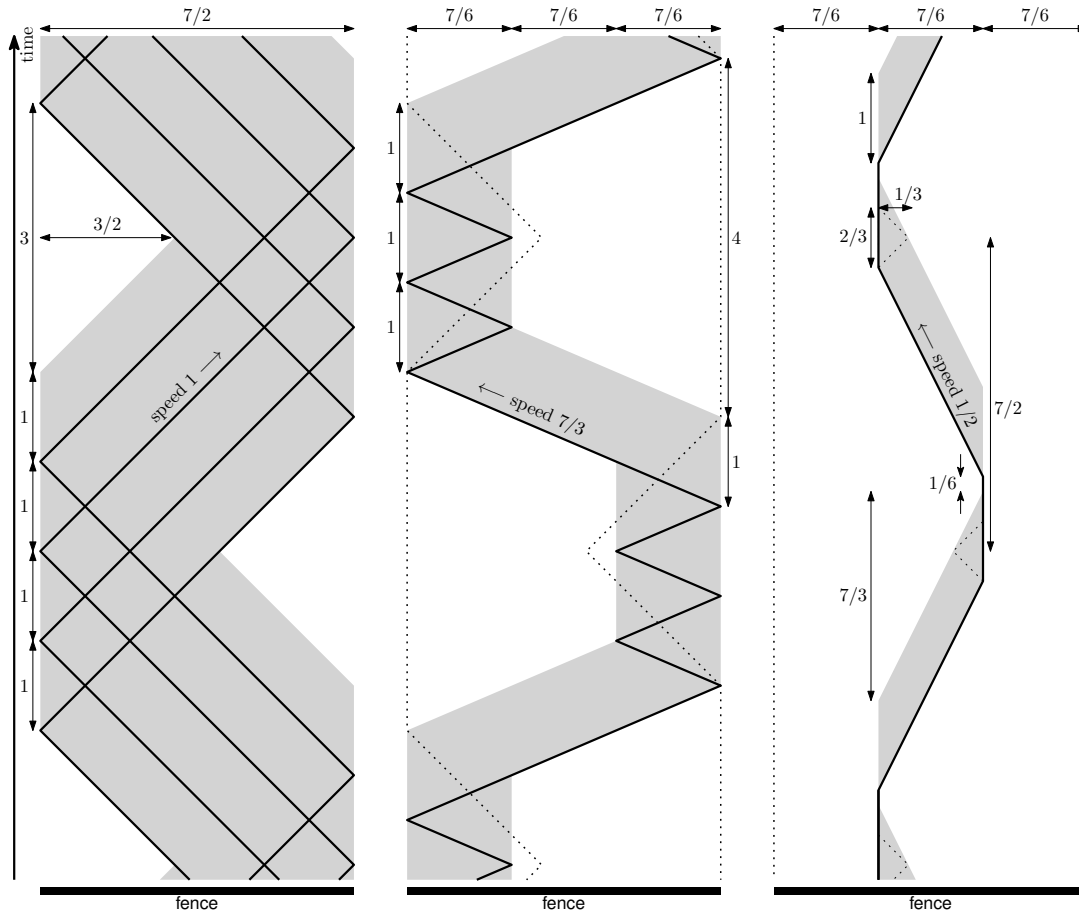
Theorem 4 *For three agents, no strategy patrols a longer fence than the partition-based strategy.*

We do not know whether the conjecture is true for four or five agents.

A revised conjecture

An easy argument about the area of the shaded regions in the diagram shows that no strategy can patrol a fence longer than $v_1 + \dots + v_k$. This upper bound is twice as big as what the partition-based strategy achieves. Our example in the figure outperforms the partition-based strategy, but barely. Thus we ask if we can improve the upper bound, so that a weakened version of Czyzowicz et al.'s conjecture holds:

Open problem. Is there a constant $c < 1$ such that for any k and any v_1, \dots, v_k , the agents cannot patrol a fence longer than $c(v_1 + \dots + v_k)$?



- [1] J. Czyzowicz, L. Gásieniec, A. Kosowski and E. Kranakis. Boundary patrolling by mobile agents with distinct maximal speeds. In *Proceedings of the 19th Annual European Symposium on Algorithms (ESA 2011)*, LNCS 6942, pp. 701–712.
- [2] A. Kawamura and Y. Kobayashi. Fence patrolling by mobile agents with distinct speeds. To be presented at the *23rd International Symposium on Algorithms and Computation (ISAAC 2012)*.

Watchman Paths in Disk Grids

Michael Biro*

Justin Iwerks*

Abstract

We study shortest watchman paths in rectangular arrangements of tangent unit-radius disks with disk centers on a square grid lattice. Upper and lower bounds are given for the length of shortest paths that see all of the boundary of each disk.

1 Introduction

The watchman tour problem in polygons involves finding a shortest tour so that every point in the polygon is seen from at least one point along the route [1, 2, 3]. We explore a related problem, which we will call the DISK GRID PATH PROBLEM: Given a rectangular arrangement of tangent unit-radius disks with disk centers on a square grid lattice, find the length of a shortest watchman path that sees all of the boundary of each disk.

An equivalent statement of the DISK GRID PATH PROBLEM is to find a shortest path with length $L(m, n)$ that travels through each of the $m \times n$ ($m \leq n$) “pockets” defined by the regions around the disks (see Figure 1).

Lemma 1.1. *A watchman path P with minimum length $L(m, n)$ is non-crossing.*

Proof. If P crosses itself, we can reroute the crossing segments locally so that the crossing is eliminated and the path remains connected. This local change yields a watchman path with strictly shorter length. \square

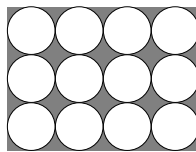


Figure 1: The 3×4 disk grid with 20 shaded pockets

We will assume that for the DISK GRID PATH PROBLEM, a given pocket is never visited twice. We

leave the resolution of this conjecture as an open problem:

Conjecture 1.2 (The simplicity conjecture). *An optimal solution to the DISK GRID PATH PROBLEM never visits the same pocket twice.*

Lemma 1.3. *Given the simplicity conjecture, the DISK GRID PATH PROBLEM is equivalent to finding a MAXIMUM-TURN HAMILTONIAN PATH in an $m \times n$ square grid graph.*

Proof. By the simplicity conjecture, an optimal path visits each vertex of the grid graph once. If the pocket is visited on a turn, the vertex contributes a cost of $\pi/2$ to the path, which is less than its cost of 2 if the pocket was visited on a straight path. There are always two terminating endpoints, so their contributions are equal and have cost 1. Thus, the shortest path is exactly the one with the most turns. \square

Let $T(m, n)$ be the number of turns in a MAXIMUM-TURN HAMILTONIAN PATH of an $m \times n$ square grid graph. Then, the length of the shortest watchman path, $L(m, n)$, is given by

$$\begin{aligned} L(m, n) &= \frac{\pi}{2}T(m, n) + 2 + 2(nm - T(m, n) - 2) \\ &= 2(nm - 1) - (2 - \frac{\pi}{2})T(m, n) \end{aligned}$$

Therefore, in order to find the minimum-length watchman path in disk grids, we bound the number of right-angle turns in a Hamiltonian path on an $m \times n$ grid graph.

2 Results

We summarize our results in Table 1, of lower and upper bounds on $T(m, n)$, $m \leq n$. This, in turn, bounds $L(m, n)$, via the above equation.

Conjecture 2.1. *$T(m, n) = nm - m$ for m, n even.*

2.1 Upper Bounds

Examine two adjacent vertices in the grid graph. There are three types of ways the vertices can be

*Dept. Applied Mathematics and Statistics, Stony Brook University, mbiro@ams.stonybrook.edu, jiwerts@gmail.com

m	n	lower bound	upper bound
odd	$= m$	$nm - n - 1$	$nm - n - 1$
odd	odd \setminus even	$nm - n$	$nm - n$
even	odd	$nm - m$	$nm - m$
even	even	$nm - m$	$nm - 4$

Table 1: Combinatorial bounds for $T(m, n)$

visited. The equivalence classes up to symmetry are in Figure 2 (square vertices are endpoints):

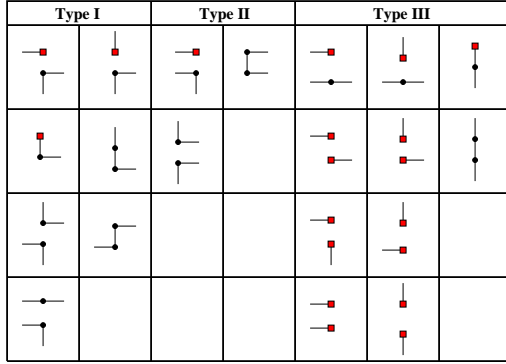


Figure 2: The types of vertex visitations

We divide the grid into horizontal strips of thickness 2, then analyze which vertex types can occur.

Lemma 2.2. *A horizontal strip of the grid graph that contains a right-pointing Type I pair of vertices must contain at least 1 additional non-turn vertex.*

Proof. In all cases, either a non-turn vertex appears, or a new Type I pair arises to the right side of the old pair. A Type I pair cannot appear on the right boundary, so there must be a non-turn vertex. \square

Corollary 2.3. *If a strip contains a Type I pair then it contains at least two non-turn vertices.*

Proof. All Type I pairs either contain a non-turn vertex already, or are symmetric so that there is a non-turn both to the right and to the left of the pair. \square

Lemma 2.4. *A strip with odd width must contain a Type I or Type III pair of vertices. Specifically, such a strip must contain at least 2 non-turn vertices.*

Proof. Each of the Type II vertices involves exactly two pairs of vertices so one pair must be left out. \square

Theorem 2.5. *The upper bounds for $T(m, n)$ in Table 1 hold.*

Proof. Due to space constraints, we only show the case where m is odd and n is even. The remaining cases are similar. Take the $m \times n$ grid graph and cut it into $\frac{n}{2}$, $2 \times m$ strips. Since m is odd, by Lemma 2.4, there is a pair of non-turn vertices in each of the strips, leaving at most $2m - 2$ turns per strip. Thus, there are at most $\frac{n}{2}(2m - 2) = nm - n$ turns in an optimal path. \square

2.2 Lower Bounds

Watchman paths that achieve the number of turns in Table 1 can be constructed (in general) via a spiraling pattern. Figure 3 gives several examples.

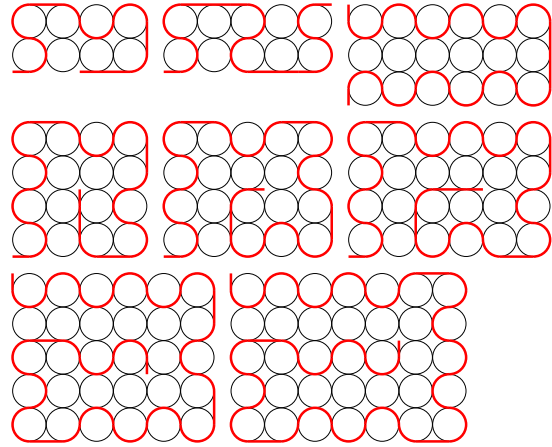


Figure 3: Examples of watchman paths that achieve the lower bounds given in Table 1

3 Conclusion

For future work, we would like to establish tight bounds for the unresolved case when both m and n are even. In addition, we would like to prove the simplicity conjecture for paths and consider the closely related DISK GRID CYCLE PROBLEM.

References

- [1] W. Chin and S. Ntafos. Optimum watchman routes. *Information Processing Letters* 28(1):39-44, 1988.
- [2] A. Dumitrescu and C. D. Tóth. Watchman tours for polygons with holes. *Computational Geometry: Theory and Applications* 45(7):326-333, 2012.
- [3] S. Carlsson, H. Jonsson, and B. J. Nilsson. Finding the shortest watchman route in a simple polygon. *Discrete and Computational Geometry* 22(3):377-402, 1999.

Visibility Problems Concerning One-Sided Segments

Jonathan Lenchner*

Eli Packer†

1 Introduction

We study a family of line segment visibility problems related to classical art gallery problems which are motivated by monitoring and surveillance requirements in commercial data centers. In traditional art gallery problems (see [4], [3] and [8]) an entire polygonal region must be kept under surveillance. In our case it is a prescribed collection of non-overlapping line segments in the interior of the polygon which must be kept under surveillance. Moreover, in many cases, it is required to see just one side of each segment. Some of our early results attacking simple variants of this problem were described in [2].

We consider distinct cases where the segments to be monitored are either all vertical, all axis-aligned, or alternatively, all arbitrarily aligned. Segments are assumed to be non-intersecting. Within these cases we identify several variants of the basic visibility problem. Namely, if visibility must be from a given side, but that side is specified by the problem poser, we say the problem is an instance of the **Poser's Choice** problem. If, on the other hand, the solver has the choice of which side to monitor the segment from, we say that it is an instance of the **Solver's Choice** problem. Variants of the Solver's Choice problem have been studied by Czyzowicz et al. [1], Toth [5] and Urrutia [8]. A final variant is where the solver must monitor the entire segment from both sides (a variant also considered by Toth [5]).

In general we are interested in many aspects of these problems, from solving particular instances exactly, or with some approximation guarantee, to achieving hardness results, or achieving so-called combinatorial bounds, which say that for an arbitrary set of n segments, to see all segments using one of the visibility models may require some number, $f(n)$, cameras. We consider both theoretical cameras with unlimited angular visibility and models of real cameras with some degree of restricted angular visibility or minimum/maximum depth of field restrictions. In [2] we showed that it was NP-hard to solve the Poser's Choice problem for the case of all vertical segments and cameras with limited angle of visibility. This result can be extended to cameras of unlimited angle of visibility in all the variants of the problem we have mentioned.

2 Results

In [2] we established hardness results for problems with either theoretical or realistic cameras. In this abstract we focus our discussion on describing some of what we know about the polynomial bounds for models involving theoretical cameras, i.e. cameras

with unlimited angular visibility and no depth of field constraints. These results are summarized in Figure 1.



	Solver's Choice 	Solver's Choice 	Poser's Choice (All Segments from same specified side)	Poser's Choice	Both Sides
Vertical	U $n/3$ L $n/3$	U $n/3$ L $n/3$	U $n/2$ L $n/2$	U $n/2$ L $n/2$	U $2n/3$ L $2n/3$
Orthogonal	U $n/2$ (C) L $n/3$	U $n/2$ L $n/3$	U $n/2$ L $n/2$	U $3n/4$ L $2n/3$	U $4n/5$ (T) L $2n/3$
Arbitrary	U $n/2$ (T) L $2n/5$ (U)	U $3n/4$ L $2n/5$ (U)	U $3n/4$ L $n/2$	U $3n/4$ L $2n/3$	U $4n/5$ (T) L $4n/5$ (T)

Figure 1. A table summarizing what we know for the various problem variants. The 'U'-prefixed number in each cell denotes the best-known combinatorial upper bound, while the 'L'-prefixed number denotes the best-known lower bound. Results with a following (C) are due to Czyzowicz et al. [1], those with a following (T) are due to Toth [5] or [6], those with a following (U) are due to Urrutia [8], and the unlabeled ones are due to us.

In all cases we are looking for the minimum number of cameras that will suffice to see all segments in the worst case. All results are modulo additive constants. Thus the theoretical upper bounds are equal to the theoretical lower bounds. However, as the reader will undoubtedly notice, in most cases, there is a gap in our knowledge.

The results along the top row of Figure 1, for all vertical segments, were presented in [2], though at that time we did not make the more subtle distinctions between the two types of Solver's Choice and two types of Poser's Choice, so, in effect, only columns two, four and five were considered. As one moves from the top-left of this table to the bottom-right the problems become consistently harder. Thus the number of cameras required to solve cell (i, j) is less than or equal to the number of cameras needed to solve either cell $(i + 1, j)$ or $(i, j + 1) \forall i, j$. Moreover, the same is true for any established upper and lower bounds. Our interest in the subtly different variations, e.g. of Solver's and Poser's Choice, is so that we can try to characterize precisely where the requirement for more cameras comes from as we move from the easier to harder problems. The lower bounds in the table are established by giving specific examples of segment configurations and arguing that (at least) the given number of cameras are required. The upper bounds are obtained by systematically proving that the given number of cameras can always be used to see the requisite number of segments. Additional problem gradations are possible.

Beginning in row 2 of Figure 1, Czyzowicz et al. [1] established the first interesting result: the upper bound of $\frac{n}{2}$ for the case of all axis-aligned segments under Solver's Choice where the Solver can

*IBM T.J. Watson Research Center

†IBM Haifa Research Lab

choose to see some points from one side and some from the other. The argument is an elegant exploitation of the following [7]:

Theorem. (Tutte) *A graph G has a perfect matching iff every subset of vertices S is such that the number of connected components of $G \setminus S$ of odd order is less than or equal to the number of vertices in S .*

The argument begins, WLOG, by extending the segments so that each end is within some common small epsilon of another edge, or the boundary. These segments give rise to a dissection of the original rectangle into “rooms” with tiny passageways between some adjacent pairs. Form a graph where the nodes of the graph are the rooms and there is an edge between nodes if there is a tiny passageway between them. Then use Tutte’s Theorem to get a near-perfect matching of the rooms. Use the near perfect matching to situate a set of $\lceil \frac{n+1}{2} \rceil$ cameras at the passageway to each pair of rooms, which together see all points on each of the needed segments from one side or another.

A more careful, and consistent, camera placement enables one to extend the Czyzowicz et al. argument to give the identical bound for the more constrained Solver’s Choice problem, as well as most-constrained Poser’s Choice problem. For all the problems on orthogonal segments, gaps exist between the best known upper and lower bounds, except in the case of the most constrained Poser’s Choice problem, where a lower bound of $\frac{n}{2}$ is carried over from the case of all vertical segments – just consider n vertical segments all spaced very close to one another and of height $h - \epsilon$ (h being the height of the rectangle), where the poser requires you to see all segments from the left. Cameras can effectively see at most two segments entirely and a tiny bit of any other segment. Hence $\lceil \frac{n+1}{2} \rceil$ cameras are required.

The next interesting case we get to, and the only additional one we will consider in this short article, is that of Poser’s Choice for the axis-aligned case. A simple example, establishing the $\frac{2n}{3}$ lower bound for this problem, pointed out to us four years ago by Toth [6], is given in Figure 2.

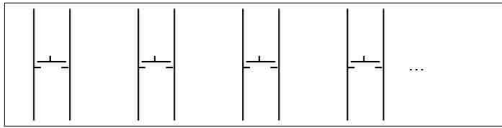


Figure 2. A set of segments for the full Poser’s Choice problem requiring $\frac{2n}{3}$ cameras. The sides of the segments which need to be seen are indicated with little “ticky” marks - i.e. very tiny, orthogonally protruding line segments. The segments in each of the “H”s require two cameras for all of the specified segment sides to be seen entirely.

Finally, a $\frac{3n}{4}$ upper bound is established by virtue of the following:

Theorem. *Given n axis-aligned segments contained in a bounding rectangle, it is always possible to see the Poser’s Choice of sides using at most $\lceil \frac{3n}{4} \rceil$ cameras.*

Proof. (Sketch) Extend the segments as in the Czyzowicz et al. argument, and again use the near-perfect matching to pair up

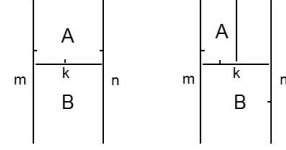


Figure 3. An “H” example (left) and an “h” example (right), in which a single camera cannot see all points on the required segment sides.

rooms. Call matches of the form shown in Figure 3 “bad matches” since a single camera cannot entirely see all the needed segment sides. These are the only cases of two adjoining rooms in which a single camera does not suffice to see all required segment sides. Note that in each of these two examples, one can use *two* cameras to entirely see the needed segment sides in the two rooms, marked respectively A and B in each example. Moreover, it is easy to see that if the three called out lines, m, k, n are part of one bad match, where two cameras must be expended to see all needed segment sides, then they are not part of any additional bad matches. Thus there are at most $n/3$ bad matches in total.

There are then two cases: (i) There are $b \leq \frac{n}{4}$ bad matches, or (ii) there are $\frac{n}{4} < b \leq \frac{n}{3}$ bad matches. In case (i) we use 2 cameras in each bad match and 1 camera in each good match. In case (ii) Suppose there are $\frac{n}{3} - h$ bad matches for $0 \leq h < \frac{n}{12}$. Use 2 cameras to see each of the 3 defining line segments (i.e. the analogs of m, n, k in Figure 3) and 1 camera to see each remaining line segment. In each case a computation shows that we use at most $\frac{3n}{4}$ cameras. \square

References

- [1] J. Czyzowicz, E. Rivera-Campo, J. Urrutia, and J. Zaks. On illuminating line segments in the plane. *Discrete Mathematics*, 137:147–153, 1995.
- [2] R. Das, J. Kephart, J. Lenchner, and E. Packer. Internal surveillance problems in data centers. *Proceedings of the Fall Workshop in Computational Geometry*, 2008.
- [3] D. T. Lee and A. K. Lin. Computational complexity of art gallery problems. *IEEE Trans. Inform. Theory*, 32(2):276–282, 1986.
- [4] J. O’Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, Oxford, U.K., 1987.
- [5] C. Tóth. Illuminating disjoint line segments in the plane. *Discrete and Computational Geometry*, 30:489–505, 2003.
- [6] C. Toth. Personal communication, 2008.
- [7] W. T. Tutte. The factorization of linear graphs. *J. London Math. Soc.*, 22:107–111, 1947.
- [8] J. Urrutia. Art gallery and illumination problems. In J. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 973–1027. Elsevier Science Publishers, Amsterdam, 2000.

Chromatic Clustering in High Dimensional Space

Hu Ding Jinhui Xu

Department of Computer Science and Engineering
State University of New York at Buffalo
{huding, jinhui}@buffalo.edu

1 Overview

Clustering is one of the most fundamental problems in computer science and finds applications in many different areas [2, 3, 5, 6, 11, 14, 15, 17, 19]. Most existing clustering techniques assume that the to-be-clustered data items are independent from each other. Thus each data item can “freely” determine its membership within the resulting clusters, without paying attention to the clustering of other data items. In recent years, there are also considerable attentions on clustering dependent data and a number of clustering techniques, such as correlation clustering, point-set clustering, ensemble clustering, and correlation connected clustering, have been developed [3, 6, 11, 14].

In this paper, we consider the following new type of clustering problems, called *Chromatic Clustering*, for dependent data. Let $\mathcal{G} = \{G_1, \dots, G_n\}$ be a set of n point-sets with each $G_i = \{p_1^i, \dots, p_{k_i}^i\}$ consisting of $k_i \leq k$ points in \mathbb{R}^d space. A chromatic partition of \mathcal{G} is a partition of the $\sum_{1 \leq i \leq n} k_i$ points into k sets, U_1, \dots, U_k , such that each U_i contains no more than one point from each G_j for $j = 1, 2, \dots, n$, where the dimensionality d could be very high. The chromatic k -means clustering (or k -CMeans) of \mathcal{G} is to find k points $\{m_1, \dots, m_k\}$ in \mathbb{R}^d space and a chromatic partition U_1, \dots, U_k of \mathcal{G} such that $\frac{1}{n} \sum_j \sum_{q \in U_j} \|q - m_j\|^2$ is minimized. The problem is called full k -CMeans if $k_1 = k_2 = \dots = k_n = k$. Similarly we can define chromatic k -Median clustering (or k -CMedians) for \mathcal{G} . Chromatic clustering captures the mutual exclusiveness relationship among data items and is a rather useful model for various applications. Due to the additional chromatic constraint, chromatic clustering is thus expected to simultaneously solve the “coloring” and clustering problems, which significantly complicates the problem. We are able to show that the chromatic clustering problem is challenging to solve even for the case that each color is shared only by two data items.

Related works: As its generalization, chromatic clustering is naturally related to the traditional clustering problem. Due to the additional chromatic constraint, chromatic clustering could behave quite differently from its counterpart. For example, the k -means algorithms in [5, 8, 12, 13, 18] relies on the fact that all input points in a Voronoi cell of the optimal k mean points belong to the same cluster. However, such a key locality property no longer holds for the k -CMeans problem.

Chromatic clustering falls in the umbrella of clustering with constraint. For such type of clustering, several solutions exist for some variants [4, 7, 10]. Unfortunately, due to their heuristic nature, none of them can yield quality guaranteed solutions for the chromatic clustering problem. The first quality guaranteed solution for chromatic clustering was obtained recently by Ding and Xu. In [14], they considered a special chromatic clustering problem, where every point-set has exactly k points in the first quadrant, and the objective is to cluster points by cones apexed at the origin, and presented the first PTAS for constant k . The k -CMeans and k -CMedians problems considered in this paper are the general cases of the chromatic clustering problem. Very recently, Arkin *et al.* [1] considered a chromatic 2D 2-center clustering problem and presented both approximation and exact solutions.

1.1 Main Results and Techniques

In this paper, we present three main results, a constant approximation and a $(1 + \epsilon)$ -approximation for k -CMeans and their extensions to k -CMedians.

- **Constant approximation:** We show that given any λ -approximation for k -means clustering, it could yield a $(18\lambda + 16)$ -approximation for k -CMeans. This not only provides a way for us to generate an initial constant approximation solution for k -CMeans through some k -means algorithm, but more importantly reveals the intrinsic connection between the two clustering problems.
- **$(1 + \epsilon)$ -approximation:** We show that a near linear time $(1 + \epsilon)$ -approximation solution for k -CMeans can be obtained using an interesting sphere peeling algorithm. Due to the lack of locality property in k -CMeans, our sphere peeling algorithm is quite different from the ones used in [5, 18], which in general do not guarantee a $(1 + \epsilon)$ -approximation solution for k -CMeans as shown by our first result. Our sphere peeling algorithm is based on another standalone result, called *Simplex Lemma*. The simplex lemma enables us to obtain an approximate mean point of a set of unknown points through a grid inside a simplex determined by some partial knowledge of the unknown point set. A unique feature of the simplex lemma is that the complexity of the grid is *independent of the dimensionality*, and thus can be used to solve problems in high dimensional space. With the simplex lemma, our sphere peeling algorithm iteratively generates the mean points of k -CMeans with each iteration building a simplex for the mean point.
- **Extensions to k -CMedians:** We further extend the idea for k -CMeans to k -CMedians. Particularly, we show that any λ -approximation for k -medians can be used to yield a $(3\lambda + 2)$ -approximation for k -CMedians. With this and a similar sphere peeling technique, we obtain a $(1 + \epsilon)$ -approximation for k -CMedians.

References

1. Esther M. Arkin, Jos Miguel Daz-Bez, Ferran Hurtado, Piyush Kumar, Joseph S. B. Mitchell, Beln Palop, Pablo Prez-Lantero, Maria Saumell, Rodrigo I. Silveira: Bichromatic 2-Center of Pairs of Points. *LATIN 2012*: 25-36
2. David Arthur, Sergei Vassilvitskii: "k-means++: the advantages of careful seeding". *SODA 2007*: 1027-1035
3. Nikhil Bansal, Avrim Blum, Shuchi Chawla: "Correlation Clustering". *Machine Learning* 56(1-3): 89-113 (2004)
4. S.Basu, Ian Davidson: Clustering with Constraints Theory and Practice. *ACM KDD 2006*
5. M.Badouiu, S.Har-Peled, P.Indyk, "Approximate clustering via core-sets", *Proceedings of the 34th Symposium on Theory of Computing*, pp. 250–257, 2002.
6. C.Bhm, K.Kailing, P.Krger, A.Zimek, "Computing Clusters of Correlation Connected Objects". *Proc. ACM SIGMOD International Conference on Management of Data (SIGMOD'04), Paris, France. pp. 455-467. doi:10.1145/1007568.1007620.*
7. Amir Ben-Dor, Ron Shamir, and Zohar Yakhini. Clustering gene expression patterns. *Journal of Computational Biology*, 6(3/4):281-297, 1999.
8. Ke Chen: On Coresets for k-Median and k-Means Clustering in Metric and Euclidean Spaces and Their Applications. *SIAM J. Comput.* 39(3): 923-947 (2009)
9. S. Dasgupta, "The hardness of k-means clustering". *Technical Report*, 2008.
10. Ayhan Demiriz, Kristin Bennett, and Mark J. Embrechts. Semi-supervised clustering using genetic algorithms. *In Artificial Neural Networks in Engineering, pages 809-814. ASME Press, 1999.*
11. Erik Demaine, Dotan Emanuel, Amos Fiat, and Nicole Immorlica. "Correlation clustering in general weighted graphs". *Theor. Comput. Sci.*, 361(2):172-187, 2006
12. Dan Feldman, Michael Langberg: A unified framework for approximating and clustering data. *STOC 2011*: 569-578
13. Dan Feldman, Morteza Monemizadeh, Christian Sohler: A PTAS for k-means clustering based on weak coresets. *Symposium on Computational Geometry 2007*: 11-18
14. H.Ding, J.Xu, "Solving Chromatic Cone Clustering via Minimum Spanning Sphere", *ICALP*, 2011
15. S. Har-Peled and S. Mazumdar, "Coresets for k-Means and k-Median Clustering and their Applications," *Proc. 36th ACM Symposium on Theory of Computing*, pages 291-300, 2004.
16. Mary Inaba, Naoki Katoh, Hiroshi Imai, "Applications of Weighted Voronoi Diagrams and Randomization to Variance-Based K-Clustering (Extended Abstract)". *Symposium on Computational Geometry 1994*: 332-339
17. S. G. Kolliopoulos and S. Rao, "A nearly linear-time approximation scheme for the euclidean k-median problem," *Proc. 7th Annu. European Sympos. Algorithms*, pages 378-389, 1999.
18. A. Kumar, Y. Sabharwal, S. Sen, "Linear-time approximation schemes for clustering problems in any dimensions". *J. ACM* 57(2):2010
19. R.Ostrovsky, Y.Rabani, L.J.Schulman, and C.Swamy. "The Effectiveness of Lloyd-Type Methods for the k-Means Problem". *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*. pp. 165-174.

Kernel Distance for Geometric Inference

Jeff M. Phillips
University of Utah

Bei Wang
University of Utah

This abstract considers geometric inference from a noisy point cloud using the kernel distance. Recently Chazal, Cohen-Steiner, and Mérigot [2] introduced *distance to a measure*, which is a distance-like function robust to perturbations and noise on the data. Here we show how to use the kernel distance in place of the distance to a measure; they have very similar properties, but the kernel distance has several advantages.

- The kernel distance has a small coreset, making efficient inference possible on millions of points.
- Its inference works quite naturally using the super-level set of a kernel density estimate.
- The kernel distance is Lipschitz on the outlier parameter σ .

Kernels, Kernel Density Estimates, and Kernel Distance

A *kernel* is a similarity measure $K : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^+$; more similar points have higher value. For the purposes of this article we will focus on the Gaussian kernel defined $K(p, x) = \sigma^2 \exp(-\|p - x\|^2/2\sigma^2)$.

A *kernel density estimate* represents a continuous distribution function over \mathbb{R}^d for point set $P \subset \mathbb{R}^d$:

$$\text{KDE}_P(x) = \frac{1}{|P|} \sum_{p \in P} K(p, x).$$

More generally, it can be applied to any measure μ (on \mathbb{R}^d) as $\text{KDE}_\mu(x) = \int_{p \in \mathbb{R}^d} K(p, x) \mu(p) dp$.

The *kernel distance* [3, 5] is a metric between two point sets P and Q , or more generally two measures μ and ν (as long as K is positive definite, e.g. the Gaussian kernel). Define $\kappa(P, Q) = \frac{1}{|P|} \frac{1}{|Q|} \sum_{p \in P} \sum_{q \in Q} K(p, q)$. Then the kernel distance is defined

$$D_K(P, Q) = \sqrt{\kappa(P, P) + \kappa(Q, Q) - 2\kappa(P, Q)}.$$

For the kernel distance $D_K(\mu, \nu)$ between two measures μ and ν , we define κ more generally as $\kappa(\mu, \nu) = \int_{p \in \mathbb{R}^d} \int_{q \in \mathbb{R}^d} K(p, q) \mu(p) \nu(q) dp dq$. When the points set Q (or measure ν) is a single point x (or unit Dirac mass at x), then the important term in the kernel distance is $\kappa(P, x) = \text{KDE}_P(x)$ (or $\kappa(\mu, x) = \text{KDE}_\mu(x)$).

Distance to a Measure: A Review

Let S be a compact set, and $f_S : \mathbb{R}^d \rightarrow \mathbb{R}$ be a distance function to S . As explained in [2], there are a few properties of f_S that are sufficient to make it useful in geometric inference such as [1]:

- (F1) f_S is 1-Lipschitz: for all $x, y \in \mathbb{R}^d$, $|f_S(x) - f_S(y)| \leq \|x - y\|$.
- (F2) f_S^2 is 1-semiconcave: the map $x \in \mathbb{R}^d \mapsto (f_S(x))^2 - \|x\|^2$ is concave.

Given a probability measure μ on \mathbb{R}^d and let $m_0 > 0$ be a parameter smaller than the total mass of μ , then the distance to a measure $d_{\mu, m_0} : \mathbb{R}^d \rightarrow \mathbb{R}^+$ [2] is defined for any point $x \in \mathbb{R}^d$ as

$$d_{\mu, m_0}(x) = \left(\frac{1}{m_0} \int_{m=0}^{m_0} (\delta_{\mu, m}(x))^2 dm \right)^{1/2}, \quad \text{where } \delta_{\mu, m}(x) = \inf \{r > 0 : \mu(\bar{B}_r(x)) \leq m\},$$

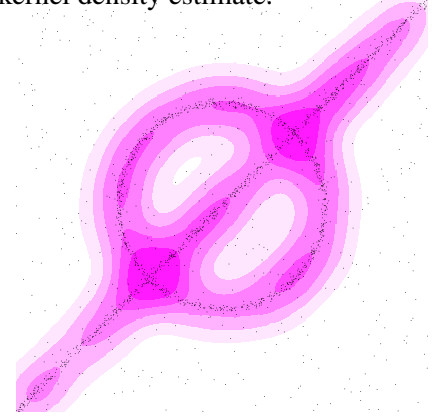


Figure 1: Geometric inference using super-level sets of kernel density estimates on 2000 points.

and where $B_r(x)$ is a ball of radius r centered at x and $\bar{B}_r(x)$ is its closure. It has been shown in [2] using d_{μ, m_0} in place of f_S satisfies (F1) and (F2), and furthermore has the following stability property:

(F3) [Stability] If μ and μ' are two probability measures on \mathbb{R}^d and $m_0 > 0$, then $\|d_{\mu, m_0} - d_{\mu', m_0}\|_\infty \leq \frac{1}{\sqrt{m_0}} W_2(\mu, \mu')$, where W_2 is the Wasserstein distance between the two measures.

Our Results

We demonstrate (with proof sketches) that similar properties hold for the kernel distance defined as $d_P(x) = D_K(P, x)$. These properties also hold on $d_\mu(\cdot) = D_K(\mu, \cdot)$ for a measure μ in place of P .

(K1) d_P is 1-Lipschitz.

This is implied by d_P^2 being 1-semiconcave.

(K2) d_P^2 is 1-semiconcave: The map $x \mapsto (d_P(x))^2 - \|x\|^2$ is concave.

In any direction, the second derivative of $(d_P(x))^2$ is at most that of a single kernel $K(p, x)$ for any p , and this is maximized at $x = p$. The second derivative of $\|x\|^2$ is 2 everywhere, thus the second derivative of $(d_P(x))^2 - \|x\|^2$ is non-positive, and hence is concave.

(K3) [Stability] If P and Q are two point sets in \mathbb{R}^d , then $\|d_P - d_Q\|_\infty \leq D_K(P, Q)$.

Using that $D_K(\cdot, \cdot)$ is a metric, we compare $D_K(P, Q)$, $D_K(P, x)$ and $D_K(Q, x)$. Note: Wasserstein and kernel distance are different *integral probability metrics* [5], so (F3) and (K3) are not comparable.

Advantages of the kernel distance.

- There exists a *coreset* $Q \subset P$ of size $O(((1/\varepsilon)\sqrt{\log(1/\varepsilon\delta)})^{2d/(d+2)})$ [4] such that $\|d_P - d_Q\|_\infty \leq \varepsilon$ and $\|\text{KDE}_P - \text{KDE}_Q\|_\infty \leq \varepsilon$ with probability at least $1 - \delta$. The same holds under a random sample of size $O((1/\varepsilon^2)(d + \log(1/\delta)))$ [3]. In ongoing work, this allows us to operate with $|P| = 100,000,000$. Bottleneck distance between persistence diagrams $d_B(\text{Dgm}(\text{KDE}_P), \text{Dgm}(\text{KDE}_Q)) \leq \varepsilon$ is preserved.
- We can perform geometric inference on noisy P by considering the superlevel sets of KDE_P ; the τ -superlevel set of KDE_P is $\{x \in \mathbb{R}^d \mid \text{KDE}_P(x) \geq \tau\}$. This follows since $d_P(\cdot)$ is *monotonic* with $\text{KDE}_P(\cdot)$; as $d_P(x)$ gets smaller, $\text{KDE}_P(x)$ gets larger. This arguably is a more natural interpretation than using the sublevel sets of some f_S . Figure 1 shows an example with 25% of P as noise.
- Both the distance to a measure and the kernel distance have parameters that control the amount of outliers allowed (m_0 for d_{μ, m_0} and σ for d_P). For d_P the smoothing effect of σ has been well-studied, and in fact $d_P(x)$ is Lipschitz continuous with respect to σ (for σ greater than a fixed constant). Alternatively, $d_{P, m_0}(x)$, for fixed x , is not known to be Lipschitz (for arbitrary P) with respect to m_0 and fixed x ; we suspect that the Lipschitz constant for m_0 is a function of $\Delta(P) = \max_{p, p' \in P} \|p - p'\|$.

References

- [1] Frédéric Chazal, David Cohen-Steiner, and Quentin Mérigot. Boundary measures for geometric inference. *Foundations of Computational Mathematics*, 10(2):221–240, 2010.
- [2] Frédéric Chazal, David Cohen-Steiner, and Quentin Mérigot. Geometric inference for probability measures. *Foundations of Computational Mathematics*, 11(6):733–751, 2011.
- [3] Sarang Joshi, Raj Varma Kommaraju, Jeff M. Phillips, and Suresh Venkatasubramanian. Comparing distributions and shapes using the kernel distance. *Proceedings 27th Annual Symposium on Computational Geometry*, 2011.
- [4] Jeff M. Phillips. eps-samples for kernels. *Proceedings 24th Annual ACM-SIAM Symposium on Discrete Algorithms (to appear)*, 2013.
- [5] Jeff M. Phillips and Suresh Venkatasubramanian. A gentle introduction to the kernel distance. arXiv:1103.1625, March 2011.

A Linear Time Euclidean Spanner On Imprecise Points

Jiemin Zeng Jie Gao
Department of Computer Science
Stony Brook University
Stony Brook, NY 11794, USA
{jiezeng, jgao}@cs.stonybrook.edu

Abstract

An s -spanner on a set of points S in \mathbb{R}^d is a graph on S where for every two points $p, q \in S$, there exists a path between them in G that is less than or equal to $s \cdot |pq|$ where $|pq|$ is the Euclidean distance between p and q . In this paper we consider the construction of an Euclidean spanner for imprecise points. In particular, we are given in the first phase a set of circles with radius r as the imprecise positions of a set of points and we preprocess them in $O(n \log n)$ time. In the second phase, the accurate positions of the points are revealed, one point for every circle, and we construct the $(1 + \varepsilon)$ -spanner in time $O(n \cdot (r + \frac{1}{\varepsilon})^{2d} \cdot \log(r + \frac{1}{\varepsilon}))$. The second phase can also be considered as an algorithm to quickly update the spanner. Our algorithm does not have any restrictions on the distribution of the points. It is the first such algorithm with linear running time.

1 Introduction

While in classical computational geometry all input values are assumed to be accurate, in the real world this assumption does not always hold. This imprecision can be modeled in many ways and they vary based on their applications [6, 8, 10, 11]. A popular model assumes that before the precise input points are known, we know the region (lines [4], circles/balls [1, 3, 7, 9], fat regions [2, 12], etc) where each point lies in.

We aim to compute a $(1 + \varepsilon)$ -spanner on a set of imprecise points. In the model our algorithm

operates in, an imprecise point p is defined to be a disk centered at a point \hat{p} with radius r (which is dependent on the distance between the closest pair of points). The assumption is that initially, the only information of an input point is p and that later, the precise location of the point \hat{p} (located somewhere within p) is revealed. Given a set of n points, an Euclidean spanner defines a graph G on the points, each edge weighted by the Euclidean length, such that the shortest path between any two points u, v in the graph is at most $1 + \varepsilon$ times the Euclidean distance of u, v .

Our algorithm preprocess a set $S = \{p_1, p_2, \dots, p_n\}$ of n imprecise points in $O(n \log n)$ time such that when $\hat{S} = \{\hat{p}_1, \hat{p}_2, \dots, \hat{p}_n\}$ is available, we can compute a $(1 + \varepsilon)$ -spanner in $O(n \cdot (r + \frac{1}{\varepsilon})^{2d} \cdot \log(r + \frac{1}{\varepsilon}))$ time where d is the dimension of the input. It is important to note that our algorithm will accept input sets with overlapping points of any depth.

2 Algorithm Definition

The algorithm we present constructs a $(1 + \varepsilon)$ -spanner or more specifically the deformable spanner (DEFSPANNER) as described in [5]. A DEFSPANNER is a specific $(1 + \varepsilon)$ -spanner construction that is designed to be easily modified and updated. More specifically, for a set of points S in \mathbb{R}^d , the DEFSPANNER is made up of a hierarchy of levels $S_{\lceil \log_2 \alpha \rceil} \subseteq S_{\lceil \log_2 \alpha \rceil - 1} \subseteq \dots \subseteq S_i \subseteq S_{i-1} \subseteq \dots \subseteq S_0 = S$ where $|S_{\lceil \log_2 \alpha \rceil}| = 1$. Any set S_i is a maximal subset of S_{i-1} where for any two points $p, q \in S_i$, $|pq| \geq 2^i$. Also, every node $p \in S_{i-1}$ is assigned a parent node q

in the level above where $|pq| \leq 2^i$. The edges of a DEFSPANNER G of a set S are determined by connecting nodes within distance $c \cdot 2^i$ in level i , where $c = 2r + \frac{16}{\varepsilon} + 4$. Such nodes are called neighbors.

In the preprocessing phase, a DEFSPANNER \hat{G} is constructed with the point set \hat{S} as in [5]. The running time is $O(n \log n)$.

When the true positions of the points are revealed, we construct a DEFSPANNER \hat{G} for \hat{S} by inserting the points one by one into \hat{G} . Initially, nodes are inserted in their top-down order in \hat{G} , but the order may change as the algorithm executes. We insert each node into the bottom level of \hat{G} if other nodes in the same level have been inserted, otherwise a new level is added below. For each node \hat{p} that we insert into level i , there are three phases in our algorithm.

Step 1: Check for demotions. First, we compare the distance between \hat{p} and any of its neighbors that have been already inserted into \hat{G} . If any of the distances are less than 2^i , then we demote the node to a lower level by delaying its insertion.

Step 2: Find a parent. We compare the distances between \hat{p} and its old parent and subsequently the old parent's neighbors. If none of those nodes are a suitable parent, we promote \hat{p} up the hierarchy and repeat our tests with its old grandparent and the old grandparent's neighbors. We repeat and promote \hat{p} until a parent is found or \hat{p} resides at the top of \hat{G} . In certain cases, instead of testing the old parent, we compare distances with an ancestor first or with the node that was considered to be too close (if \hat{p} was demoted).

Step 3: Find all neighbors. We compare distances with \hat{p} and its cousins (\hat{p} 's parent's neighbors' children) in every level it resides in. We also check to see if any of the nodes are too close (less than 2^i in level i). In this case (which only occurs with nodes that have been previously demoted in the first step), we demote the node again.

The algorithm terminates when all nodes have been inserted into \hat{G} . The cost of preprocessing is the DEFSPANNER construction cost or $O(n \log_2 \alpha(r + 1/\varepsilon)^d)$ [5]. The running time of our algorithm is $O(n \cdot (r + \frac{1}{\varepsilon})^{2d} \cdot \log(r + \frac{1}{\varepsilon}))$.

References

- [1] M. A. Abam, P. Carmi, M. Farshi, and M. Smid. On the power of the semi-separated pair decomposition. In *Proceedings of the 11th International Symposium on Algorithms and Data Structures*, WADS '09, pages 1–12, Berlin, Heidelberg, 2009. Springer-Verlag.
- [2] K. Buchin, M. Löffler, P. Morin, and W. Mulzer. Preprocessing imprecise points for delaunay triangulation: Simplified and extended. *Algorithmica*, 61(3):674–693, 2011.
- [3] O. Devillers. Delaunay triangulation of imprecise points, preprocess and actually get a fast query time. *JoCG*, pages 30–45, 2011.
- [4] E. Ezra and W. Mulzer. Convex hull of imprecise points in $o(n \log n)$ time after preprocessing. In *Proceedings of the 27th annual ACM symposium on Computational geometry*, SoCG '11, pages 11–20, New York, NY, USA, 2011. ACM.
- [5] J. Gao, L. J. Guibas, and A. Nguyen. Deformable spanners and applications. In *In Proc. of the 20th ACM Symposium on Computational Geometry (SoCG04)*, pages 179–199, 2004.
- [6] L. Guibas, D. Salesin, and J. Stolfi. Constructing strongly convex approximate hulls with inaccurate primitives. In *Proceedings of the international symposium on Algorithms*, SIGAL '90, pages 261–270, New York, NY, USA, 1990. Springer-Verlag New York, Inc.
- [7] M. Held and J. S. B. Mitchell. Triangulating input-constrained planar point sets. *Inf. Process. Lett.*, 109(1):54–56, Dec. 2008.
- [8] M. Löffler and J. M. Phillips. Shape fitting on point sets with probability distributions. *CoRR*, abs/0812.2967, 2008.
- [9] M. Löffler and J. Snoeyink. Delaunay triangulation of imprecise points in linear time after preprocessing. *Comput. Geom. Theory Appl.*, 43(3):234–242, Apr. 2010.
- [10] T. Nagai, S. Yasutome, and N. Tokura. Convex hull problem with imprecise input. In *Revised Papers from the Japanese Conference on Discrete and Computational Geometry*, JCDCG '98, pages 207–219, London, UK, UK, 2000. Springer-Verlag.
- [11] D. Salesin, J. Stolfi, and L. Guibas. Epsilon geometry: building robust algorithms from imprecise computations. In *Proceedings of the fifth annual symposium on Computational geometry*, SCG '89, pages 208–217, New York, NY, USA, 1989. ACM.
- [12] M. van Kreveld, M. Löffler, and J. S. B. Mitchell. Preprocessing imprecise points and splitting triangulations. *SIAM J. Comput.*, 39(7):2990–3000, June 2010.

An Improved Algorithm in Shortest Path Planning for a Tethered Robot

Ning Xu*, Peter Brass†, Ivo Vigan‡

1 Introduction

There is vast amount of literature focusing on motion planning for general robots. However, the same studies for tethered robots have not been investigated much. While a robot navigates in an environment with obstacles, it may meet some problems, such as a lack of power supply, or losing its wireless communication connection. If a robot is attached to a flexible tether, it can obtain sufficient power supply and stable communication through the tether.

Following [1], the shortest path planning problem for a tethered robot can be described as follows. Let E be a planar environment which consists of disjoint polygonal obstacles of n total vertices, s be the start point, and t be the destination point in the environment. Suppose a robot, modeled as a point, is attached to an anchor point u by a tether of length L . The initial configuration of the tether is considered as a polyline X of k total vertices from s to u . The goal is to find the shortest path from s to t subject to the tether length constraint.

In this paper, we assume that (1) neither the robot nor any part of the tether can enter the interior of any obstacle, and (2) the robot can cross the tether, i.e., the tether can be self intersecting.

First, we consider two different models of the shortest path planning problem. In the first model, the tether is automatically retracted and is kept taut, i.e., the tether is always the shortest path in its homotopy equivalent class. In the second model, the tether can only be retracted while the robot back-

tracks along the tether, because the tether may be too heavy to be dragged in some case.

Figure 1 illustrates an example of the two models.

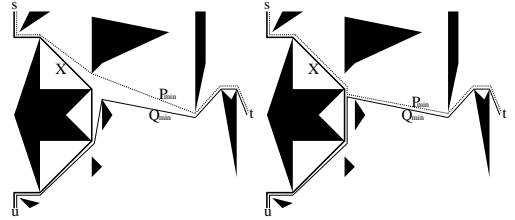


Figure 1: An example of finding the shortest path for the tethered robot. P_{min} is the shortest path, and Q_{min} is the tether configuration after the robot reaches t through P_{min} . (left) the first model; (right) the second model.

Furthermore, we consider the problem of finding the shortest path for the tethered robot to visit a sequence of target points in order. We call this problem the *sequential monitoring problem*.

Theorem 1. *If the tether is automatically retracted, the shortest path from the starting point s to the destination point t can be computed in $O(kn^2 \log n)$ time.*

Theorem 2. *If the tether can only be retracted while the robot backtracks along the tether, the shortest path from the starting point s to the destination point t can be computed in $O((k+n) \log(k+n) + n^2)$ time.*

Theorem 3. *The sequential monitoring problem is NP-hard when the number of target points is $O(n)$, even if the initial tether length is zero, and all obstacles are rectilinear polygons.*

The proof of Theorem 3 is omitted in the abstract.

*Graduate Center, CUNY, nxu@gc.cuny.edu

†City College of New York, CUNY, peter@cs.ccny.cuny.edu

‡Graduate Center, CUNY, ivigan@gc.cuny.edu

2 The Algorithm

Let P be a path, we denote by \bar{P} be the reverse path of P . If the path Q starts where the path P ends, we denote $P \circ Q$ the concatenation of P and Q .

In the first model, the tether is automatically retracted and is kept taut.

We refer to the destination point t , all bending points on X and all obstacle points as *terminals*. For a terminal v , a point c on X is an *event point* if v becomes visible at c while one moves along X from u to s . For a terminal v and an event point c , we denote by $\mathcal{P}_{c,v}$ (or $\mathcal{Q}_{c,v}$) the shortest path homotopy equivalent to the path which is the concatenation of the subpath of X from s (or u) to c and line segment (c, v) , respectively. We also denote by $SP(v, t)$ the shortest path from s to t . Figure 2 illustrates an example.

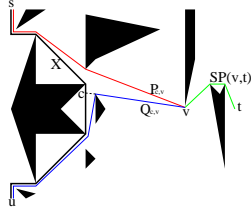


Figure 2: An example of an event point. c is an event point with respect to the terminal v . The red, blue and green path represents $\mathcal{P}_{c,v}$, $\mathcal{Q}_{c,v}$ and $SP(v, t)$ respectively.

The algorithm to find the shortest path in the first model is described in the Algorithm 1.

In the second model, the tether can only be retracted while the robot backtracks along the tether.

A point on X is *feasible* if the robot can leave from X at this point and arrives t subject to the tether length constraint.

For an obstacle point v , a point c on X is a *candidate point* if v is visible from c , and the path concatenated by the subpath of X from s to c , (c, v) and the shortest path from v to t has the length L .

The algorithm to find the shortest path in the second model is described in the Algorithm 2.

References

- [1] P. G. Xavier. Shortest path planning for a tethered robot or an anchored cable. In *ICRA*, pages 1011–1017, 1999.

Algorithm 1: Algorithm SP1

```

1 Triangulate the environment  $E$ ;
2 Compute  $H(X)$  and  $H(\bar{X})$  and store the funnels
  associated with each triangle on the sleeve;
3 Construct the Euclidean shortest path map from  $t$  to
  every terminal and compute the shortest paths;
4 foreach terminal  $v$  do
5   Partition  $X$  into a set of subpaths, such that each
   subpath is concave with respect to  $v$ ;
6   Compute all event points on  $X$  and associate these
   points to the subpaths which contain them;
7   foreach subpath do
8     Use binary search to find the last event point  $c$ 
     on the subpath with the longest  $\mathcal{Q}_{c,v}$  subject to
     that  $\mathcal{Q}_{c,v} \circ SP(v, t)$  is no longer than  $L$ ;
9     Compute  $\mathcal{P}_{c,v} \circ SP(v, t)$ , and compare its length
     with the best solution found so far. Set the best
     solution to  $\mathcal{P}_{c,v} \circ SP(v, t)$  if it is shorter;
10  end
11 end

```

Algorithm 2: Algorithm SP2

```

1 Construct the Euclidean shortest path map from  $t$  to
  every terminal and compute the shortest paths;
2 If  $s$  is a feasible point, directly return the path
   $X \circ SP(s, t)$  as the shortest path;
3 Use binary search to find the line segment of  $X$  that
  contains the last feasible point;
4 foreach obstacle point  $v$  do
5   Check whether there exists a candidate point with
   respect to  $v$ ;
6   Compare with the candidate points obtained before,
   choose the point is farther to  $u$  along  $X$ ;
7 end
8 Choose the path that the robot leaves from  $X$  at the best
  candidate point;

```

Bounded Stretch Homotopic Routing Using Hyperbolic Embedding of Sensor Networks

Kan Huang*, Chien-Chun Ni[†], Rik Sarkar[‡], Jie Gao[†] and Joseph S. B. Mitchell*

*Department of Applied Mathematics and Statistics, Stony Brook University. {khuang, jsbm}@ams.stonybrook.edu

[†]Department of Computer Science, Stony Brook University. {chni, jgao}@cs.stonybrook.edu

[‡]Institut Für Informatik, Freie Universität Berlin, Germany. sarkar@inf.fu-berlin.de

In this paper we consider lightweight routing in a wireless sensor network deployed in a complex geometric domain Σ with holes. Our goal is to find short paths of different *homotopy types*, i.e., paths that go around holes in different ways. In the example of Figure 1, there are three holes in the network and there are many different ways to “thread” a route from s to t . Observe that paths α, β, γ are all different in a global sense; in that, e.g., one cannot deform α to β without “lifting” it over some hole. In contrast, paths γ and δ are only different in a local manner; one can deform γ to δ continuously through local changes, keeping δ within the domain. This difference is characterized by the *homotopy type* of a path. Two paths in a Euclidean domain are *homotopy equivalent* if one can continuously deform one to the other. A set of paths that are pairwise homotopy equivalent are said to have the same homotopy type. The number of homotopy types is infinitely many (assuming there is at least one hole), as a path can loop around a hole k times, for any integer k ; however, for most routing scenarios we only care about a finite number of homotopy types, corresponding to paths in the dual graph of a triangulation of Σ that do not repeat triangles (and thus do not loop around holes).

One heuristic algorithm gives a heuristic path for a given case. The ratio between the length of the heuristic path and the length of the optimal path is defined as the *stretch* of this algorithm for the case.

We introduce a routing framework that guarantees constant worst case stretch for a given homotopy type. We assume that the network is deployed in a geometric domain that is represented by a polygon Σ . We decompose the domain Σ into a triangulation, by including certain diagonals connecting vertices of the polygon Σ . The corners of each triangle are stored locally, only at the nodes that are inside this triangle, along with the corners of the (at most 3) triangles that are adjacent to it. The dual graph of the triangulation is a planar graph \mathcal{D} .

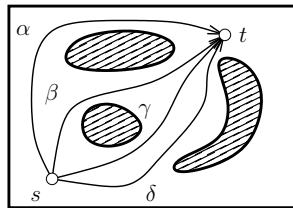


Fig. 1. The network has 3 holes (shaded). Paths α, β, γ have distinct homotopy types; γ and δ are homotopy equivalent.

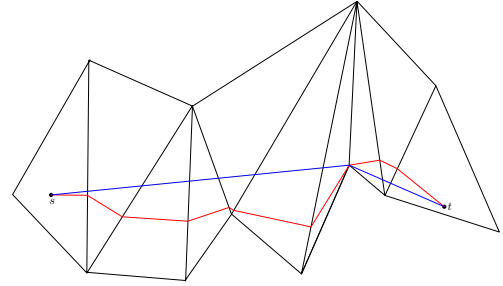


Fig. 2. The solid path is the optimal path; the dashed path is the greedy path.

By removing cut edges, one per hole from \mathcal{D} , we obtain a tree T . We then embed the tree in the hyperbolic plane, such that by tiling copies of the tree we obtain an infinite repeating tree \mathcal{T} .

We use a two-level structure in our scheme.

Level 1: Using a similar idea as in [1], we can use a greedy algorithm to find a path in the universal covering space of \mathcal{D} : the tree \mathcal{T} , with only knowledge of the triangles that contain the source and the destination. This top-level greedy algorithm reveals a sequence of triangles that contains the shortest path of the required homotopy type.

Level 2: Now we have a sequence, $\Delta = \{\Delta_1, \Delta_2, \dots, \Delta_n\}$, of triangles Δ_i , with adjacent triangles in the sequence sharing a common edge. We develop a greedy, local algorithm that “navigates” inside the triangles with total travel length at most a constant times the shortest path inside Δ . The idea is to move through the sequence greedily, always taking the shortest path to the boundary of the next triangle. This local algorithm does not need to know the entire sequence of triangles but only the current triangle and the shared diagonal with the next triangle. The following theorem is a major technical contribution of this paper.

Theorem 1. *The length of the greedy path is at most $15\pi + 2$ times the length of the shortest path.*

Our low-level greedy routing algorithm within a sequence of triangles can be extended to greedy routing inside a sequence of consecutively adjacent simple polygons, with the same worst-case stretch, as we can always further triangulate each simple

¹This paper has been submitted to INFOCOM 2013.

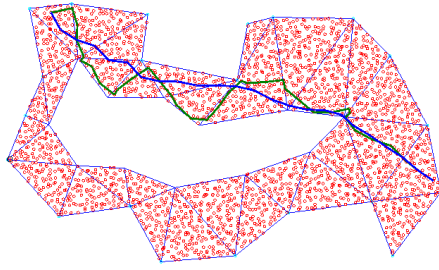


Fig. 3. The blue line is shortest path; the green line is the path used by our algorithm.

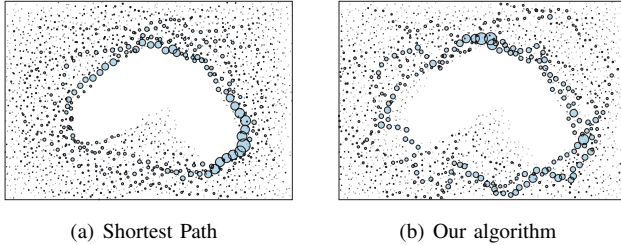


Fig. 4. In this figure, each node is represented by a circle, and the diameter of circle is proportional to the traffic load at that node. For shortest path in 4(a), the loads are largely around the boundary. Our Greedy routing method 4(b) gives better load balance.

polygon. Therefore, as a bonus feature, our new algorithm can replace the local routing scheme in the previous geometric routing schemes that use network decompositions [3]–[5] and provide constant stretch in the local routing part.

Here are the simulations of our algorithm in terms of the routing stretch and load balancing. From an evaluation point of view, we are interested in the performance of the Level 2 of the algorithm.

Fig. 3 is a polygon domain with a hole. The stretch is 1.34.

Fig. 4 shows the traffic load distribution of the shortest path algorithm and our algorithm.

REFERENCES

- [1] W. Zeng, R. Sarkar, F. Luo, X. D. Gu, and J. Gao, “Resilient routing for sensor networks using hyperbolic embedding of universal covering space,” in *Proc. of the 29th Annual IEEE Conference on Computer Communications (INFOCOM’10)*, March 2010, pp. 1694–1702.
- [2] B. Karp and H. Kung, “GPSR: Greedy perimeter stateless routing for wireless networks,” in *Proc. of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, 2000, pp. 243–254.
- [3] Q. Fang, J. Gao, L. Guibas, V. de Silva, and L. Zhang, “GLIDER: Gradient landmark-based distributed routing for sensor networks,” in *Proc. of the 24th Conference of the IEEE Communication Society (INFOCOM)*, vol. 1, March 2005, pp. 339–350.
- [4] G. Tan, M. Bertier, and A.-M. Kermarrec, “Convex partition of sensor networks and its use in virtual coordinate geographic routing,” in *INFOCOM*, 2009, pp. 1746–1754.
- [5] X. Zhu, R. Sarkar, and J. Gao, “Segmenting a sensor field: Algorithms and applications in network design,” *ACM Trans. Sen. Netw.*, vol. 5, no. 2, pp. 12:1–12:32, Apr. 2009. [Online]. Available: <http://doi.acm.org/10.1145/1498915.1498918>
- [6] J. Gao and L. Guibas, “Geometric algorithms for sensor networks,” *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 370, no. 1958, pp. 27–51, 2012. [Online]. Available: <http://rsta.royalsocietypublishing.org/content/370/1958/27.abstract>
- [7] N. M. Amato, M. T. Goodrich, and E. A. Ramos, “A randomized algorithm for triangulating a simple polygon in linear time,” *Discrete Comput. Geom.*, pp. 245–265, 2001.
- [8] B. Chazelle, “Triangulating a simple polygon in linear time,” *Discrete Comput. Geom.*, vol. 6, no. 5, pp. 485–524, Aug. 1991. [Online]. Available: <http://dx.doi.org/10.1007/BF02574703>
- [9] R. Bar-Yehuda and B. Chazelle, “Triangulating disjoint Jordan chains,” *Internat. J. Comput. Geom. Appl.*, vol. 4, no. 4, pp. 475–481, 1994.
- [10] R. Seidel, “A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons,” *Comput. Geom. Theory Appl.*, vol. 1, no. 1, pp. 51–64, Jul. 1991. [Online]. Available: [http://dx.doi.org/10.1016/S0925-7721\(99\)00042-5](http://dx.doi.org/10.1016/S0925-7721(99)00042-5)
- [11] J. Hershberger and J. Snoeyink, “Computing minimum length paths of a given homotopy class,” *Comput. Geom. Theory Appl.*, vol. 4, no. 2, pp. 63–97, Jun. 1994. [Online]. Available: [http://dx.doi.org/10.1016/0925-7721\(94\)90010-8](http://dx.doi.org/10.1016/0925-7721(94)90010-8)
- [12] P. Bose and P. Morin, “Online routing in triangulations,” in *Proceedings of the 10th International Symposium on Algorithms and Computation (ISAAC ’99)*, 1999, pp. 113–122.
- [13] E. Kranakis, H. Singh, and J. Urrutia, “Compass routing on geometric networks,” in *Proc. 11th Canadian Conference on Computational Geometry*, 1999, pp. 51–54.
- [14] P. Bose and P. Morin, “Online routing in triangulations,” *SIAM J. Comput.*, vol. 33, no. 4, pp. 937–951, 2004.
- [15] —, “Competitive online routing in geometric graphs,” *Theor. Comput. Sci.*, vol. 324, no. 2-3, pp. 273–288, 2004.
- [16] P. Bose, A. Brodnik, S. Carlsson, E. D. Demaine, R. Fleischer, A. López-Ortiz, P. Morin, and J. I. Munro, “Online routing in convex subdivisions,” *Int. J. Comput. Geometry Appl.*, vol. 12, no. 4, pp. 283–296, 2002.
- [17] J. S. Mitchell, “Geometric shortest paths and network optimization,” in *Handbook of Computational Geometry*. Elsevier Science Publishers B.V. North-Holland, 1998, pp. 633–701.
- [18] V. J. Lumelsky and A. A. Stepanov, “Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape,” *Algorithmica*, vol. 2, pp. 403–430, 1987.
- [19] P. Eades, X. Lin, and N. C. Wormald, “Performance guarantees for motion planning with temporal uncertainty,” *Australian Computer Journal*, vol. 25, no. 1, pp. 21–28, 1993. [Online]. Available: <http://dblp.uni-trier.de/db/journals/acj/acj25.html#EadesLW93>
- [20] C. H. Papadimitriou and M. Yannakakis, “Shortest paths without a map,” *Theor. Comput. Sci.*, vol. 84, no. 1, pp. 127–150, Jul. 1991. [Online]. Available: [http://dx.doi.org/10.1016/0304-3975\(91\)90263-2](http://dx.doi.org/10.1016/0304-3975(91)90263-2)
- [21] R. Klein, “Walking an unknown street with bounded detour,” in *Proceedings of the 32nd annual symposium on Foundations of computer science*, ser. SFC’91. Washington, DC, USA: IEEE Computer Society, 1991, pp. 304–313. [Online]. Available: <http://dx.doi.org/10.1109/SFCS.1991.185383>
- [22] P. Berman, “On-line searching and navigation,” in *Online Algorithms*, 1996, pp. 232–241.
- [23] A. Hatcher, *Algebraic Topology*. Cambridge University Press, November 2001.
- [24] R. Kleinberg, “Geographic routing using hyperbolic space,” in *Proceedings of the 26th Conference of the IEEE Communications Society (INFOCOM’07)*, 2007, pp. 1902–1909.
- [25] R. Sarkar, “Low distortion delaunay embedding of trees in hyperbolic plane,” in *Proceedings of the 19th international conference on Graph Drawing*, ser. GD’11, 2011, pp. 355–366.
- [26] K. Huang, C.-C. NI, R. Sarkar, J. Gao, and J. S. B. Mitchell, “Bounded stretch homotopic routing using hyperbolic embedding of sensor networks,” <http://page.inf.fu-berlin.de/sarkar/papers/homotopic-full.pdf>.

ISOTOPY CONVERGENCE THEOREM

J.Li*

T.J.Peters†‡

November 3, 2012

Abstract

When approximating a space curve, it is natural to consider whether the knot type of the original curve is preserved in the approximant. This preservation is of strong contemporary interest in computer graphics and visualization. We establish a criterion to preserve knot type under approximation that relies upon convergence in both distance and total curvature.

Keywords: Knot; Ambient isotopy; Convergence; Total curvature; Visualization.

1 Introduction

Convergence for curve approximation is often in terms of distance, such as in Weierstrass approximation theorem [11]. But an approximation in terms of distance does not necessarily yield ambient isotopic equivalence. However, ambient isotopic equivalence is a fundamental concern in knot theory, and a theoretical foundation for curve approximation algorithms in computer graphics and visualization.

So a natural question is what criterion will guarantee ambient isotopic equivalence for curve approximation? The answer is that, besides convergence in distance, an additional hypothesis of total curvature will be sufficient, that is, convergence in both distance and total curvature.

2 Related Work

The Isotopy Convergence Theorem presented here is motivated by the question about topological integrity of geometric models in computer graphics and visualization. The publications [1, 2, 7, 9] are among the first that provided algorithms to ensure ambient isotopic approximations. The paper [6] provided existence criteria for a PL approximation of a rational spline, but did not include any specific algorithms.

Recent progress was made for the class of Bézier curves, by providing stopping criteria for subdivision algorithms to ensure ambient isotopic equivalence for Bézier

curves of any degree n [4], extending the previous work of [9], that had been restricted to degree less than 4.

This work here extends to a much broader class of curves, piecewise C^2 curves, where there is no restriction on approximation algorithms. Because of its generality, this pure mathematical result is potentially applicable to both theoretical and practical areas.

3 Preliminaries

Use \mathcal{C} to denote a compact, regular, C^2 , simple, parametric, space curve. Let $\{C_i\}_1^\infty$ denote a sequence of piecewise C^2 , parametric curves. Suppose all curves are parametrized on $[0, 1]$, that is, $\mathcal{C} = \mathcal{C}(t)$ and $C_i = C_i(t)$ for $t \in [0, 1]$. Denote the sub-curve of \mathcal{C} corresponding to $[a, b] \subset [0, 1]$ as $\mathcal{C}_{[a,b]}$, and similarly use $C_{i[a,b]}$ for C_i . Denote a total curvature as $T_\kappa(\cdot)$.

The definitions [8] of total curvatures of both PL curves and C^2 curves are standard. These can be naturally extended to define total curvatures of piecewise C^2 curves, for which the concept of exterior angles [8] is needed.

Definition 3.1 (Exterior angles of piecewise C^2 curves)

For a piecewise C^2 curve $\gamma(t)$, define the exterior angle at some t_i to be the angle between two vectors $\gamma'(t_i-)$ and $\gamma'(t_i+)$ where

$$\gamma'(t_i-) = \lim_{h \rightarrow 0} \frac{\gamma(t_i) - \gamma(t_i - h)}{h},$$

and

$$\gamma'(t_i+) = \lim_{h \rightarrow 0} \frac{\gamma(t_i + h) - \gamma(t_i)}{h}.$$

Definition 3.2 (Total curvatures of piecewise C^2 curves)

Suppose that a piecewise C^2 curve $\phi(t)$ is not C^2 at finitely many parameters t_1, \dots, t_n . Denote the sum of the total curvatures of all the C^2 sub-curves as $T_{\kappa 1}$, and the sum of exterior angles at t_1, \dots, t_n as $T_{\kappa 2}$. Then the total curvature of $\phi(t)$ is $T_{\kappa 1} + T_{\kappa 2}$.

Definition 3.3 We say that $\{C_i\}_1^\infty$ converges to \mathcal{C} in distance if for any $\epsilon > 0$, there exists an integer N such that $\max_{t \in [0,1]} |C_i(t) - \mathcal{C}(t)| < \epsilon$ for all $i \geq N$.

Definition 3.4 We say that $\{C_i\}_1^\infty$ converges to \mathcal{C} in total curvature if for any $\epsilon > 0$, there exists an integer N such that $|T_\kappa(C_i) - T_\kappa(\mathcal{C})| < \epsilon$ for all $i \geq N$. We designate this property as convergence in total curvature.

*Department of Mathematics, University of Connecticut, Storrs, ji.li@uconn.edu

†Department of Computer Science and Engineering, University of Connecticut, Storrs, tpeters@cse.uconn.edu

‡The authors express their appreciation for partial funding from the National Science Foundation under grants CMMI 1053077 and CNS 0923158. All expressions here are of the authors, not of the National Science Foundation. The authors also express their appreciation for support from IBM under JSA W1056109, where statements in this paper are the responsibility of the authors, not of IBM.

4 Isotopy Convergence

Convergence in distance provides a lower bound of the total curvatures of approximants.

Theorem 1 *If $\{C_i\}_1^\infty$ converges to \mathcal{C} in distance, then for $\forall \epsilon > 0$, there exists an integer N such that $T_\kappa(C_i) > T_\kappa(\mathcal{C}) - \epsilon$ for all $i \geq N$.*

Theorem 2 (Isotopy Convergence) *If $\{C_i\}_1^\infty$ converges to \mathcal{C} in both distance and total curvature, then there exists an N such that C_i is ambient isotopic to \mathcal{C} for all $i \geq N$.*

Let \mathfrak{S} be a set of pairwise disjoint piecewise C^2 curves, (which is a link for closed curves), satisfying the same hypotheses as \mathcal{C} . Let \mathfrak{S}_i be a set of piecewise C^2 parametric curves. The corollary below follows easily.

Corollary 3 ¹ *If the sequence $\{\mathfrak{S}_i\}_1^\infty$ converges to \mathfrak{S} in both distance and total curvature, then there exists an integer N such that \mathfrak{S}_i is ambient isotopic to \mathfrak{S} for all $i \geq N$.*

4.1 A representative example of offset curves

Offset curves are defined as locus of the points which are at constant distant along the normal from the generator curves [5]. They are widely used in various applications, and the related approximation problems were frequently studied [5]. It is well-known [10, p. 553] that offsets of spline curves need not be splines. Here we show a representative example as a catalyst to ambient isotopic approximations of offset curves.

Let $\mathcal{C}(t)$ be a compact, regular, C^2 , simple, space curve parametrized in $[a, b]$, whose curvature κ never equals 1. Then define an offset curve by

$$\Omega(t) = \mathcal{C}(t) + N(t),$$

where $N(t)$ is the normal vector at t , for $t \in [a, b]$.

For example, let $\mathcal{C}(t) = (2 \cos t, 2 \sin t, t)$ for $t \in [0, 2\pi]$ be a helix, then it is an easy exercise for the reader to verify that the above assumptions of \mathcal{C} are satisfied, with $\kappa = \frac{2}{5}$. Furthermore, it is straightforward to obtain the offset curve $\Omega(t) = (\cos t, \sin t, t)$, which is not a spline.

We first show that $\Omega(t)$ is regular. Let $s(t) = \int_a^t |\mathcal{C}'(t)| dt$ be the arc-length of \mathcal{C} . Then by Frenet-Serret formulas [3] we have

$$\Omega'(t) = \mathcal{C}'(t) + N'(t) = (1 - \kappa) \frac{ds}{dt} T + \tau \frac{ds}{dt} B,$$

where T and B are the unit tangent vector and binormal vector respectively. Since $T \perp B$, if $(1 - \kappa) \frac{ds}{dt} \neq 0$ then $\Omega'(t) \neq 0$. But $(1 - \kappa) \frac{ds}{dt} \neq 0$ because $\kappa \neq 1$ and $\mathcal{C}(t)$ is regular by the assumption. Thus $\Omega(t)$ is regular.

Now we define a sequence $\{\Omega_i(t)\}_{i=1}^\infty$ to approximate $\Omega(t)$ by setting

$$\Omega_i(t) = \mathcal{C}(t) + \frac{i-1}{i} N(t).$$

It is obvious that $\{\Omega_i(t)\}_{i=1}^\infty$ converges in distance to $\Omega(t)$. For the convergence in total curvature, note that

$\lim_{i \rightarrow \infty} \Omega_i'(t) = \Omega'(t)$ and $\lim_{i \rightarrow \infty} \Omega_i''(t) = \Omega''(t)$. It follows that

$$\lim_{i \rightarrow \infty} \Omega_i'(t) \times \Omega_i''(t) = \Omega'(t) \times \Omega''(t).$$

Since $|\Omega'(t)| \neq 0$ due to the regularity of $\Omega(t)$. Therefore

$$\lim_{i \rightarrow \infty} \frac{\Omega_i'(t) \times \Omega_i''(t)}{|\Omega_i'(t)|^3} = \frac{\Omega'(t) \times \Omega''(t)}{|\Omega'(t)|^3}.$$

This implies that, at each t , the curvature in the sequence converges to the curvature of $\Omega(t)$. Then the convergence in total curvature follows.

By the Isotopy Convergence Theorem (Theorem 2), we conclude that there exists a positive integer N such that $\Omega_i(t)$ is ambient isotopic to $\Omega(t)$ whenever $i > N$.

5 Conclusion

We derived the Isotopy Convergence Theorem as motivated by applications for knot theory, computer graphics, visualization and simulations. Future research directions may include using the Isotopy Convergence Theorem in knot classification and discovering applications in the area of computational topology.

References

- [1] N. Amenta, T. J. Peters, and A. C. Russell. Computational topology: Ambient isotopic approximation of 2-manifolds. *Theoretical Computer Science*, 305:3–15, 2003.
- [2] L. E. Andersson, S. M. Dorney, T. J. Peters, and N. F. Stewart. Polyhedral perturbations that preserve topological form. *CAGD*, 12(8):785–799, 2000.
- [3] M. P. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice Hall, Upper Saddle River, NJ, 1976.
- [4] J. Li, T. J. Peters, and J. A. Roulier. Ambient isotopy under subdivision. *Preprint*, 2012.
- [5] T. Maekawa. An overview of offset curves and surfaces. *Computer-Aided Design*, 31:165173, 1999.
- [6] T. Maekawa, N. M. Patrikalakis, T. Sakkalis, and G. Yu. Analysis and applications of pipe surfaces. *CAGD*, 15(5):437–458, 1998.
- [7] L. E. Miller. *Discrepancy and Isotopy for Manifold Approximations*. PhD thesis, University of Connecticut, U.S., 2009.
- [8] J. W. Milnor. On the total curvature of knots. *Annals of Mathematics*, 52:248–257, 1950.
- [9] E. L. F. Moore, T. J. Peters, and J. A. Roulier. Preserving computational topology by subdivision of quadratic and cubic Bézier curves. *Computing*, 79(2-4):317–323, 2007.
- [10] L. Piegl and W. Tiller. *The NURBS Book*. Springer, New York, 2nd edition, 1997.
- [11] W. Rudin. *Principles of mathematical analysis (3rd. ed.)*. McGraw-Hill, 1976.

¹We appreciate the insightful comment regarding this corollary provided by an anonymous reviewer in the committee of the 22nd Annual Fall Workshop on Computational Geometry.

Medial Residue On a Piecewise Linear Surface

Erin W. Chambers*

Tao Ju†

David Letscher‡

1 Introduction

The medial axis of an object is a skeletal structure originally defined by Blum [1]. Formally, the medial axis of an object is the set of points having more than one closest point on the boundary of the object; alternatively, it can also be thought of as the set of centers of discs with maximal size that fit within the object, or in a variety of other ways. The medial axis is centered within the object, homology equivalent to the object if it is an open bounded subset of \mathbb{R}^n [4], and (at least) one dimension lower than that of the object. These properties make the medial axis ideal for many applications including shape analysis and robotic path planning.

We are interested in defining a similar skeletal structure on a surface S that inherits the properties of the medial axis. Such a structure could then be used for applications such as shape analysis of surface patches as well as path planning in non-planar domains. We are particularly interested in piecewise smooth surfaces, which are more representative of typical outputs of discrete surface reconstruction algorithms (e.g., triangulated meshes) than globally smooth surfaces.

A natural approach would be to replace the Euclidean distances in the medial axis definition by geodesic distances over S . Indeed, Wolter [8] defines the *geodesic medial axis* on a smooth Riemannian manifold as the centers of *geodesic discs* with maximal size that fit in S . Interestingly, when S is only piecewise smooth, such an approach is not sufficient. Various definitions of the medial axis which are equivalent in \mathbb{R}^n may not yield the same structure on S , and none of these structures guarantees the essential properties of the medial axis (being low-dimensional

and homotopy preserving).

In this paper, we propose a new skeleton definition on a piecewise linear surface S , which we call the *medial residue*, and prove that the structure is a finite curve network that is homotopy equivalent to S . When S is a planar domain, the medial residue is equivalent to the medial axis, and so it is a natural extension of the medial axis onto surfaces. We also develop an efficient algorithm to compute the medial residue on a triangulated mesh, which builds on prior work to compute geodesic distances [5, 6].

2 The medial residue

Let S be a piecewise linear surface in \mathbb{R}^3 . We first consider the set of points on S that do not have a unique direction for shortest geodesic paths to ∂S , denoted \mathcal{M}^{SPD} . Note that \mathcal{M}^{SPD} reduces to the medial axis when S has no curvature. It is also not difficult to show that \mathcal{M}^{SPD} is always a finite curve network. However, \mathcal{M}^{SPD} may not preserve the homotopy of S around non-smooth, concave vertices (where the accumulative angle around the vertex is greater than 2π). For example, consider a concave vertex p that is in \mathcal{M}^{SPD} and that has a neighborhood on S (which we call a *shadow*) such that the shortest path from any point x in the shadow to ∂S goes through p . Note that any point in the shadow would have a unique shortest path direction, and hence \mathcal{M}^{SPD} would avoid the entire shadow, which can potentially cause disconnection in \mathcal{M}^{SPD} .

To achieve homotopy equivalence, we will add a curve subset of the shadows at concave vertices. While the actual geometry of these additional curve segments does not affect the topology of our medial residue, we would like these curves to be “centered”, just like the medial axis. Naturally, we consider straight curves that bisect each shadow zone. We can formalize the notion of “straight” (which was proposed in [7]) and “bisect” as follows:

Definition 2.1. *We say a curve γ is straight if for every point $p \in \gamma$ the left and right curve angles at p are equal.*

*Department of Mathematics and Computer Science, Saint Louis University. Research partially supported by NSF grant CCF 1054779.

†Department of Computer Science and Engineering, Washington University in St. Louis. Research partially supported by NSF grant IIS-0846072.

‡Department of Mathematics and Computer Science, Saint Louis University

On a smooth surface, all geodesics are straight, and in fact this concept is equivalent to being a geodesic. However, on piecewise linear surfaces, there are geodesics that are not straight and straight curves that are not geodesic.

Definition 2.2. A curve γ bisects a piecewise differentiable curve X at time t if $\gamma(t) \in X$ and the two angles bounded by γ and the tangent of X at $\gamma(t)$ are equal.

Our medial residue is simply the union of \mathcal{M}^{SPD} and the straight bisectors of the shadows, or formally:

Definition 2.3. The medial residue, \mathcal{MR} consists of any point $x \in S$ where we either have $x \in \mathcal{M}^{SPD}$ or where there are two distinct shortest paths from x to the boundary, γ_1 and γ_2 , parameterized by arc length, which first intersect $\mathcal{M}^{SPD} \cup \partial S$ at $\gamma_1(t) = \gamma_2(t)$ such that $\gamma_1([0, t]) = \gamma_2([0, t])$ is straight and bisects $\mathcal{M}^{SPD} \cup \partial S$ at $\gamma_1(t)$.

The usefulness of medial residue is reflected in the following theorem:

Theorem 2.4. If S is a PL surface then \mathcal{MR} is a finite curve network homotopy equivalent to S .

3 Algorithm

Given a flat piecewise linear surface (i.e., a polyhedral surface), an algorithm exists that can compute, in $O(n^2 \log n)$ time for a mesh with n edges, a subdivision on each polyhedron face that captures the combinatorial structure of the distance function from a set of point sources [5, 6]. We first show that both the complexity and the correctness of the algorithm still hold when the point sources are replaced by edges on the boundary of the surface. Furthermore, we can show that the \mathcal{M}^{SPD} consists of a subset of arcs and vertices in this subdivision, which can be identified in $O(n^2)$ time. Finally, the bisector curves (the second part of \mathcal{MR}) can be added in $O(n^2)$ time. Hence the end-to-end complexity of computing \mathcal{MR} is $O(n^2 \log n)$.

4 The cut residue

The medial axis has strong connections to the cut locus [9]. However, even defining the cut locus on piecewise linear surfaces is a challenge, since the tangent space is not well defined on a non-smooth surface. The most commonly used definition of the cut locus

of a point x in a non-smooth setting is the closure of the set of points that have two distinct geodesics to x . As with the medial axis, this definition gives problems when applied to a piecewise smooth manifold. As a result, algorithms for computing cut locus on a triangulated mesh either uses approximation [2] or are limited to convex surfaces [3]. By replacing ∂S in our medial residue definition with a point source x , we can similarly define a *cut residue* from x that is equivalent to the cut locus when S is a smooth surface. The homotopy and curve network properties in Theorem 2.4 can be shown to hold for cut residue when S is a piecewise linear surface, and the algorithm of medial residue can be easily adapted as well to allow efficient and exact computation of a cut-locus-like structure on arbitrary polyhedral surfaces.

References

- [1] H. Blum. A transformation for extracting new descriptors of form. *Models for the Perception of Speech and Visual Form*, pages 362–80, 1967.
- [2] Tamal K. Dey and Kuiyu Li. Cut locus and topology from surface point data. In *Proceedings of the 25th annual symposium on Computational geometry*, SCG '09, pages 125–134, New York, NY, USA, 2009. ACM.
- [3] Jin-ichi Itoh and Robert Sinclair. Thaw: A tool for approximating cut loci on a triangulation of a surface. *Experimental Mathematics*, 13(3):309–325, 2004.
- [4] André Lieutier. Any open bounded subset of r^n has the same homotopy type as its medial axis. *Computer-Aided Design*, 36(11):1029 – 1046, 2004. [jce:title;Solid Modeling Theory and Applications; /ce:title;.](#)
- [5] Joseph S. B. Mitchell, David M. Mount, and Christos H. Papadimitriou. The discrete geodesic problem. *SIAM J. Comput.*, 16(4):647–668, August 1987.
- [6] David Mount. Voronoi diagrams on the surface of a polyhedron. Technical report, Dept. of Computer Science, Univ. of Maryland, Baltimore, MD, 1985.
- [7] Konrad Polthier and Markus Schmies. Straightest geodesics on polyhedral surfaces. In *ACM SIGGRAPH 2006 Courses*, SIGGRAPH '06, pages 30–38, New York, NY, USA, 2006. ACM.
- [8] F.-E. Wolter and K.-I. Friese. Local and global geometric methods for analysis interrogation, reconstruction, modification and design of shape. In *Proceedings of the International Conference on Computer Graphics*, CGI '00, pages 137–, Washington, DC, USA, 2000. IEEE Computer Society.
- [9] Franzerich Wolter. Cut locus and medial axis in global shape interrogation and representation. In *MIT Design Laboratory Memorandum 92-2 and MIT Sea Grant Report*, 1992.

Burning the Medial Axes of 3D Shapes

Erin W. Chambers*

Tao Ju†

David Letscher‡

Lu Liu§

1 Introduction

The medial axis of an object, originally proposed by Blum [1], is the set of points having more than one closest point on the boundary of the object. It has been widely used as a shape descriptor due to its many properties, such as being “thin” (being one dimension lower than the object), homology equivalent to the object [3], and capturing the shape features.

The long term goal of our work is to develop the definitions of lower-dimensional shape descriptors as subsets of the medial axis (e.g., medial point of a 2D object, medial curve or point of a 3D object). These even “thinner” descriptors are useful in a range of applications such as shape alignment (using the medial point), 3D shape matching and deformation (using the medial curve).

There have been several definitions of a “medial point” on the medial axis of a 2D object, each defined as the local maximum of some function over the medial axis. The function proposed by Ogniewicz and Ilg [6], called *Potential Residue (PR)*, at a medial axis point x is the minimum length of the object boundary curve between the closest boundary points to x . The function proposed in our prior work [4], called *Extended Distance Function (EDT)*, equals the time at which x is burned away by a fire that is ignited from the ends of the medial axis curves and propagating geodesically along the curves at a constant speed. Both functions were shown to have a unique local maximum for a simply connected 2D object. The local maximum of EDT was also experimentally observed to be more stable than that of PR under boundary perturbations.

For a 3D object, the only mathematical definition of a medial curve that we are aware of was given by Dey and Sun [2]. The authors generalize the PR function to the medial axis of a 3D object (which they

called the *Medial Geodesic Function (MGF)*), so that the value at a medial axis point x is the minimum geodesic distance on the object boundary between the closest boundary points to x . The medial curve is then made up of the singular set of MGF.

In this paper, we propose a new function definition over the medial axis of a 3D shape that generalizes the EDT over the medial axis of a 2D shape. The new function, which we call the *burn time (BT)*, captures the arrival time of a fire ignited from the border of the medial axis sheets and propagating geodesically along the sheets at constant speed. We prove several essential properties of BT that are analogous to EDT. As an on-going work, we are investigating the definition of medial curve based on the singular set of BT, which has the potential to be a more stable descriptor than the MGF-based definition [2].

2 The burn time function

Consider an object $S \subset \mathbb{R}^3$ whose medial axis M is a compact piecewise smooth cell complex. Let $f : \partial M \rightarrow \mathbb{R}$ be the Euclidean distance from points on ∂M to S . Note that f is a 1-Lipschitz function.

M in general is not a 2-manifold, but a collection of sheets joined at non-manifold curves and points. We decompose M into *manifold regions* (denoted by $M^{(2)}$), consisting of all points with a neighborhood in M which is homeomorphic to either an open disk (an interior point) or a half-open disk (on the boundary ∂M), *singular curves* (denoted $M^{(1)}$), consisting of all points with a neighborhood homeomorphic to a union of open and half-open disks identified along an arc, and *singular points* (denoted $M^{(0)}$). We refer to the union of $M^{(1)}$ and $M^{(0)}$ as the *singular set* of M , and use the notation $M^{(s)}$. We say a curve $\gamma : I \rightarrow M$ *does not cross the singular set* if it can be perturbed infinitesimally to a path which avoids the singular set entirely; otherwise, the curve *crosses the singular set*.

Burning on M proceeds similar to [4]. The fire is ignited from each point $x \in \partial M$ at time $f(x)$, and propagates geodesically on M at constant speed. The fire quenches as the fronts meet. When a fire front hits some point $x \in M^{(s)}$ such that x still has some un-burned disk neighborhood, the front dies out and

*Department of Mathematics and Computer Science, Saint Louis University. Research partially supported by NSF grant CCF 1054779.

†Department of Computer Science and Engineering, Washington University in St. Louis. Research partially supported by NSF grant IIS-0846072.

‡Department of Mathematics and Computer Science, Saint Louis University

§Google, Inc.

does not propagate further. With the non-uniform ignition time, burning carries the distance to the object boundary. The dying-out rule ensures that the burning is not affected by small sheets in M arising from perturbations of the object boundary.

We give an explicit definition of burn time that is analogous to definition of the geodesic distance to the boundary. While the latter is the length of the shortest path, we consider the length of some shortest *path tree* that branches at the singular set $M^{(s)}$. Formally,

Definition 2.1 *A path tree for $x \in M$ is a map $t : T \rightarrow M$ where T is a rooted tree, t maps the root to x , and every leaf of T is mapped to the boundary of M , such that:*

- *t maps any vertex of T to $M^{(s)}$.*
- *for every vertex v of T and disk D embedded in M with $t(v) \in D$, $t(T_v) \cap D \neq \{t(v)\}$ (where T_v is the subtree of T rooted at v)*
- *t maps every edge to a path that does not cross the singular set*

Intuitively, the branching rule means that each vertex of the path tree will have at least one outgoing child path that lies on each disk neighborhood of the vertex. We further define the *length* of a path tree as the supremum of that length (in M) of any path from the root to a leaf plus the function value f on that leaf. We then define *burn time (BT)* $BT_M(x) = \inf_{(t,T)} \text{len}(t,T)$. We say a path tree is *minimal* if it gives a path tree for every point in T , so for every $p \in T$, $BT_M(t(p)) = \text{len}(f, T_p)$.

BT is a generalization of the geodesic distance function from a manifold surface to a non-manifold surface. When M consists only of manifold regions, $BT_M(x)$ is the shortest geodesic distance from x to ∂M , and the minimal path tree at x is the shortest geodesic path (void of any branching vertices). When $M^{(s)}$ is non-empty, $BT_M(x)$ still behaves like a geodesic distance function in the manifold regions while exhibiting similar properties to EDT [4]:

Proposition 2.2 *$BT_M(x)$ has these properties:*

1. *It is 1-Lipschitz over a manifold region or along a singular curve.*
2. *It is upper semi-continuous everywhere. Furthermore, $BT_M(x) = \lim_{n \rightarrow \infty} BT_M(x_n)$ for some sequence $\{x_n\}$ converging to x .*
3. *It has no local minima away from ∂M .*
4. *$\{x \in M \mid BT_M(x) = \infty\}$ is equal to the maximal closed subcomplex of M .*

3 Future work

We are currently investigating an algorithm to compute BT over a discretization of the medial axis as a triangulated mesh. In this setting, a minimal path tree crosses a triangle or an edge for a bounded number of times, and hence we expect BT to be computable, for example using a front-advancing algorithm (akin to that for computing geodesic distances [5]). While it is natural to consider the singular set of BT as the medial curve, we have observed that this set alone may not preserve the homotopy of the medial axis, and we are investigating means to restore the homotopy by adding additional structures. Finally, it would be interesting to study the stability of both EDT and BT under boundary perturbations.

References

- [1] H. Blum. A transformation for extracting new descriptors of form. *Models for the Perception of Speech and Visual Form*, pages 362–80, 1967.
- [2] Tamal K. Dey and Jian Sun. Defining and computing curve-skeletons with medial geodesic function. In *SGP '06: Proceedings of the fourth Eurographics symposium on Geometry processing*, pages 143–152, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.
- [3] André Lieutier. Any open bounded subset of r^n has the same homotopy type as its medial axis. *Computer-Aided Design*, 36(11):1029 – 1046, 2004. Solid Modeling Theory and Applications.
- [4] Lu Liu, Erin W. Chambers, David Letscher, and Tao Ju. Extended grassfire transform on medial axes of 2d shapes. *Comput. Aided Des.*, 43:1496–1505, November 2011.
- [5] Joseph S. B. Mitchell, David M. Mount, and Christos H. Papadimitriou. The discrete geodesic problem. *SIAM J. Comput.*, 16(4):647–668, August 1987.
- [6] R. Ogniewicz and M. Ilg. Voronoi skeletons: Theory and applications. In *Proc. Conf. on Computer Vision and Pattern Recognition*, pages 63–69, 1992.

A New Approach to Output-Sensitive Voronoi Diagrams and Delaunay Triangulations

Gary L. Miller and Donald R. Sheehy

Abstract

We present an algorithm for computing Voronoi diagrams and Delaunay triangulations of point sets in \mathbb{R}^d . We also give an output-sensitive analysis, proving that the running time is at most $O(m \log n \log \Delta)$, where n is the input size, m is the output size, and the spread Δ is the ratio of the diameter to the closest pair distance. For many realistic settings, the spread is polynomial in n , in which case we have the only known algorithm that is within a poly-logarithmic factor of optimal for the entire range of output sizes and any fixed dimension.

1 Introduction

Delaunay refinement starts with the Delaunay triangulation of a set of points P and then proceeds to add extra points called *Steiner points* to improve the *quality* of the Delaunay simplices (see for example [4]). Here quality could take different definitions depending on the application, and we call the output a *quality mesh* on a *well-spaced* superset of P . This simple procedure has a worst-case running time that is at least the cost of building Del_P , the Delaunay triangulation of the input points, since that is the first step. In 2006, the Sparse Voronoi Refinement (SVR) algorithm of Hudson, Miller, and Phillips reordered the priorities of the standard algorithm and proved a nearly optimal bound of the running time for any fixed dimension d [5]. In particular, they showed that for many inputs, one can compute the Delaunay triangulation of a well-spaced superset of the input in less time than it would take to compute the Delaunay triangulation of the input alone. This is only possible because of the large gap between the best-case and worst-case complexity of Delaunay triangulations as a function of $n := |P|$ [8]. The quality condition guarantees that the refined output lies close to the best-case, i.e. all vertices touch only a constant number of Delaunay simplices.

In this paper, we turn this story around and explore the reverse question: If computing a Delaunay triangulation of P is no longer a prerequisite for computing a quality mesh, then might it be possible to use the quality mesh to efficiently compute the Delaunay triangulation of P ? Indeed, we give a simple algorithm that removes all of the Steiner points by a simple local flipping routine, leaving behind Del_P . We show how to characterize the flips in terms of the intersection of two Voronoi diagrams. Then, we bound the total number of combinatorial changes throughout the algorithm by bounding these intersections, which can be done using standard tools from the mesh generation literature.

2 Related Work

Previous output sensitive methods for Voronoi diagrams in higher dimensions were based on computing convex hulls. The shelling approach of Seidel achieves $O(n^2 + m \log n)$ running time [7]. This was improved slightly to lessen the quadratic preprocessing by Matousek and Schwartzkopf [6]. Another approach is a “gift-wrapping” algorithm due to Chan [2]. Later improvements by Chan et al. give an $O(m \log^2 n)$ for Voronoi diagrams in \mathbb{R}^3 [3]. Similarly, an $O(m \log^3 n)$ algorithm was given for Voronoi diagrams in \mathbb{R}^4 by Amato and Ramos[1].

3 The Algorithm

There are three phases to the algorithm. In the first phase, a quality mesh is constructed using SVR within a bounding box containing the points. This mesh has $O(n \log \Delta)$ vertices and $O(n \log \Delta)$ simplices. The second phase uses local flips to remove the Steiner points. These flips are ordered to maintain a weighted Delaunay triangulation with the weights of all input points increasing uniformly from 0 to ∞ . The potential flips are stored in a heap and ordered according

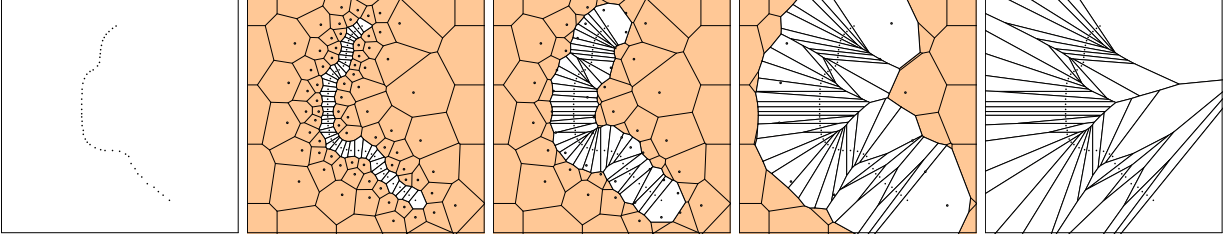


Figure 1: An illustration of the algorithm from left to right. Starting with a point set, it is extended to a quality mesh. Then the weights of the input points are increased until the extra cells disappear.

to the weights of the input points when the flip will occur. In the third and final phase, the boundary vertices are removed. These vertices must be handled separately because they appear in the weighted Delaunay triangulation for all possible weight assignments to the points.

4 Analysis

When do flips happen? The key to the analysis is to bound the number of flips that occur in the transformation from Vor_M to Vor_P . We do this by observing that a flip happens at time t exactly if there is a degenerate $(d+2)$ -tuple of points under the standard lifting

$$p \mapsto (p, \|p\|^2 - w_p^2),$$

where the weight w_p is t for input points and 0 for Steiner points. That is, all $d+2$ lifted points lie on a common hyperplane. This hyperplane in \mathbb{R}^{d+1} has a dual point using the duality:

$$(x_1, \dots, x_{d+1}) \Leftrightarrow y_d = 2x_1y_1 + \dots + 2x_dy_d - x_d.$$

For such collections of points, the dual point projects to the *orthocenter* of the weighted points, the center of a sphere that intersects tangentially each the spheres with center p and radius w_p . By partitioning these points into two sets I and S depending on whether they are input points or Steiner points, one can easily compute that the distance from the orthocenter to each point of I is the same. Similarly, the distance from the orthocenter to each point of S is the same. Thus, the orthocenter is the intersection of a face of Vor_P and a face of Vor_M . Since all flips are characterized this way, we can bound the number of flips by bounding the number of intersections between these two Voronoi diagrams.

Counting flips. In many problems, it is quite challenging to bound the number of flips, but several factors make it possible for our algorithm. First, the radius of the Voronoi cell of a Steiner point in a quality mesh is proportional to its distance to the nearest input point. Second, the Voronoi cells of Steiner points in Vor_M have only a constant number of faces. So, by an easy packing argument, we get that the number of face-face intersections between Vor_P and Vor_M is at most $O(m \log \Delta)$. That is, each $(d-k)$ -face of Vor_P can only intersect $O(\log \Delta)$ k -faces of Vor_M . So the total number of flips is $O(m \log \Delta)$ and the total running time is $O(m \log n \log \Delta)$, where the extra log term comes from the heap operations needed to order the flips.

References

- [1] Nancy M. Amato and Edgar A. Ramos. On computing voronoi diagrams by divide-prune-and-conquer. In *Symposium on Computational Geometry*, pages 166–175, 1996.
- [2] Timothy M. Chan. Output-sensitive results on convex hulls, extreme points, and related problems. *Discrete & Computational Geometry*, 16(4):369–387, 1996.
- [3] Timothy M. Chan, Jack Snoeyink, and Chee-Keng Yap. Primal dividing and dual pruning: Output-sensitive construction of four-dimensional polytopes and three-dimensional voronoi diagrams. *Discrete & Computational Geometry*, 18(4):433–454, 1997.
- [4] Siu-Wing Cheng, Tamal K. Dey, and Jonathan Richard Shewchuk. *Delaunay Mesh Generation*. CRC Press, 2012.
- [5] Benoît Hudson, Gary Miller, and Todd Phillips. Sparse Voronoi Refinement. In *Proceedings of the 15th International Meshing Roundtable*, pages 339–356, Birmingham, Alabama, 2006. Long version available as Carnegie Mellon University Technical Report CMU-CS-06-132.
- [6] Jiří Matoušek and Otfried Schwarzkopf. Linear optimization queries. In *Symposium on Computational Geometry*, pages 16–25, 1992.
- [7] Raimund Seidel. Constructing higher-dimensional convex hulls at logarithmic cost per face. In *STOC: ACM Symposium on Theory of Computing*, 1986.
- [8] Raimund Seidel. On the number of faces in higher-dimensional Voronoi diagrams. In *Proceedings of the 3rd Annual Symposium on Computational Geometry*, pages 181–185, 1987.

Solving The Cutting Flow Problem for Prismatic Mesh Subdivision

Xiaotian Yin*

Wei Han[†]

Xianfeng Gu[‡]

Shing-Tung Yau[§]

November 5, 2012

Abstract

This paper is motivated by the problem of subdividing a prismatic mesh to a tetrahedral mesh (without inserting Steiner points) so as to not only match arbitrarily prescribed boundary conditions but also allow arbitrary topologies in the base mesh. We explore all possible combinations of these two factors, and propose a complete solution to this 3D problem by converting it to an equivalent 2D graph problem, called "cutting flow problem". For each case, we not only prove the sufficient and necessary condition for the existence of solutions, but also provide linear and provable algorithms to compute a solution whenever there is one.

1 Motivations

A **prismatic mesh** consists of a set of triangular prisms, where each prism is a volumetric element bounded by two triangular faces and three quadrilateral faces, and different prisms are glued together along same type of faces (i.e. triangle to triangle, quadrangle to quadrangle). It in general comes in layers, where each layer is an extrusion of a triangular mesh (i.e. **base mesh**) along a line interval (i.e. fiber).

prismatic meshes are often required to be converted to tetrahedral meshes, especially for the purpose of computation and simulation. In finite element methods, many solvers are designed for tetrahedral meshes and do not support prismatic elements. In computer graphics, many efficient algorithms for volume rendering, iso-contouring and particle advection only work for meshes of tetrahedra. Therefore how to triangulate a prismatic mesh becomes a desirable task.

Splitting a single prism into three tetrahedra is an easy task, but cutting a set of prisms consistently is much more challenging. Here we only consider conversions without inserting additional points (i.e. Steiner points). Under certain circumstances user may wish to have control on the boundary triangulation, i.e. the subdivision of the quadrilateral faces on the boundary of the prismatic mesh, and the internal subdivision must conform to such boundary conditions. In addition, the underlying base mesh may have various topologies, which could bring another level of difficulty to the problem of extending the boundary triangulation into the inside.

There has been a rich literature on triangulating non-tetrahedral volumetric meshes. As an example, [1] proposed an algorithm to subdivide a volumetric mesh consisting of mixed elements (pyramids, prisms, and hexahedra) into tetrahedra by comparing and ordering vertex indices. However, most of these works do not discuss fixed boundary conditions, which makes a completely different problem. In one of our earlier work [2], we studied this problem with prescribed boundary conditions, but only for topological disks. To our best knowledge, the result presented here is the first complete solution to prismatic mesh subdivision for all possible boundary conditions and base topologies.

2 Problem Statement

We subdivide each layer in a prismatic mesh separately, and formulate this problem as an equivalent 2D graph flow problem in the underlying base mesh. The work is based on the following intuition.

For every individual prism in the mesh, as shown in Figure 1, each quadrilateral face should be split into two triangles, either through the *diagonal* (lower-left to upper-right) or *anti-diagonal* (lower-right to upper-left). We model such a splitting process by assigning directed flows across edges of the base triangular face. If a quadrilateral face is split along diagonal, we put a flow into the base triangle across the corresponding

*Mathematics Department, Harvard University, xyin@math.harvard.edu.

[†]Mathematics Department, Harvard University, weihan@math.harvard.edu.

[‡]Computer Science Department, Stony Brook University, gu@cs.sunysb.edu.

[§]Mathematics Department, Harvard University, yau@math.harvard.edu.

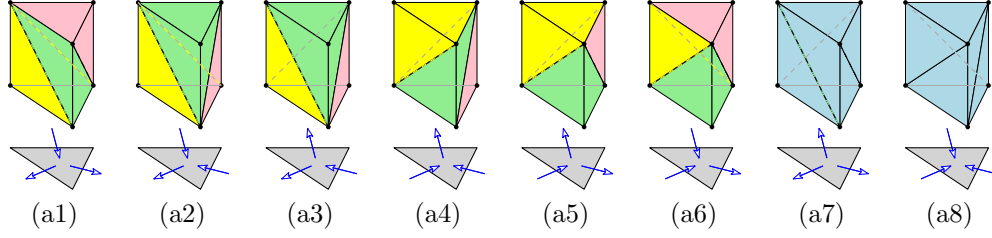


Figure 1: The intuition: subdividing each individual prism by cutting flow. (a1) to (a6) are valid cuttings, (a7) and (a8) are invalid.

base edge; otherwise, put a flow out of the base triangle. As shown in Figure 1, a splitting over a prism is valid if and only if there are both inflow and outflow in the base triangle. In addition, two adjacent prisms should have a consistent flow on their common quadrilateral face.

To formalize this problem, we need some notations here. Given a triangular mesh G (i.e. a primal graph), denote its *augmented dual graph* (or *dual graph* for short) as $\tilde{G}^* = (V^* \cup \tilde{V}, E^* \cup \tilde{E})$, where E^* and \tilde{E} are sets of dual edges corresponding to the inner and boundary primal edges in G respectively, V^* is a set of dual vertices corresponding to the primal faces in G , \tilde{V} is a set of virtual dual vertices placed off the boundary of G to bound the edges in \tilde{E} .

Now we can formally define the original 3D problem of prismatic mesh subdivision as an equivalent 2D flow problem in the augmented dual graph.

PROBLEM 1. (*The Cutting Flow Problem*) Given a triangular mesh M with augmented dual graph $\tilde{G}^* = (V^* \cup \tilde{V}, E^* \cup \tilde{E})$, find a flow (called *cutting flow*) on the edge set of \tilde{G}^* , such that:

- *Fixed Boundary*: The flow on \tilde{E} is given as input and cannot be changed.
- *No Source/Sink*: Every vertex in V^* must have both inflow and outflow.

3 Results

In this work we provide a complete solution to the cutting flow problem (and therefore the original problem of prismatic mesh subdivision). We consider all possible boundary conditions (fixed or free) and all possible base mesh topologies (simply-connected or multiply-connected, planar or non-planar). For each combination of these two factors, we not only prove the sufficient and necessary condition for existence of solutions, but also provide efficient algorithms to find a solution if there is one.

The results are summarized in table 1, case by case. Note that this problem is solvable in most cases, except for a special case on simply-connected planar domain. In fact, the problem is not solvable if and only if the dual graph is a tree and the boundary condition is uniform. In any other case, the problem can be solved by a linear algorithm.

<i>Topo</i> <i>Bnd</i>	<i>Simply-Connected</i> <i>& Planar</i>	<i>Multiply-Connected</i> <i>& Planar</i>	<i>Multiply-Connected</i> <i>& Non-Planar</i>	<i>Simply-Connected</i> <i>& Non-Planar</i>
<i>Fixed</i>	Conditionally Solvable	Always Solvable	Always Solvable	Always Solvable
<i>Free</i>	Always Solvable			

Table 1: Results for all possible cases of different boundary conditions ("Bnd") and base topologies ("Topo").

References

- [1] J. Dompierre, P. Labb, M.-G. Vallet, and R. Camarero. How to subdivide pyramids, prisms, and hexahedra into tetrahedra. In *8th International Meshing Roundtable*, pages 195–204, 1999.
- [2] X. Yin, W. Han, X. Gu, and S.-T. Yau. The cutting pattern problem for tetrahedral mesh generation. In *20th International Meshing Roundtable*, pages 217–236, 2011.

Packing disks that touch the boundary of a square

Adrian Dumitrescu* Csaba D. Tóth†

Abstract. It is shown that the total perimeter of n pairwise disjoint disks lying in the unit square $U = [0, 1]^2$ and touching the boundary of U is $O(\log n)$, and this bound is the best possible.

1 Introduction

Given a collection of geometric objects \mathcal{O} , and a container $U \subseteq \mathbb{R}^d$, a *packing* is a finite set of translates of objects from \mathcal{O} that are pairwise disjoint and lie in the container C . Extremal properties of packings (e.g., the densest packing of unit balls) are classical problems in discrete geometry. We consider a new variant of the problem related to TSP with neighborhoods (TSPN).

In the *Euclidean Traveling Salesman Problem* (ETSP), given a set S of n points in \mathbb{R}^d , we wish to find closed polygonal chain (*tour*) of minimum Euclidean length whose vertex set is S . The Euclidean TSP is known to be NP-hard, but it admits a PTAS in \mathbb{R}^2 . In the *TSP with Neighborhoods* (TSPN), given a set of n sets (neighborhoods) in \mathbb{R}^d , we wish to find a closed polygonal chain of minimum Euclidean length that has a vertex in each neighborhood. The neighborhoods are typically simple geometric objects such as disks, polygons, line segments, or lines. TSPN is also NP-hard; it admits a PTAS for certain types of neighborhoods [5], but is hard to approximate for others [1].

For n connected (possibly overlapping) neighborhoods in the plane, TSPN can be approximated with ratio $O(\log n)$ by the algorithm of Mata and Mitchell [4]. At its core, the $O(\log n)$ -approximation relies on the following early result by Levcopoulos and Lingas [3]: every (simple) rectilinear polygon P with n vertices, r of which are reflex, can be partitioned into rectangles of total perimeter $\text{per}(P) \log r$ in $O(n \log n)$ time.

One approach to approximate TSPN (in particular, it achieves a constant-ratio approximation for unit disks) is the following. Given a set S of n neighborhoods, compute a maximal subset $I \subseteq S$ of pairwise disjoint neighborhoods (i.e., an independent set), compute a good tour for I , and then augment it by traversing the boundary of each set in I . Since each neighborhood in $S \setminus I$ intersects some neighborhood in I , the augmented tour

visits all objects in S . This approach is particularly appealing since good approximation algorithms are often available for pairwise disjoint neighborhoods [5]. The bottleneck of this approach is extending a tour of I by the total perimeter of the objects in I . This lead us to the following problem [2] (see Fig. 1):

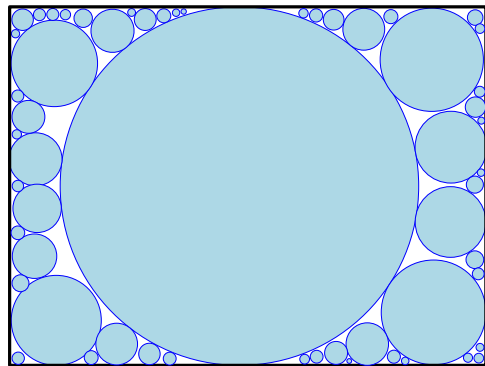


Figure 1: A packing of disks in a rectangle, with all disks touching the boundary.

Given a simple polygonal domain P in the plane and n disjoint disks lying in P and touching the boundary of P , what is the maximum ratio of the total perimeter of the disks and the perimeter of P ? We address this problem in the simple setting where P is a unit square.

Theorem 1 *The total perimeter of n pairwise disjoint disks lying in the unit square $U = [0, 1]^2$ and touching the boundary of U is $O(\log n)$. Apart from the constant factor, this bound is the best possible.*

2 Proof of Theorem 1

It is enough to bound the total diameter of n disks.

Upper bound. Let S be a set n disjoint disks in the unit square $U = [0, 1]^2$ that touch the bottom side of U . Shrink each disk $D \in S$ by a factor $\rho \in (\frac{1}{2}, 1]$ from its common point with the x -axis such that its radius becomes $1/2^k$ for some integer $k \in \mathbb{N}$. The disks remain disjoint, they still touch the bottom side of U , and each radius decreases by a factor of at most 2. Partition the resulting disks into subsets as follows. For $i = 1, \dots, \lfloor \log_2 n \rfloor$, let S_i denote the set of disks of radius $1/2^i$; and let S_0 be the set of disks of radius less than $1/n$. The sum of diameters of the disks in S_i ,

*Department of Computer Science, University of Wisconsin-Milwaukee. Email: dumitres@uwm.edu.

†Department of Mathematics and Statistics, University of Calgary, and Department of Computer Science, Tufts University. Email: cdtoth@ucalgary.ca.

$i = 1, \dots, \lfloor \log_2 n \rfloor$, is at most 1, since their horizontal diametrical segments are collinear and disjoint. The sum of diameters of the disks in S_0 is at most 2 since there are at most n disks altogether. Hence the sum of diameters of all original disks is at most $2(2 + \lfloor \log_2 n \rfloor)$, as required.

Lower bound construction. We construct a packing of $O(n)$ disks in the unit square $[-\frac{1}{2}, \frac{1}{2}] \times [0, 1]$ such that every disk touches the x -axis, and the sum of their diameters is $\Omega(\log n)$. To each disk we associate its vertical *projection interval* (on the x -axis). The algorithm greedily chooses disks of monotonically decreasing radii such that (1) every diameter is $1/16^k$ for some $k \in \mathbb{N}_0$; and (2) if the projection intervals of two disks overlap, then one interval contains the other.

For $k = 0, 1, \dots, \lfloor \log_{16} n \rfloor$, denote by S_k the set of disks of diameter $1/16^k$, constructed by our algorithm. We recursively allocate a set $X_k \subset [-\frac{1}{2}, \frac{1}{2}]$ to S_k , and then choose disks in S_k such that their projection intervals lie in X_k . Initially, $X_0 = [-\frac{1}{2}, \frac{1}{2}]$, and S_0 contains the disk of diameter 1 inscribed in $[-\frac{1}{2}, \frac{1}{2}] \times [0, 1]$. The length of each maximal interval $I \subseteq X_k$ will be a multiple of $1/16^k$, so I can be covered by projection intervals of interior-disjoint disks of diameter $1/16^k$ touching the x -axis. Every interval $I \subseteq X_k$ will have the property that any disk of diameter $1/16^k$ whose projection interval is in I is disjoint from any (larger) disk in S_j , $j < k$.

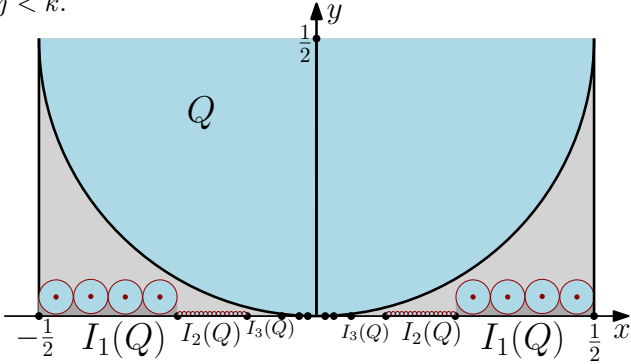


Figure 2: Disk Q and the exponentially decreasing pairs of intervals $I_k(Q)$, $k = 1, 2, \dots$

Consider the disk Q of diameter 1, centered at $(0, \frac{1}{2})$, and tangent to the x -axis (see Fig. 2). It can be easily verified that: (i) the locus of centers of disks tangent to both Q and the x -axis is the parabola $y = \frac{1}{2}x^2$; and (ii) any disk of diameter $1/16$ and tangent to the x -axis whose projection interval is in $I_1(Q) = [-\frac{1}{2}, -\frac{1}{4}] \cup [\frac{1}{4}, \frac{1}{2}]$ is disjoint from Q . Similarly, for all $k \in \mathbb{N}$, any disk of diameter $1/16^k$ and tangent to the x -axis whose projection interval is in $I_k(Q) = [-\frac{1}{2^k}, -\frac{1}{2^{k+1}}] \cup [\frac{1}{2^{k+1}}, \frac{1}{2^k}]$ is disjoint from Q . For an arbitrary disk D tangent to the x -axis, and an integer $k \geq 1$, denote by $I_k(D) \subseteq [-\frac{1}{2}, \frac{1}{2}]$ the pair of intervals corresponding to $I_k(Q)$; for $k = 0$, $I_k(D)$ consists of only one interval.

We can now recursively allocate intervals in X_k and choose disks in S_k ($k = 0, 1, \dots, \lfloor \log_{16} n \rfloor$) as follows. Recall that $X_0 = [-\frac{1}{2}, \frac{1}{2}]$, and S_0 contains a single disk of unit diameter inscribed in the unit square $[-\frac{1}{2}, \frac{1}{2}] \times [0, 1]$. Assume that we have already defined the intervals in X_{k-1} , and selected disks in S_{k-1} . Let X_k be the union of the interval pairs $I_{k-j}(D)$ for all $D \in S_j$ and $j = 0, 1, \dots, k-1$. Place the maximum number of disks of diameter $1/16^k$ into S_k such that their projection intervals are contained in X_k . For a disk $D \in S_j$ ($j = 0, 1, \dots, k-1$) of diameter $1/16^j$, the two intervals in X_{k-j} each have length $\frac{1}{2} \cdot \frac{1}{2^{k-j}} \cdot \frac{1}{16^j} = \frac{8^{k-j}}{2} \cdot \frac{1}{16^k}$, so they can each accommodate the projection intervals of $\frac{8^{k-j}}{2}$ disks in S_k .

We prove by induction on k that the length of X_k is $\frac{1}{2}$, and so the sum of the diameters of the disks in S_k is $\frac{1}{2}$, $k = 1, 2, \dots, \lfloor \log_{16} n \rfloor$. The interval $X_0 = [-\frac{1}{2}, \frac{1}{2}]$ has length 1. The pair of intervals $X_1 = [-\frac{1}{2}, -\frac{1}{4}] \cup [\frac{1}{4}, \frac{1}{2}]$ has length $\frac{1}{2}$. For $k = 2, \dots, \lfloor \log_{16} n \rfloor$, the set X_k consists of two types of (disjoint) intervals: (a) The pair of intervals $I_1(D)$ for every $D \in S_{k-1}$ covers half of the projection interval of D . Over all $D \in S_{k-1}$, they jointly cover half the length of X_{k-1} . (b) Each pair of intervals $I_{k-j}(D)$ for $D \in S_{k-j}$, $j = 0, \dots, k-2$, has half the length of $I_{k-j-1}(D)$. So the sum of the lengths of these intervals is half the length of X_{k-1} ; although they are disjoint from X_{k-1} . Altogether, the sum of lengths of all intervals in X_k is the same as the length of X_{k-1} . By induction, the length of X_{k-1} is $\frac{1}{2}$, hence the length of X_k is also $\frac{1}{2}$, as claimed. This immediately implies that the sum of diameters of the disks in $\bigcup_{k=0}^{\lfloor \log_{16} n \rfloor} S_k$ is $1 + \frac{1}{2} \lfloor \log_{16} n \rfloor$. Finally, one can verify that the total number of disks used is $O(n)$. Write $a = \lfloor \log_{16} n \rfloor$. Indeed, $|S_0| = 1$, and $|S_k| = |X_k|/16^{-k} = 16^k/2$, for $k = 1, \dots, a$, where $|X_k|$ denotes the total length of the intervals in X_k . Consequently, $|S_0| + \sum_{k=1}^a |S_k| = O(16^k) = O(n)$, as required.

References

- [1] M. de Berg, J. Gudmundsson, M. J. Katz, C. Levcopoulos, M. H. Overmars, and A. F. van der Stappen, TSP with neighborhoods of varying size, *J. of Algorithms*, **57(1)** (2005), 22–36.
- [2] A. Dumitrescu and C. D. Tóth, The traveling salesman problem for lines, balls and planes, *Proc. 24th SODA*, 2013, SIAM, to appear.
- [3] C. Levcopoulos and A. Lingas, Bounds on the length of convex partitions of polygons, *Proc. 4th FST-TCS*, vol. 181 of LNCS, 1984, Springer, pp. 279–295.
- [4] C. Mata and J. S. B. Mitchell, Approximation algorithms for geometric tour and network design problems, *Proc. 11th SOCG*, 1995, ACM, pp. 360–369.
- [5] J. S. B. Mitchell, A constant-factor approximation algorithm for TSP with pairwise-disjoint connected neighborhoods in the plane, *Proc. 26th SOCG*, 2010, ACM, 183–191.

Sum of Squared Edges for MST of a Point Set in a Unit Square

Oswin Aichholzer* Sarah R. Allen^{†‡} Greg Aloupis[§] Luis Barba^{¶§}
 Prosenjit Bose[¶] Jean-Lou De Carufel[¶] John Iacono[‡] Stefan Langerman[§]
 Diane L. Souvaine[†]

1 Introduction

Let the *weight* of a tree be the sum of the squares of its edge lengths. Given a set of points P in the unit square let $W(P)$ be weight of the minimum spanning tree of P . If P is simply the four corners of the square, then $W(P) = 3$. Gilbert and Pollack [2] demonstrated that $W(P) = O(1)$ and this was extended to an arbitrary number of dimensions by Bern and Eppstein [1]. While more recent divide-and-conquer approaches have shown that $W(P) \leq 4$, no point sets are known with $W(P) > 3$, and hence it has been widely conjectured (e.g. see [3]) that $W(P) \leq 3$. Here we show that $W(P) < 3.411$.

For a point set P in a unit square, $MST(P)$ denotes a minimum spanning tree of P . Let $MST_k(P)$ denote the subgraph of $MST(P)$ in which all edges of length greater than k have been removed from $MST(P)$. For any given point $X \in P$, define $MST_k(X, P)$ to be the connected component of $MST_k(P)$ containing X . Let \boxplus be the corners of the unit square.

Lemma 1. $W(P) \leq W(P \cup \boxplus)$.

Lemma 2. No edge in $MST(P \cup \boxplus)$ has length greater than 1.

2 Result

By Lemma 1 it suffices to consider only point sets that include the corners of an enclosing unit square.

Kruskal's MST construction algorithm considers all edges defined by P in sorted order. When an edge is considered, it is added to the existing graph only if it doesn't create a cycle. Let e_m be the m^{th} edge added. At step $m = 0$ no edges have been added and at step $m = |P| - 1 =: M$, $MST(P)$ is complete.

*Institute for Software Technology, Graz University of Technology, supported by the ESF EUROCORES programme EuroGIGA CRP ComPoSe, Austrian Science Fund (FWF): I648-N18.

[†]Tufts University, Supported in part by NSF CCF-0830734.

[‡]Polytechnic Institute of New York University.

[§]Université Libre de Bruxelles. (G. Aloupis: Chargé de Recherches du F.R.S.-FNRS; S. Langerman: Directeur de Recherches du F.R.S.-FNRS.)

[¶]Carleton University.

At each step of Kruskal's algorithm, each connected component of edges is a tree. We define t_X^m to be the tree at step m that contains point X . It helps to initialize the algorithm at $m = 0$ by letting every point X of P be a singular tree t_X^0 that is augmented when X is an endpoint of an edge that is added. We also initialize $e_0 = 0$. Notice that $t_X^m = MST_{|e_m|}(X, P)$. Let $\mathcal{CH}(t)$ denote the vertices of the convex hull of a tree t . If X is on $\mathcal{CH}(t_X^m)$, let $\angle^m(X)$ be the range of angles for which it is extreme. We set $\angle^0(X) = [0^\circ, 360^\circ]$. Over time this range of angles is reduced, and may have size 0 if X is no longer on the hull. At any time m , for any given connected component Z , the set of all $\angle^m(X)$ for each point $X \in Z$ partitions the angle range $[0, 360]$.

With this in place we define the region $C^m(X)$. If at time m , X is on $\mathcal{CH}(t_X^m)$ and extreme in some range $[\alpha, \beta] = \angle^m(X)$ then $C^m(X)$ is the sector of a circle centered at X , with radius $\frac{|e_m|}{2}$ and spanning the angle range $[\alpha, \beta]$. If X is not on the hull of t_X^m , then $C^m(X)$ is empty. Let $C^{*m}(X)$ be the union of all the sectors that X has defined up to step m ; that is $C^{*m}(X) = \cup_{\mu=0}^m C^\mu(X)$.

For a tree t_X^m , we define the region A_X^m as follows.

$$A_X^m = \bigcup_{Y \in t_X^m} C^{*m}(Y).$$

A_X^m is contained in the union of discs of radius $\frac{|e_m|}{2}$ centered on all points of P in the same component as X . Points in different components have distance greater than $|e_m|$, otherwise an edge between them would have already been added. Thus if A_X^m and A_Y^m are different, then they are disjoint. Let A^m denote union of all these regions and let Φ^m denote the area of all such regions defined at time m .

At time m , there are $|P| - m$ trees. Recall that points of P not yet joined to other points are also considered to be trees. Let $\ell^m = |e_m|^2$.

Lemma 3. $\Phi^{m+1} = \Phi^m + \frac{\pi}{4}(|P| - m)(\ell^{m+1} - \ell^m)$.

Proof. At time m , each point X on $\mathcal{CH}(t_X^m)$ has a sector $C^m(X)$ with radius $\frac{|e_m|}{2} = \frac{\sqrt{\ell^m}}{2}$. From our definition of $C^m(X)$, the sectors of all points on the

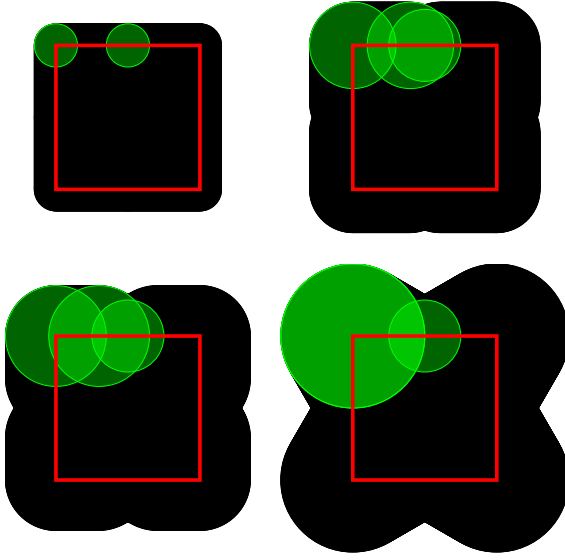


Figure 1: Depiction of R for $d \in \{0.3, 0.6, 0.7, 1.0\}$.

convex hull of t_X^m partition a circle of radius $\frac{\sqrt{\ell^m}}{2}$, which has area $\frac{\pi\ell^m}{4}$. From step m to $m+1$, the radius of each of these sectors increases to $\frac{\sqrt{\ell^{m+1}}}{2}$ and the total area of the partitioned circle increases to $\frac{\pi\ell^{m+1}}{4}$. There are $|P| - m$ trees that each have this growth, and whose regions are disjoint, so multiplying the difference $\frac{\pi}{4}(\ell^{m+1} - \ell^m)$ from each tree by the number of trees, $|P| - m$, gives the result. \square

Let W^m denote the sum of the weights of all trees at time m .

Lemma 4. $W^m = \frac{4}{\pi}\Phi^m - (|P| - m)\ell^m$.

Proof. We induct on m . For the base case, $m = 0$, the spanning tree consists of no edges and all points are disconnected. Consequently, $W^0 = 0$, $\Phi^0 = 0$, and $\ell^0 = 0$. Assume that the statement holds for W^m . We will prove that it holds for W^{m+1} .

Because e_{m+1} is the edge added at step $m+1$, we start with $W^{m+1} = W^m + \ell^{m+1}$. By the induction hypothesis, we substitute W^m to obtain $\frac{4}{\pi}\Phi^m - (|P| - m)\ell^m + \ell^{m+1}$. By substitution, using Lemma 3, this equals $\frac{4}{\pi}\Phi^{m+1} - (|P| - m)\ell^{m+1} + \ell^{m+1}$. Simple rearranging yields the claimed result. \square

Lemma 5. Let d denote $|e_M|$. If $d \leq \frac{1}{2}$, $\Phi^M \leq 2d + \frac{\pi d^2}{4} + 1$. Otherwise, $\Phi^M \leq d^2\sqrt{3} - \frac{1}{\sqrt{3}} + \frac{5\pi d^2}{12} + 4(d - d^2) + 1$.

Proof. In Figure 1, we depict a region R that we claim covers A^M . For every point x on each edge e of the square, define a circle of radius $\min\{\frac{d}{2}, \frac{f}{2}\}$, where f is the distance from x to the farther endpoint of e .

This circle is meant to represent a specific radius of the growing circle that corresponds to e_m as the algorithm progresses. If x were a point in P , then its circle would intersect the equivalent growing circles centered on the endpoints of e . Therefore x would no longer be on the convex hull of its component, after the two endpoints join. This would further imply that no sector of x could keep expanding. We define R to be the union of all such circles centered on the boundary of the square, together with the square region itself. This represents an upper bound on the region that A^M can occupy, as the extreme case occurs when points in P are located on the boundary of the square.

It remains to show that R cannot grow any more on account of points of P inside the square. Suppose that an interior point y grows some sector $C^m(y)$ that contributes towards Φ^M outside R . Without loss of generality let this extra contribution be closest to the top edge e of the square. Just like above, $C^m(y)$ can only grow above e if y is part of the upper hull of t_y^m and that cannot happen if y is in the same component as both endpoints of e . Let x be the orthogonal projection of y on e and assume without loss of generality that the endpoint of e farthest to y is the right endpoint r . Therefore the endpoint of e farthest to x is also r . Furthermore, the midpoints of \overline{xr} and \overline{yr} have the same x -coordinate. Therefore, the portion of $C^m(y)$ above e is contained in the circle of radius $\min\{\frac{d}{2}, \frac{f}{2}\}$ centered at x , which contradicts the assumption. All that remains is to calculate the area of R . This can be done algebraically but details are omitted from this version. \square

Theorem 6. For any set of points P in the unit square, $W(P) \leq \frac{3\sqrt{3}+4}{\pi} - \frac{1}{\pi\sqrt{3}} + \frac{2}{3} \approx 3.4101$.

Proof. From Lemma 1, we can assume that P includes the corners of its enclosing unit square. $W^M = W(P)$, and by Lemma 4 is equal to $\frac{4}{\pi}\Phi^M - \ell^M$. This in turn is bounded in terms of d in Lemma 5. Combining, we obtain the following upper bounds on $W(P)$ in terms of d : $\frac{4d^2\sqrt{3}+16(d-d^2)+4}{\pi} - \frac{4}{\pi\sqrt{3}} + \frac{5d^2}{3} - d^2$ when $d > 0.5$ and $\frac{8d-4}{\pi}$ for $d \leq 0.5$. This function is monotonically increasing for $0 \leq d \leq 1$, so substituting $d = 1$ and simplifying gives the claimed bound. \square

References

- [1] M. W. Bern and D. Eppstein. Worst-case bounds for subadditive geometric graphs. In *Symposium on Computational Geometry*, pages 183–188, 1993.
- [2] E. N. Gilbert and H. O. Pollack. Steiner minimal trees. *SIAM Journal in Applied Mathematics*, 16(1):1–29, 1968.
- [3] D. B. West. <http://www.math.uiuc.edu/~west/regs/mstsq.html>.

Computing Small Hitting Sets for Convex Ranges.

Stefan Langerman*

Mudassir Shabbir†

William Steiger‡

November 5, 2012

Abstract

Let S be a set of n given points in R^2 . If A is a convex subset of R^2 its “size” is defined as $|A \cap S|$, the number of points of S it contains. We describe an $O(n(\log n)^4)$ algorithm to find points $z_1 \neq z_2$, at least one of which must meet any convex set of size greater than $4n/7$; z_1 and z_2 comprise a hitting set of size two for such convex ranges. This algorithm can then be used to construct (i) three points, one of which must meet any convex set of size $> 8n/15$; (ii) four points, one of which must meet any convex set of size $> 16n/31$; (iii) five points, one of which must meet any convex set of size $> 20n/41$.

Let S be a set of n given points in general position in R^2 . If A is a convex subset of R^2 , its “size” is defined to be $|A \cap S|$, the number of points of S that it contains. The (Tukey) depth of a point $z \in R^2$ is defined as the minimum (over all halfspaces h containing z) of $|S \cap h|$, the size of the smallest halfspace containing z . It is familiar that there always exists a point $z \in R^2$ (z not necessarily in S) with depth $d(z) \geq n/3$. Such a point is called a *centerpoint* for S . The constant $c = 1/3$ is best-possible: for every $c > 1/3$ there are sets S with respect to which NO point has depth cn . The interesting algorithm of Jadhav and Mukhopadhyay [2] computes a centerpoint in linear time.

Alternatively, if z is a centerpoint for S , *every* convex set of size $> 2n/3$ MUST contain z . A centerpoint may thus be said to “hit” all convex subsets of R^2 with more than $2/3$ of the points of S . For this reason, centerpoint z is called a *hitting-set (of size 1)* for convex sets of size $> 2n/3$. Mustafa and Ray [4], following related work of Aronov et. al. [1], studied the possibilities for hitting sets with more than one point, a natural extension of the notion of centerpoint. They showed that given $S \subseteq R^2$ there are points $z_1 \neq z_2$ (not necessarily in S) such that every convex set of size $> 4n/7$ must meet at least one of them. In addition they showed the constant $4/7$ to be best possible for hitting sets of size 2: for every $c < 4/7$ there are sets S for which, whatever pair $x \neq y$ be chosen, there is a convex subset containing $> cn$ points of S , but containing *neither* x nor y . In [1] it had been shown that the optimal constant c was in the interval $[5/9, 5/8]$.

Let $c_k \in (0, 1)$ be the smallest constant for which, given any set S of n points in R^2 , there are distinct points z_1, \dots, z_k , at least one of which must meet any convex set of size $> c_k n$. We know $c_1 = 2/3$ and $c_2 = 4/7$. Mustafa and Ray were able to show that $c_3 \in (5/11, 8/15]$, that $c_4 \leq 16/31$ and that $c_5 \leq 20/41$.

*Directeur de Recherches du F.R.S.-FNRS. Département d’Informatique, Université Libre de Bruxelles.

†Department of Computer Science, Rutgers University, 110 Frelinghuysen Road, Piscataway, New Jersey 08854-8004. Work supported by grant 0944081, National Science Foundation, USA.

‡Department of Computer Science, Rutgers University, 110 Frelinghuysen Road, Piscataway, New Jersey 08854-8004. Work supported by grant 0944081, National Science Foundation, USA.

Here we address some algorithmic questions about *finding* small hitting sets. The details are contained in the following statement, and its proof.

Theorem 1 *Let S be a set of n given points in general position in R^2 and take $c_2 = 4/7$. Then in $O(n(\log n)^4)$, distinct points z_1, z_2 may be found so that if A is a convex set of size $> c_2 n$, at least one of these points is in A .*

Consider the set \mathcal{R} of all convex subsets of size $> c_2 n$. For each pair $A \neq B$ in \mathcal{R} consider $A \cap B$. Note that $|A \cap B| > n/7$, so there is a point $p_{A,B} = (u, v) \in A \cap B$ of minimal y -coordinate. The existence proof in [4] showed that z_1 may be taken as such a point, but one for a pair A', B' where $p_{A',B'} = (u, v)$ has v as large as possible (a point in the intersection of two ranges whose lowest point is highest). They also showed that z_2 may then be taken as the (usual) centerpoint for $S \setminus (A' \cap B')$ and everything works out.

Let $p = (u, v)$ be the lowest point in $A' \cap B'$ - the intersection of two ranges, each of size at least $c_2 n$ - where v is as large as possible (its a highest lowest point). The proof of the theorem relies on understanding what such a point looks like in the line arrangement dual to S . We combine this with tools introduced by Matoušek [3] to compute z_1 in the stated complexity. Once we have z_1, z_2 - the centerpoint of $S \setminus (A' \cap B')$ - can be found in linear time.

It is easy now to show

Corollary 1 *As in Theorem 1, in $O(n(\log n)^4)$ we can find (i) distinct points z_1, z_2, z_3 , one of which must meet any convex set of size $> 8n/15$; (ii) points z'_1, z'_2, z'_3, z'_4 , one of which must meet any convex set of size $> 16n/31$; (iii) points $z''_1, z''_2, z''_3, z''_4, z''_5$, one of which must meet any convex set of size $> 20n/41$.*

References

- [1] B. Aronov, F. Aurenhammer, F. Hurtado, S. Langerman, D. Rappaport, S. Smorodinsky, and C. Seara. "Small Weak-epsilon Nets. *Computational Geometry: Theory and Applications*, 42(2) 455-462, (2009).
- [2] S. Jadhav and S. Mukhopadhyay. "Computing a Centerpoint of a Finite Planar Set of Points in Linear Time". *Discrete and Computational Geometry* 12(1) (1994), 291-312.
- [3] J. Matoušek. "Computing the Center of a Planar Point Set." *Discrete and Computational Geometry: Papers from the DIMACS Special Year* Amer. Math. Soc., J.E. Goodman, R. Pollack, W. Steiger, Eds, (1992), 221-230.
- [4] N. Mustafa and S. Ray. "An Optimal Extension of the Centerpoint Theorem." *Computational Geometry: Theory and Applications* 42(7), 505-510, (2009).

Finding the minimum-width V-shape with few outliers

Boris Aronov¹
Özgür Özkan

John Iacono
Mark Yagnatinsky

Polytechnic Institute of New York University

Abstract

A V-shape is an infinite polygonal region bounded by two pairs of parallel rays emanating from two vertices (see Figure 1). We describe a randomized algorithm that, given n points and a number k , finds the minimum-width V-shape enclosing all but k of the points with probability at least $1 - 1/n^c$ for any c , requiring $O(n^2)$ space with expected running time $O(cn^2(k+1)^4 \log n(\log n \log \log n + k))$.

1 Introduction

Motivation. The motivation for this problem comes from curve reconstruction: given a set of points sampled from a curve in the plane, find a shape close to the original curve. It has been suggested in [AD-V] that in an area where the curve makes a sharp turn, it makes sense to model the curve by a *V-shape*. The authors remark that it would be natural to investigate a variant that can handle a small number of outliers. We investigate that variant here. The problem is an instance of a large class of problems known as *geometric optimization* or *fitting* questions, (see [GeomOpt] for a survey).

Definitions. Consider two rays with a common vertex. Call these the *outer rays*. Make a copy of the outer rays superimposed on the original and call the copy the *inner rays*. Translate the inner rays while keeping their common vertex within the convex hull of the outer rays. The region between the inner and outer rays is called a *V-shape* (see Figure 1).

A *strip* is the region bounded by two parallel lines. Observe that a V-shape is contained in the union of two strips. The *width* of a strip is the distance

¹Work by B.A. on this paper has been supported by NSF Grants CCF-08-30691, CCF-11-17336, and CCF-12-18791, and by NSA MSP Grant H98230-10-1-0210.

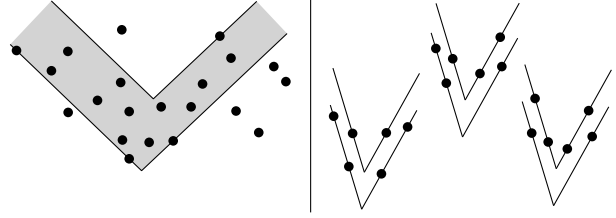


Figure 1: Left: a V-shape with six outliers. Right: a both-outer V-shape, an inner-outer V-shape, and a both-inner V-shape (in left-to-right-order).

between its two lines. The *width* of a V-shape is the width of its wider strip. An *outlier* of a V-shape is a point not contained in that V-shape.

Previous work. In [AD-V], the authors develop an algorithm for covering a point set with a V-shape of minimum width that runs in $O(n^2 \log n)$ time and uses $O(n^2)$ space. They also find a constant-factor approximation algorithm with running time $O(n \log n)$, and a $(1 + \varepsilon)$ -approximation algorithm with a running time of $O((n/\varepsilon) \log n + n/(\varepsilon^{3/2}) \log^2(1/\varepsilon))$, which is $O(n \log n)$ for a constant ε .

Result. Given a set of n points in the plane and an integer k , we show how to find the minimum-width V-shape enclosing all but k of the points.

2 The algorithm

Theorem 1. *There is a randomized algorithm that, given n points and a number k , finds the minimum-width V-shape enclosing all but k of the points with probability at least $1 - 1/n^c$ for any c , requiring $O(n^2)$ space with expected running time $O(cn^2(k+1)^4 \log n(\log n \log \log n + k))$.*

Proof. A V-shape is *locally minimal* with respect to a point set P if there is no way to decrease the width of either of the strips by a slight translation of one of the rays or by a slight simultaneous rotation of two parallel rays, without increasing the number of outliers. (Intuitively, both of its strips should hug the part of the point set they cover.) Since there exists a locally minimal V-shape that achieves the smallest possible width of all V-shapes, it is safe to focus only on locally minimal V-shapes.

We divide locally minimal V-shapes into the same three classes as [AD-V] (see Figure 1). A *both-outer* V-shape is a locally minimal V-shape where both outer

rays have two points on them. A *both-inner* V-shape is a locally minimal V-shape where both inner rays have two points on them. An *inner-outer* V-shape is a locally minimal V-shape where one of the outer rays and one of the inner rays has two points on it. The algorithm works by finding the minimum-width V-shape of each class, and returning the one that has the smallest width of all three.

Our approach for the both-outer case and the inner-outer case was inspired by the approach of [AD-V] for the inner-outer case, except we use a binary search for one step where they use total enumeration. When there are zero outliers, our algorithm for the both-outer and inner-outer cases would be easier to implement than theirs, at the cost of a logarithmic factor in the running time. However, most of the complexity of their solution was in the both-outer case, and we use their both-outer algorithm as a black box in our both-outer algorithm, by running it on random subsets of the point set (or the entire point set when there are zero outliers).

We handle both-inner V-shapes and inner-outer V-shapes in almost the same way (see Figure 2). We begin by enumerating the edges at levels 0 through k of the point set. An *edge at level k* of a point set P is a directed edge connecting two points in the set such that exactly k of the points lie to the left of the directed line through the edge (so in general position there are $n - k - 2$ points to the right). For example, an edge at level 0 is a directed edge of the convex hull. Each enumerated edge e is considered as a candidate for one of the outer rays to go through. The points to the left of e are considered outliers already accounted for. For each e , we do a binary search among points not yet considered outliers. The order for the search is by perpendicular distance from e , which represents the width of the first candidate strip. For each point of the search we find the second strip that has the smallest possible width and still covers the remaining points, except the outliers. If the second strip is wider than the first, the binary search moves farther out from e so that the second strip has fewer points, otherwise it moves closer. To find the second strip, we again enumerate the edges at levels 0 through k of the remaining points. The precise definition of “remaining” here is the key difference between the both-outer and the inner-outer algorithm; we will gloss over this subtle point for now. By now we have chosen three rays, and have no freedom for the fourth: it is dictated by how many more outliers we need. The running time is $O(n^2(k+1)^2 \log^2 n)$.

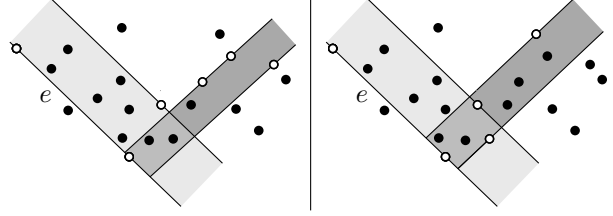


Figure 2: Snapshot of inner-outer algorithm (left) and both-outer algorithm (right).

The best deterministic algorithm we have for finding the minimum-width both-inner V-shape runs in $O(n^3 k^2 \log n)$ time. Instead, we use a randomized algorithm that simply takes many random samples of the given point set. For each sample, it enumerates *all* both-inner V-shapes (with no outliers) using the algorithm from [AD-V]. We show that with probability $1 - n^c$, the minimum-width both-inner V-shape with k outliers will be one of the V-shapes enumerated. The V-shapes we enumerate might have more than k outliers, so we use a range searching data structure from [Range-Search, pages 2–3] to check that, and discard the V-shapes that have too many. The running time of the both-inner case is $O(cn^2(k+1)^4 \log n(\log n \log \log n + k))$, which dominates the run time of the other two cases. \square

Acknowledgments

We would like to thank Muriel Dulieu for participating in discussions of a simpler version of the problem, and Sarel Har-Pelad for the randomized algorithm.

References

- [AD-V] B. Aronov and M. Dulieu, How to cover a point set with a V-shape of minimum width, *WADS* 2011, pages 61–72.
- [GeomOpt] P. K. Agarwal and M. Sharir, Efficient Algorithms for Geometric Optimization, *ACM Computing Surveys*, Vol. 30 Issue 4, 1998, pages 412–458.
- [Range-Search] R. Cole and C. K. Yap, Geometric retrieval problems, *Information and Control*, Vol. 63, Issues 1–2, Oct.–Nov. 1984, pages 39–57.