Young Reseachers Forum: Book of Abstracts

Computational Geometry Week 2014, Kyoto, Japan

Quickly Placing a Point to Maximize Angles

Boris Aronov* aronov@poly.edu
Mark Yagnatinsky† myag@cis.poly.edu

Polytechnic School of Engineering, NYU, Brooklyn, New York

Abstract

Given a set P of n points in the plane in general position, and a set of non-crossing segments with endpoints in P, we seek to place a new point q such that the constrained Delaunay triangulation of $P \cup \{q\}$ has the largest possible minimum angle. The expected running time of our (randomized) algorithm is $O(n^2 \log^2 n)$ on any input, improving the cubic time of the best previously known algorithm. Our algorithm is somewhat complex, and along the way we develop a simpler cubic-time algorithm quite different from the ones already known.

1 Introduction

Often in applications one needs a triangulation (or more generally a mesh) of a point set in the plane, and the triangulation should be as "nice" as possible. What counts as "nice" varies with the application, but one common desire is for the triangles produced to be fat, instead of long and skinny. The ideal in this case is equilateral triangles, but this is usually impossible. However, it is also overkill, since it often suffices that all angles are in a Goldilocks range of "not too big and not too small"; say between 30 and 120 degrees. One common way to formalize the wish for fat triangles is to ask for the smallest angle in the triangulation to be as big as possible. It is well known that the Delaunay triangulation of a planar point set has precisely this property. Unfortunately, the smallest angle in a Delaunay triangulation can be arbitrarily small. Sometimes, it is acceptable to introduce extra points, known as Steiner points, so as to get a better triangulation. However, it is desirable to avoid introducing too many, because they increase the memory and time requirements of all algorithms that operate on the triangulation. There are two natural approaches this problem. One is: given that we want all angles to measure at least x degrees, how many additional points do we need? The other is: given a budget of k points, how large can we force the smallest angle to be? The fixed-budget question was actually addressed in [AAF10], which presented an algorithm that, given a point set P, finds the best placement of k additional points $q_1 \dots q_k$, so that the minimum angle in the Delaunay triangulation of $P \cup \{q_1 \dots q_k\}$ is maximized. In fact, the problem they solved was slightly more general, in that the input also included a set E of non-crossing *edges* (that is, line segments connecting points of P), which must be in the final triangulation. (These are sometimes called *constrained edges*, or simply *constraints*, and the resulting triangulation is called a constrained Delaunay triangulation.) This generalization allows one to triangulate a simple polygon, by specifying the polygon boundary as the set of mandatory edges, and more generally handle real-world applications with boundary conditions.

Unfortunately, the running time of the algorithm in [AAF10] is $n^{O(k)}$, because it relies on explicit construction of high-dimensional arrangements. They also present an algorithm for the case $k = 1$, which runs in time $O(n^{4+\varepsilon})$. In [AY13], we present a slightly super-cubic algorithm for the same problem. Finally, in [AY13fw], a very simple algorithm was presented for the case where E is empty, running in $O(n^{2+\varepsilon})$ time. A slight tweak handles the non-empty case, slowing the running time down to $O(n^{3+\varepsilon})$. In this paper we present yet another near-cubic time algorithm to this problem, and then improve it to $O(n^2 \log^2 n)$ by removing the main bottleneck.

2 The algorithm

We first describe a *decision procedure*, which takes as input a point set P, an edge set E, and a number z representing an angle measurement. It determines whether there is a placement of a new point q such that all angles in T_q, the Delaunay triangulation of $P \cup \{q\}$, have measure at least z. It does this by actually computing the locus V_z of all such placements, and reporting whether it is empty. This decision procedure is used as a black box by a search procedure, which uses it to implicitly search a space of $O(n^3)$ critical values of z in a manner somewhat similar to parametric search.

*Research supported by NSF grants CCF-11-17336 and CCF-12-18791.

†Research supported by GAANN Grant P200A090157 from the US Department of Education and by NSF grant CCF-11-17336.

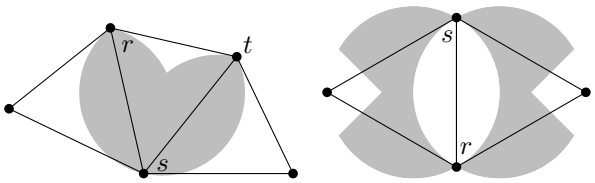


Figure 1: left: $G_{\angle rst}$, right: B_{rs}, where z is 45 degrees.

2.1 Decision procedure

Let T be the constrained Delaunay triangulation of P, and let T_q be the triangulation after point q is added. The decision procedure works by computing two types of regions: "bad" regions, and "good" regions. Each angle a of T is associated with a *good region*, G_a. If the measure of a is at least z, then G_a is the entire plane. Otherwise, it is the locus of points such that placing q there will eliminate at least one of a's two bounding edges from T_q. (Intuitively, if a is small, we want to merge it with a neighboring angle.) Each edge $e = rs$ of T is associated with an *existence region*. This is the locus E_e of points such that placing the new point q there will result in $\triangle qrs$ being a part of T_q. For a fixed value of z, the *bad region* B_e, is simply that part of E_e where some angle of $\triangle qrs$ has measure less than z (see Figure 1). All angles in T_q have measure at least z if and only if q is in all good regions and no bad regions. The decision procedure explicitly computes the locus V_z of such placements for q, and reports whether it is empty. It computes V_z by constructing the good and bad regions, building the arrangement induced by their boundaries, and then traversing this arrangement to label each arrangement feature (face, edge, vertex) as in V_z or not in V_z, in time $O(n^2 \log n)$.

We note that this is already enough to numerically approximate the best achievable angle. We know that it must be between zero and sixty degrees, so we simply do a binary search on that interval, looking for the largest value of z for which V_z is not empty.

2.2 Search procedure

We seek the highest value of z such that V_z contains at least one point. Observe that as z changes, the curves defining good and bad regions also change: some lines rotate, some circles grow, etc. We visualize the dependence on z by letting it represent the third dimension. Note that the arcs delimiting good and bad regions are contained in low-degree algebraic surfaces (after suitable re-parametrization). Let $\mathcal{A}$ be the arrangement of the set S of the $O(n)$ surfaces that arise in this way.

Viewed in this manner, the curves defining V_z become unions of two-dimensional faces of $\mathcal{A}$ and the regions they bound are now solid; they correspond to unions of some three-dimensional cells in $\mathcal{A}$. Two surfaces of S intersect in a curve, and intersecting a third gives a set of discrete points, which are vertices of $\mathcal{A}$. Since these are bounded-degree algebraic surfaces, this set of vertices has a size which is upper-bounded by a constant. If the optimum z is determined by three objects, it must occur at one of these vertices, otherwise, it is defined by one or two objects and can be found using brute force. (It can be shown that the optimum can not be defined by more than three objects.) This is enough to achieve a running time of $O(n^3 \log n)$: compute all vertices, and then sort them by z coordinate. After that, do binary search using the decision procedure.

To speed this up, we need to avoid enumerating all vertices of $\mathcal{A}$, which means we can't afford to find all intersections. Instead, we wish to do a binary-like search on the z values that we would obtain from the enumeration of triples of surfaces of S, without actually enumerating them all. We first pick a random subset of all of those triples: each triple is picked with probability $1/n$, so the expected sample size is quadratic. For each chosen triple, we compute the relevant arrangement vertices and collect their z values. This gives us a set Z of z values, whose expected size is quadratic. Thus, Z partitions the set of $O(n^3)$ candidate z values into $O(n^2)$ intervals, with the largest interval containing $O(n \log n)$ values in expectation. We can afford to do a binary search on Z, which will narrow it down to a single interval $I = [z_0, z_1]$, and then enumerate all vertices within that interval.

We now sketch how to enumerate every vertex in I. Pick a surface s from S. Intersecting s with all other surfaces in S produces a collection of curves in s forming an arrangement. We find the vertices of this arrangement whose z coordinate lie in $[z_0, z_1]$ by using a "plane" sweep from z_0 to z_1, in $O((k_s + n) \log n)$ time, where k_s is the number of vertices found. Repeating this for all surfaces s produces the list of all the vertices of $\mathcal{A}$ lying in the slab $[z_0, z_1]$ in time $O((n^2 + \sum k_s) \log n)$, where we already observed that $\sum k_s$ is $O(n \log n)$, in expectation. Performing a binary search on this set identifies the critical value of z in time $O(n^2 \log^2 n)$, as claimed.

References

[AAF10] B. Aronov, T. Asano, and S. Funke. Optimal triangulations of points and segments with Steiner points. *Int. J. Comput. Geom. Appl.*, 20(1) 89–104, '10

[AY13] B. Aronov and M. Yagnatinsky. How to place a point to maximize angles. *CCCG* 2013, pp. 259–263; see also `http://arxiv.org/abs/1310.6413`.

[AY13fw] B. Aronov and M. Yagnatinsky. A simple way to place a point to maximize angles; presented at *FWCG '13*; `http://www-cs.engr.ccny.cuny.edu/~peter/fwcg13/abstracts/m_yagnatinski.pdf`.

Linear Space Adaptive Data Structure for Planar Range Reporting

Ananda Swarup Das * Prosenjit Gupta†

Abstract

Let S be a set of n points on an $n \times n$ integer grid. The maximal layer of S is a set of points in S that are not dominated by any other point in S. Considering Q as an axes-parallel query rectangle, we design an adaptive space efficient data structure using layers of maxima (iterative maximal layers) for reporting the points in $Q \cap S$. Our data structure needs linear space and can be queried in time $O(\log^{\varepsilon} n + A \log^{\varepsilon} n + k)$. Here A is the number of layers of maxima with points in the query rectangle, k the size of the output and ε is a small arbitrary constant in the range of $(0, 1)$. Also, $A \leq k$. In the worst case, the query time of our data structure is $O(k \log^{\varepsilon} n + k)$. Our model of computation is the word RAM with size of each word being $\Theta(\log n)$.

1 Introduction

In this work, we study the problem of orthogonal range searching for planar points. As stated in [4], intuitively, an adaptive algorithm is the one which enforces its running time to be small for easier instances of the problem while allowing the run time to be large for difficult instances. Thus, an adaptive algorithm focuses on the special instances (say sortedness of the data for example) and performs better for the special cases. For a concrete example, consider the problem of computing the maximal layer for a planar point set. Let $p = (p_x, p_y)$ and $q = (q_x, q_y)$ be two points in $\mathbb{R}^2$ with distinct coordinates. The point p dominates q if $q_x < p_x$ and $q_y < p_y$. Given a point set, the maximal layer is the subset of the points that are not dominated by any other point in the set. It is known that the maximal layer for a point set can be computed in $O(n \log n)$ time. However, if the points are already sorted in non-increasing order of their x-coordinates, then one can traverse the point set from left to right. The first point (the one with the maximum x-coordinate) is sure to be a maximal point. The next point (the one with the second most maximum x-coordinate) will also be a maximal point if its y-coordinate is greater than the y-coordinate of the last discovered maximal point. Thus, if the data is already sorted, the maximal layer for the data set can be found in $O(n)$ time.The previous and the only known adaptive data structure for the problem of orthogonal planar range searching [1] solve the problem in pointer machine model. Our model of computation is the word RAM with size of each word equal to $\Theta(\log n)$. Our proposed algorithm decomposes the point set into maximal layers and can be queried in time $O((1 + A) \log^{\varepsilon} n + k)$ where A is the number of layers of maxima with points in the query rectangle, k the size of the output and ε is a small arbitrary constant in the range of $(0, 1)$. Also, $A \leq k$. Thus, if $A = O(\frac{k}{\log^{\varepsilon} n})$, our algorithm has a query time of $O(\log^{\varepsilon} n + k)$. The proposed data structure needs linear space. In the worst case that is when $A = O(k)$, our query algorithm has a run time of $O(k \log^{\varepsilon} n + k)$ which is at par with the performance of the best known linear space data structure of [3].

1.1 Layers of Maxima

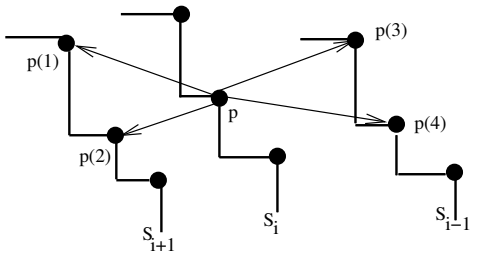


Figure 1: The staircase like chains are the maximal layers. For any point p in a layer, we maintain four pointers for the point. We call them navigational pointers.

Layers of maxima as defined in [2] is the iterative lists of maximal points discovered as follows: Find the maximal points for the set S. Denote the set of these maximal points as S_1. Remove these maximal points from S and then find the maximal points in the remaining set. Denote the new set of maximal points as S_2. Iterate until the set S is empty. The *iteration index i* at which a point p becomes a maximum point is defined to be its layer. See Figure 1. It should be noted that for each layer, the points can be arranged in increasing order of their x-coordinates and can be joined by using alternate horizontal and vertical segments such that the

*Defended Ph.D. thesis on 13th November 2013 at International Institute of Information Technology, Hyderabad, India, anandaswarup@gmail.com

†Department of Computer Science and Engg. Heritage Institute of Technology Kolkata, India, prosenjit_gupta@acm.org

layers are disjoint as shown in Figure 1. We will use the *iteration indices* to distinctly denote the *layer indices*.

1.2 The Range Successor Problem

The range successor problem is defined as *"Given a set S of n points in $\mathbb{R}^2$, preprocess it into a data structure such that given a query $Q = (-\infty, a] \times [c, d]$, we can report the point with the largest x coordinate in $S \cap Q$"*. It is known from Theorem 1 of [5] that the points of the S can be preprocessed into a data structure of size $O(n)$ to answer range successor queries in time $O(\log^{\varepsilon} n)$.

2 The Data Structure and the Query Algorithm

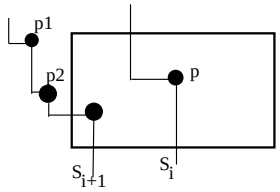


Figure 2: The navigational pointers for the point p in S_i will point to the points p_1 and p_2 of the layer S_{i+1}. Both these points are outside the query rectangle. Yet the layer S_{i+1} intersects the query rectangle and has a point in the query rectangle.

Intuitively, our algorithm for reporting the points in Q is as simple as scanning some arrays that store the points of the maximal layers for the point set S. The details follow. We compute the points of S in layers of maxima. For each point p in S_i, we maintain four navigational pointers respectively pointing to (i) the point p_1 which is the first point above p in S_{i+1}; (ii) the point p_2 which is the first point below p in S_{i+1}; (iii) the point p_3 which is just above p in S_{i-1} and (iv) the point p_4 which is just below p in S_{i-1}. See Figure 1. We then construct four range successor data structures of [5] to report the extreme point in any three sided orthogonal query rectangle. It must be noted that each layer of maxima is stored as an array and the points are stored in decreasing order of their y-coordinates. The layer index m of a layer S_m is used to distinctly denote the corresponding array A_m storing its points. Any layer of maxima can enter a query rectangle by intersecting the upper horizontal boundary or the left vertical boundary. Similarly, it can exit a query rectangle by intersecting the lower horizontal boundary or the right vertical boundary. Given a query rectangle $Q = [a, b] \times [c, d]$, we find the point with maximum x-coordinate in $(-\infty, b] \times [c, d]$. Let the point p reported in previous step belong to the layer S_i. In this paper, we explain our algorithm assuming the layer S_i enters Q by intersecting the upper horizontal boundary and exits by intersecting the lower horizontal boundary of Q. The other cases can be handled similarly. Traverse the layer S_i starting from p and report the points that are in Q. Next, we will traverse the layers that are intersecting Q and are to the left and right of S_i. Since the traversals are analogous, we will elaborate on how to traverse the layers $S_j : j > i$. For each point p of the layer S_i, we check the points p_1, p_2 in S_{i+1}. See Figure 1. If $p_1 \in S \cap Q$ or $p_2 \in S \cap Q$ for any point p of S_i, we visit the layer S_{i+1}. Else we have either encountered a *bad layer* which crosses q but do not have any point in q or the scene is similar to the one depicted in Figure 2. Thus, when while considering the navigational pointers for the each point p in the layer S_i, we check if the vertical segment abutting at the point intersects the upper horizontal boundary of Q. If "Yes", then this layer is a bad layer and hence we find the point with maximum x-coordinate in $(-\infty, p_1(x)] \times [c, d]$ and traverse the corresponding chain. Else, we find the rightmost point $p = (p(x), p(y))$ in Q for the current layer. We then find the topmost point in $[a, p(x)] \times (-\infty, p(y)]$. It should be noted that the first point of S_{i+1} in Q has to be below the point p as p has a navigational pointer to p_2 in the layer S_{i+1}. We stop when we encounter the first chain $S_{j'} : j' > i$ such that j' does not intersect Q. Thus, we conclude:

Theorem 1 *For any $0 < \varepsilon < 1$, there exists a linear space data structure for a static data set of n points on an $n \times n$ grid such that given an axes-parallel query rectangle q, we can report in time $O(\log^{\varepsilon} n + A \log^{\varepsilon} n + k)$, the points in $S \cap Q$.*

References

[1] D. Arroyuelo, F. Claude, R. Dorrigiv, S. Durocher, M. He, A. López-Ortiz, J. I. Munro, P. K. Nicholson, A. Salinger, and M. Skala. Untangled monotonic chains and adaptive range search. *Theor. Comput. Sci.*, 412(32):4200–4211, 2011.

[2] A. L. Buchsbaum and M. T. Goodrich. Three-dimensional layers of maxima. In *ESA*, volume 2461 of *Lecture Notes in Computer Science*, pages 257–269. Springer, 2002.

[3] T. Chan. Persistent predecessor search and orthogonal point location on the word *RAM*. In *SODA*, pages 657–666, 2002.

[4] E. D. Demaine, A. López-Ortiz, and J. I. Munro. Adaptive set intersections, unions, and differences. In *SODA*, pages 743–752, 2000.

[5] Y. Nekrich and G. Navarro. Sorted range reporting. In *SWAT*, pages 271–282, 2012.

Steiner Point Reduction in Planar Delaunay Meshes

Ahmed Abdelkader* Scott A. Mitchell† Mohamed S. Ebeida†

Abstract

We develop a mesh simplification strategy that preserves angle bounds. Preliminary results for a sample of planar Delaunay meshes are then presented. We demonstrate significant improvements of Triangle output, in terms of the number of Steiner points needed for a required angle bound, specially for large bounds where Triangle is known to possibly perform poorly.

1 Introduction

Given an input Delaunay mesh, one goal of many mesh improvement algorithms is to reduce the number of points while preserving angle bounds, e.g., [1]. In this paper, we develop a sampling-based technique that achieves this goal by replacing a pair of neighboring points with one point. We introduce a set of constraints for the location of the new point based on the desired minimum angle and compute an explicit representation of the solution region, which we then sample from to find the replacement point, as shown in Figure 1. This strategy generalizes edge collapse, as it possibly combines edge swaps to update the mesh after replacement.

The simplification concept we adopt is called *sifting* and was first presented in [2]. While in this paper we deal with arbitrary Delaunay meshes, e.g., output of Delaunay Refinement (DR) [3], the study in [2] focused mainly on *explicit sizing functions* and used constraints based on separation of points and maximal coverage of the domain. Thanks to the angle bounds, the number of constraints is bounded by a constant and all updates are local. With the sampling region defined, a replacement sample can be found in constant time [4]. In the same spirit, we call our method *Delaunay Sifting* (DS).

We develop the sift algorithm in Section 2. Then, in Section 3, we show improvement of several Delaunay meshes generated by Triangle [3]. Finally, we conclude and highlight ongoing work in Section 4.

2 The Delaunay Sifting Algorithm

We define the sampling region as the set of valid replacement points that preserve the minimum angle constraints. We denote the minimum angle by α.

*University of Maryland, College Park, akader@cs.umd.edu
†Sandia National Laboratories, : msebeid@sandia.gov

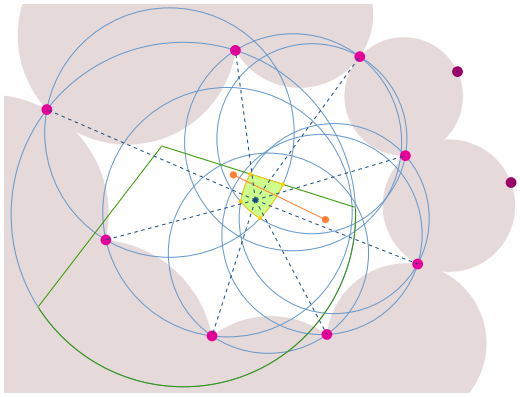


Figure 1: Sifting example (orange: candidate, purple: neighbors, green: sampling region, blue: replacement point, dashed lines: retriangulation, other: constraints).

Given an edge, removing all edges incident on its two end points creates a *gap* in the mesh. The gap is a polygonal region that can be partitioned into triangles by connecting all vertices on the boundary to the replacement of the removed pair of points. Shrinking the gap, by creating *ear* triangles when possible, can also help make sifting possible. Below, we list the constraints that limit the location of valid replacement points.

2.1 Constraint(1) - Neighboring Circumcircles

Neighboring triangular faces surrounding the gap each gives rise to a circumcircle. By definition of the Delaunay triangulation, such circumcircles should be empty.

2.2 Constraint(2) - No Thin Triangles

If we think of any given gap edge as the base of a new triangle and the sample as its apex, then we need to enforce the lower bound for both the apex angle and base angles. For the apex angle, we can simply require the sample point to fall within a circle having the edge as a chord with a central angle of 2α. As for the base angles, the sample point has to fall in the intersection of two half-planes, each making an angle of α with the base.

2.3 Constraint(3) - Boundary Segments

If the candidate edge is incident on a boundary segment, sifting must ensure the boundary face remains the same.

If only one end of the edge is a boundary vertex, it *cannot* be a corner on the boundary. If the edge as a whole is a boundary segment, it must be on a larger segment with boundary neighbors on both sides. In both cases, the new sample will be constrained to lie on the larger boundary segment.

2.4 Putting It All Together

Applying all constraints simultaneously to all gap edges, we compute the desired sampling region, as in Figure 1. We further classify these constraints into two types: inclusive and exclusive constraints. From 2.1, we get a set of exclusive circles and from 2.2 and 2.3, we get two inclusive sets of arc-gons and possibly a line segment. This reduces to a set of boolean conditions that can be easily checked to test any candidate replacement point.

3 Implementation and Results

We opt for a maximal random sifting strategy and apply the DS operation in a uniformly random manner. At each iteration, we generate a random permutation of edges and attempt to sift them in that order. Whenever sifting is successful, we start a new iteration. Otherwise, no more sifting is possible. Other execution strategies can be considered as future work, e.g., smallest-angles first. Random order allows us to explore different evolution paths in our experiments, as we seek the minimum possible number of points, and it also has proven advantages, from an efficiency standpoint. In addition, choosing replacement points randomly preserves desirable spectral properties of the mesh, e.g., blue noise.

In Figure 2, we show a sample of results for our experiments with planar meshes. For each model, we show the input domain, the output of `triangle -q35`[1] and the sifted mesh. For each mesh, we indicate the number of sample points and the sifting ratio achieved. The sifting ratio is defined as the ratio between the reduced number of points and the original number of points. Table 1 shows a summary of more settings.

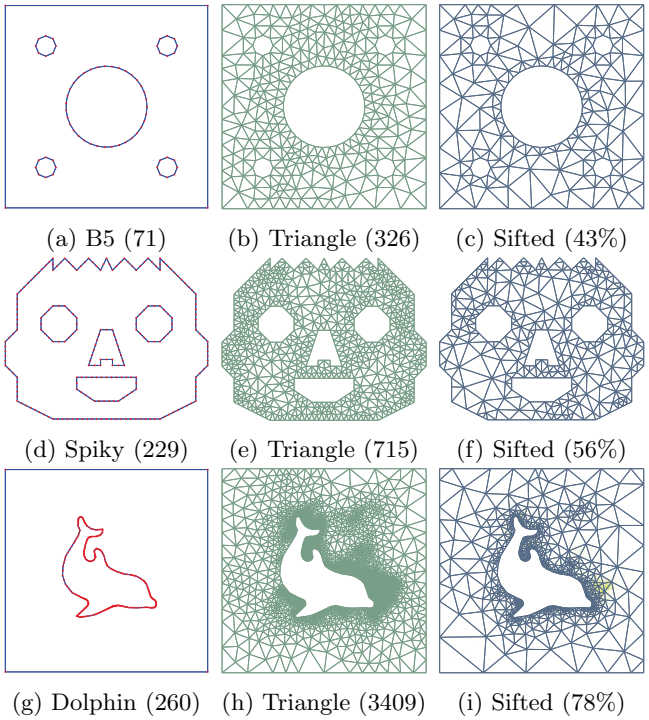
4 Conclusions and Future Work

Our current implementation can perform a few thousand DS attempts per second. The total number of attempts for a given mesh was observed to be linear in the number of edges. Equivalently, it is linear in the number of vertices when taking into account the minimum angle constraints, which bound the degree of vertex connectivity. A slight decrease in the median of angles was also observed. Currently, we are working on extensions to curved surfaces and the derivation of analytical models of both improvement and convergence.

[1] Generate a Delaunay mesh with angles no less than 35 °.

./triangle -q		20		30		35	
`B5`	71	139	17%	197	29%	326	43%
`Spiky`	229	330	54%	505	59%	715	56%
`Dolphin`	260	471	31%	865	49%	3409	78%

Table 1: Size of Triangle outputs and achievable sifting.



(a) B5 (71) (b) Triangle (326) (c) Sifted (43%)
(d) Spiky (229) (e) Triangle (715) (f) Sifted (56%)
(g) Dolphin (260) (h) Triangle (3409) (i) Sifted (78%)

Figure 2: Model, triangle -q35 and sifted meshes.

References

[1] Alper Üngör. Off-centers: A new type of steiner points for computing size-optimal quality-guaranteed delaunay triangulations. In *LATIN 2004: Theoretical Informatics*, pages 152–161. Springer, 2004.

[2] Mohamed S Ebeida, Ahmed H Mahmoud, Muhammad A Awad, Mohammed A Mohammed, Scott A Mitchell, Alexander Rand, and John D Owens. Sifted disks. In *Computer Graphics Forum*, volume 32, pages 509–518. Wiley Online Library, 2013.

[3] Jonathan R Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational geometry*, 22(1):21–74, 2002.

[4] Mohamed S Ebeida, Andrew A Davidson, Anjul Patney, Patrick M Knupp, Scott A Mitchell, and John D Owens. Efficient maximal poisson-disk sampling. In *ACM Transactions on Graphics (TOG)*, volume 30, page 49. ACM, 2011.

Constraint-based surface mapping via hyperbolic orbifold metrics

Alex Tsui*

Abstract

We present a novel method for constructing a minimally distorting surface-to-surface map that is constrained by open curve landmarks with distinguished endpoints. The method is based on harmonic relaxation over a hyperbolic metric in which an optimal diffeomorphism uniquely exists, and the resulting maps are able to match point landmark and curve landmark endpoints exactly while interior curve points are configured in an optimal way. We apply our method to three diverse datasets of biological manifolds, including human faces, human cortical surfaces, and monkey skulls, and we analyze the behavior of the maps in terms of overall geometric distortion, accuracy of landmark matching, and ability to discriminate between classes of surfaces.

1 Introduction

In biomedical imaging, an important application is the analysis of anatomical manifolds. For example, we extract brain meshes from different MRI head scans and compare their geometry. To compare meshes, we need a map that assigns points on one surface to corresponding points on the other. In addition, there are sets of points on each mesh called anatomical landmarks that should be respected by the map. Our goal is to compute this constrained map in a distortion-minimizing manner. Suppose your input is two topologically equivalent triangulated surfaces with landmark curves and the problem is to build a map between them that respects the landmarks yet minimizes distortion. One prior approach [7] treats each surface as if it had boundaries at the landmark curves and then map these secondary surfaces. The surfaces are identified, and the vertices are diffused to reduce distortion, with landmark points being constrained to slide along the boundary. Our approach improves on this by enforcing that curve landmark endpoints are mapped exactly.

2 Method

Our method for mapping between surfaces S_1 and S_2 extends prior work [9]. We compute a discrete conformal map of the surface with landmark curves to *hyperbolic orbifold* by solving an instance of the discrete conformal mapping problem given in [1]. The orbifold is formed by cutting the mesh along the landmark curves and gluing two copies of the surface along the boundaries, resulting in a high-genus surface that admits the prescribed hyperbolic orbifold metric with cone singularities at the curve landmark endpoints. We then compute an arbitrary continuous, bijective initial map from S_1 to S_2. Finally, we apply the theory [3] that there is a unique map that minimizes Dirichlet energy, approximated by

$$E(f) = \frac{1}{2}\sum_{e_{ij}} w_{ij}(f(v_j) - f(v_i))^2, \qquad (1)$$

diffusing the vertex positions by gradient descent to minimize this energy. The gain from this method is in the added control of this diffusion: landmark curve endpoints must correspond precisely, landmark curves

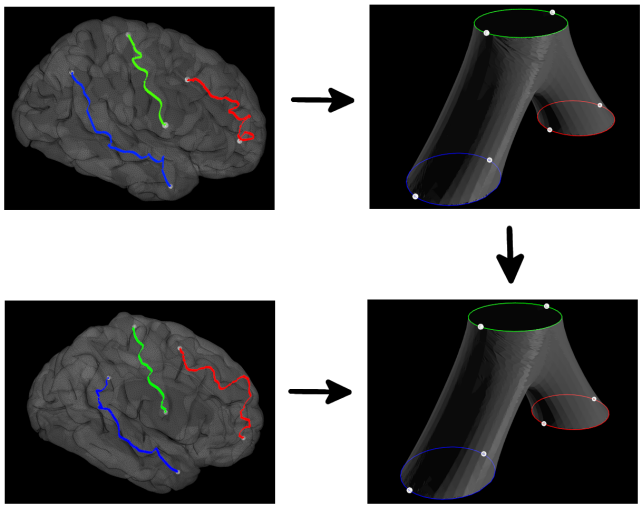


Figure 1: Brains have anatomical landmarks which must be mapped in correspondence (the frontal, central, and temporal sulci are major brain landmarks shown in red, green, and blue, respectively). We conformally map them to orbifolds (i.e. hyperbolic surfaces with cone singularities at landmark endpoints). The pants above show half of the doubled surface that forms this orbifold. An initial map between orbifolds keeps landmarks in correspondence (white points indicate curve landmark endpoints that are mapped exactly), and minimizing Dirichlet energy obtains a minimum distorting map that maintains correspondence.

*Department of Computer Science, University of California Davis, atsui@ucdavis.edu

correspond with landmark curves with freedom to slide between endpoints, and surface interior points can move freely on the surface to reduce distortion.

3 Results

We have a dataset of 100 brain meshes extracted from MRI, 20 scanned monkey skulls, and 900 scanned faces. We ran initial experiments on the brains and faces and plan future work with skulls. We compare our method, `HyperbolicOrbifold` which fixes curve landmark endpoints, with the `HyperbolicPants` method [7], which does not, and the `ConformalLeastSquares` method [5], which computes an optimal spherical conformal map. We compare how much distortion each method produces in the maps (see Table 1) and how well landmarks are matched (see Table 2). *Dilatation*, defined by the ratio of the large and small singular values of of the Jacobian of the map from original to hyperbolic surface Γ/γ (see [6] for equations), estimates local angle distortion (1 being ideal/conformal). *Elastic energy* [5], defined as

$$L(f) = \sum_{e_{ij} \in E} \left(\frac{||f(v_j) - f(v_i)||}{||v_j - v_i||} - 1 \right)^2, \qquad (2)$$

estimates deviation from isometry (0 being ideal). This suggests that the hyperbolic methods induce greater distortion but have greater landmark accuracy than the conformal method, and with fixed endpoints, landmark matching can be improved with slightly more distortion.

Method	Dilatation	Elastic energy
`HyperbolicOrbifold`	1.761 (0.41)	198297 (48032)
`HyperbolicPants`	1.726 (0.37)	165031 (52379)
`ConformalLeastSquares`	1	45764 (13565)

Table 1: Average brain distortion behavior across methods (standard deviation in parentheses).

Method	Min	Mean	Max
`HyperbolicOrbifold`	0.0	0.0	0.0
`HyperbolicPants`	0.072	1.14	12.9
`ConformalLeastSquares`	2.95	12.8	60.8

Table 2: Average brain endpoint mismatch behavior. Units are in millimeters.

We tried to use distortion as an index of shape difference to perform clustering on the mesh datasets. For instance, do maps from young brains to old brains on average induce more distortion than maps between two young brains? Table 3 gives an initial look at brains.

Group	Dilatation	Elastic energy
Young	1.747693 (0.55)	197462 (87909)
Old	1.77486 (0.44)	199206 (79258)

Table 3: Average distortions (with standard deviations in parentheses) for maps from a representative young brain to other young and old brains.

3.1 Issues

The method for computing hyperbolic metrics is highly dependent on condition of the mesh. Prior work [8] notes that near-singular obtuse triangles and can be handled by flipping edges, but this does not address boundary edges. To some extent, we can perform edge flips to fair the mesh [2], but in more extreme cases, mesh points must be manually adjusted. This can be time-consuming as landmarks are specified as sequences of vertices and care must be taken to preserve their integrity between edits. We note that an alternate formulation of discrete conformality [4] attempts to address the need to perform such ad-hoc edge flips.

References

[1] A. Bobenko, U. Pinkall, and B. Springborn. Discrete conformal maps and ideal hyperbolic polyhedra. pages 1–49, 2010.

[2] N. Dyn, K. Hormann, S.-J. Kim, and D. Levin. Optimizing 3d triangulations using discrete curvature analysis. *Mathematical methods for curves and surfaces*, pages 135–146, 2001.

[3] J. Eells and J. Sampson. Harmonic mappings of riemannian manifolds. *American journal of mathematics*, 86:109–160, 1964.

[4] X. Gu, F. Luo, J. Sun, and T. Wu. A discrete uniformization theorem for polyhedral surfaces. 2013. preprint, arXiv:1309.4175 [math.GT].

[5] P. Koehl and J. Hass. Automatic alignment of genus-zero surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3):466–478, 2014.

[6] P. V. Sander, J. Snyder, S. J. Gortler, and H. Hoppe. Texture mapping progressive meshes. *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 409–416, 2001.

[7] R. Shi, W. Zeng, Z. Su, Y. Wang, H. Damasio, Z. Lu, S.-T. Yau, and X. Gu. Hyperbolic harmonic brain surface registration with curvature-based landmark matching. In *IPMI*, pages 159–170, 2013.

[8] B. Springborn, P. Schrder, and U. Pinkall. Conformal equivalence of triangle meshes. *ACM Transactions on Graphics*, 27(3), 2008.

[9] A. Tsui, D. Fenton, P. Vuong, J. Hass, P. Koehl, N. Amenta, D. Coeurjolly, C. DeCarli, and O. Carmichael. Globally optimal cortical surface matching with exact landmark correspondence. In *Information Processing in Medical Imaging*, pages 487–498. Springer, 2013.

Efficient and Robust Persistent Homology for Measures

Mickaël Buchet* Frédéric Chazal† Steve Y. Oudotl‡ Donald R. Sheehy§

Abstract

A new paradigm for point cloud data analysis has emerged recently, where point clouds are no longer treated as mere compact sets but rather as empirical measures. A notion of distance to such measures has been defined and shown to be stable with respect to perturbations of the measure. This distance can easily be computed pointwise in the case of a point cloud, but its sublevel-sets, i.e., the regions where the measure is below a specified value, which carry the geometric information about the measure, remain hard to compute or approximate. This makes it challenging to adapt many powerful techniques based on the Euclidean distance to a point cloud to the more general setting of the distance to a measure on a metric space.

We propose an efficient and reliable scheme to approximate the topological structure of the family of sublevel-sets of the distance to a measure. We obtain an algorithm for approximating the persistent homology of the distance to an empirical measure that works in arbitrary metric spaces. Precise quality and complexity guarantees are given with a discussion on the behavior of our approach in practice.

1 Introduction

Given a sample of points P from a metric space $\mathbb{X}$, the distance function d_P maps each $x \in \mathbb{X}$ to the distance from x to the nearest point of P. The related fields of geometric inference and topological data analysis have provided a host of theorems about what information can be extracted from the distance function, with a particular focus on discovering and quantifying intrinsic properties of the shape underlying a data set [3, 8]. An essential tool in topological data analysis is persistent homology, and the most common goal is to apply the persistence algorithm to distance functions, either in Euclidean space or in metric spaces [2, 6, 11]. From the very beginning, this line of work encountered two major challenges. First, distance functions are very sensitive to noise and outliers (Fig. 1 left). Second, the representations of the sublevel sets of a distance function become prohibitively large even for moderately sized data. These two challenges led to two distinct research directions. First, the distance to the data set was replaced with a distance to a measure induced by that data set [4]. The resulting theory is provably more robust to outliers, but the sublevel sets become even more complex to represent (Fig. 1 center). Towards more efficient representations, several advances in *sparse filtrations* have led to linear-size constructions [5, 9, 10], but all of these methods exploit the specific structure of the distance function and do not obviously generalize. In this work, we bring these two research directions together by showing how to combine the robustness of the distance to a measure, with the efficiency of sparse filtrations.

*Inria Saclay Île-de-France, mickael.buchet@inria.fr

†Inria Saclay Île-de-France, frederic.chazal@inria.fr

‡Inria Saclay Île-de-France, steve.oudot@inria.fr

§University of Connecticut, don.r.sheehy@gmail.com

2 Contributions

Originally introduced in Euclidean spaces [4], the *distance to a measure* can be defined on any metric space $\mathbb{X}$ and for any mass $m \in [0, 1]$. In practice, the data is given as a finite point set P, and we use the empirical measure on this space μ_P, which gives (1) where $k = mn$ is an integer and $p_i(x)$ is the i^{th} nearest neighbor of x in P. The more general definition can be found in [1].

$$d_{\mu_P,m}(x) = \sqrt{\frac{1}{k}\sum_{i=1}^{k} d_{\mathbb{X}}(x, p_i(x))^2} \qquad (1)$$

The distance to a measure is known to be stable with respect to the Wasserstein distance in Euclidean spaces. This can be generalised to any metric space and for μ and ν two probability measures.

$$||d_{\mu,m} - d_{\nu,m}||_\infty \leq \frac{1}{\sqrt{m}} W_2(\mu, \nu) \qquad (2)$$

The stability makes the distance to a measure usable for inference purposes. However, the computation of persistence diagrams requires the use of sublevel sets filtrations. While the sublevel sets of the distance to an empirical measure can be described by a union of $n^{\Theta(d)}$ balls in Euclidean spaces and approximation using the *witnessed k-distance* $d^W_{\mu_P,m}$ is possible [7], this can not be extended to other metric spaces. We provide a new approximation function $d^P_{\mu_P,m}$:

$$d^P_{\mu_P,m}(x) = \min_{p \in P} \sqrt{d_{\mu_P,m}(p)^2 + d_{\mathbb{X}}(x, p)^2} \qquad (3)$$

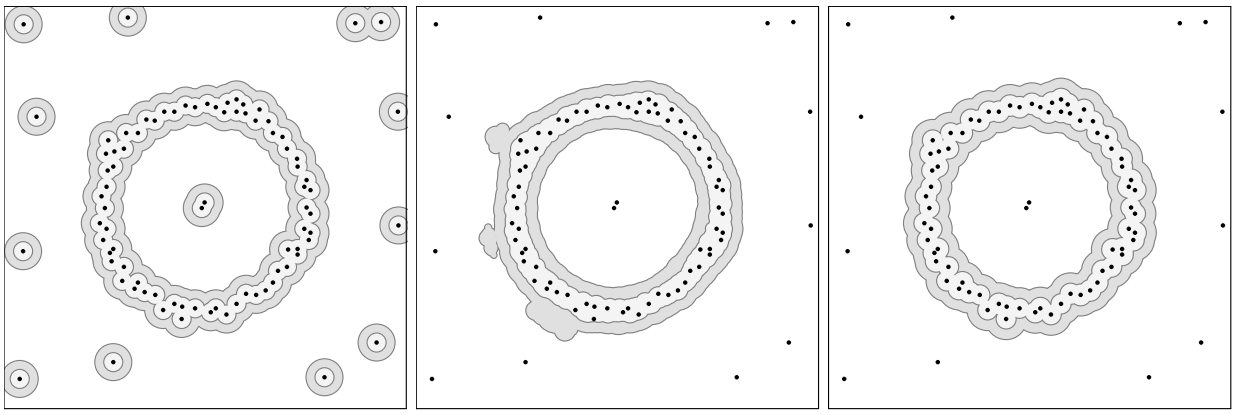


Figure 1: From left to right, two sublevel sets for d_P, $d_{\mu_P,m}$, and $d^P_{\mu_P,m}$ with $m = \frac{3}{|P|}$. The first is too sensitive to noise and outliers. The second is smoother, but substantially more difficult to compute. The third is our approximation, which is robust to noise, efficient to compute, and compact to represent.

The sublevel sets of this function are a union of $O(|P|)$ balls, the same size as the witnessed k-distance. Moreover, the approximation guarantees, independent of the input, are the same when both functions are defined.

$$\frac{1}{\sqrt{2}} d_{\mu_P,m} \leq d^P_{\mu_P,m} \leq \sqrt{3} d_{\mu_P,m} \text{ if } \mathbb{X} \text{ is Euclidean} \quad (4)$$

$$\frac{1}{\sqrt{2}} d_{\mu_P,m} \leq d^P_{\mu_P,m} \leq \sqrt{5} d_{\mu_P,m} \text{ otherwise} \quad (5)$$

The result of the approximation is a power distance. Its sub-level sets are unions of balls with different radii. The persistence of these kinds of functions can be computed using a weighted variant of the usual Vietoris-Rips filtration. However, the complexity problem is the same as the one of the Vietoris-Rips complex. We propose a linear-size approximation to the weighted Vietoris-Rips filtration for intrinsically low-dimensional metric spaces that comes with quality guarantees.

As long as the weights in the power distance are t-Lipschitz with respect to the ambient metric, it gives a multiplicative interleaving between the two Vietoris-Rips complexes and thus a stability result for the persistence diagrams in logarithmic scale. Writing d_B^{ln} the bottleneck-distance in loragithmic scale and D, D' the persistence diagrams of the weighted Vietoris-Rips filtration and its sparsification respectively, we obtain for an arbitrary sparsification factor $\epsilon < 1$,

$$d_B^{ln}(D, D') \leq \ln\left(\frac{1 + \sqrt{1+t^2}\epsilon}{1-\epsilon}\right) \quad (6)$$

The combination of the approximation of the distance to a measure and the sparsification of the weighted Vietoris-Rips complex provides a complete approximation scheme for computing the persistence diagram of the distance to an empirical measure. Along with the theoretical guarantees, it provides the first practical algorithm to infer the persistence from a point cloud using the distance to a measure and real data in a reasonable time complexity. Experimentation provide good results and a significant reduction of computation time.

References

[1] Mickaël Buchet, Frédéric Chazal, Steve Y. Oudot, and Donald R. Sheehy. Efficient and robust topological data analysis on metric spaces. *arXiv preprint arXiv:1306.0039*, 2013.

[2] Gunnar Carlsson. Topology and data. *Bull. Amer. Math. Soc.*, 46:255–308, 2009.

[3] Frédéric Chazal, David Cohen-Steiner, and André Lieutier. A sampling theory for compact sets in euclidean space. *Discrete & Computational Geometry*, 41(3):461–479, 2009.

[4] Frédéric Chazal, David Cohen-Steiner, and Quentin Mérigot. Geometric inference for probability measures. *Foundations of Computational Mathematics*, 11(6):733–751, 2011.

[5] Tamal K. Dey, Fengtao Fan, and Yusu Wang. Computing topological persistence for simplicial maps. *arXiv preprint arXiv:1208.5018*, 2012.

[6] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 454–463. IEEE, 2000.

[7] Leonidas Guibas, Dmitriy Morozov, and Quentin Mérigot. Witnessed k-distance. *Discrete & Computational Geometry*, 49(1):22–45, 2013.

[8] Partha Niyogi, Stephen Smale, and Shmuel Weinberger. Finding the homology of submanifolds with high confidence from random samples. *Discrete & Computational Geometry*, 39(1-3):419–441, 2008.

[9] Steve Y. Oudot and Donald R. Sheehy. Zigzag zoology: Rips zigzags for homology inference. In *Proceedings of the 29th annual Symposium on Computational Geometry*, pages 387–396, 2013.

[10] Donald R. Sheehy. Linear-size approximations to the Vietoris-Rips filtration. *Discrete & Computational Geometry*, 49(4):778–796, 2013.

[11] Afra Zomorodian and Gunnar Carlsson. Computing persistent homology. *Discrete & Computational Geometry*, 33(2):249–274, 2005.

Handling column accumulation in persistent homology computations.

Hubert Wagner*

Abstract

Persistent homology is one of the most practical tools in computational topology. However, to be widely applied in real world situations, it must handle large amounts of data, counted in billions (10^9) of cells and more. We outline a new technique which significantly enhances practical efficiency of the standard matrix reduction algorithm for computing persistent homology and its recent variations. To this end we engineered a new data-structure an to efficiently perform column accumulation. Our method is validated experimentally using the PHAT library.

Introduction. Persistent homology is becoming one of the most widely applicable tools in the emerging field of computational topology. For an introduction to the theory and applications, please see a recent survey [5] and the references within. Persistent studies the evolution of *homology*, or holes, of sublevel sets of a function defined on this object. Intuitively, we observe how these holes are *born* and consecutively *die*. The longer the lifetime, the more *persistent* a given feature is.

Contribution. We focus on column accumulation in matrix reduction algorithm for computing persistent homology. In particular, we show that careful implementation of this operation can yield significant speedups in practice. The technique we propose is based on our new data-structure, called a BitTree. It was engineered specifically for this task, but in fact it is a general-purpose bitset implementation. The details of this data-structure will be published separately.

Unlike usual bitset implementations, BitTree supports fast insertion, deletion, iteration as well as maximum and emptiness queries in worst case $O(log_{64}N)$ time, for subsets of $\{0, 1 \dots N-1\}$. Importantly, it only requires $\approx 2N$ bits of memory. While van Emde Boas trees [3] achieve sub-logarithmic times of crucial operations, our synthetic test show that solution is faster in most practical situations. In this extended abstract, we show that incorporating BitTree into a state-of-the-art library for persistent homology computations yields significant efficiency improvement.

*Jagiellonian University, Krakow, Poland, hub.wag@gmail.com.
Research supported by the Foundation for Polish Science IPP Programme "Geometry and Topology in Physical Models".

Algorithm. The standard algorithm for persistent homology reduces an *ordered boundary matrix* of the input *complex* [4]. The matrix encodes the boundary relations between cells of different dimensions, which build the complex. This boundary matrix is usually sparse and stored as an array of lists. Each list contains the indices of non-zero entries of a single column.

The matrix is reduced by performing left-to-right column additions, to remove all *collisions*. A collision occurs if two columns have the same lowest nonzero position, called lowest-ones. By definition, a *reduced matrix* has no collisions. Lowest ones of a reduced matrix fully encode the information about persistent homology.

Recently, a number of modifications were introduced, aiming at improving the efficiency of the original algorithm [2, 1]. The various, orthogonal, techniques are listed below.

1. Column storage and accumulation:
   - list: uses parallel scan to merge two linked-lists in linear time [4]. The non-zero entries of each column of a matrix are stored in a linked-list.
   - vector: same as above, but uses automatically growing vectors [3] for efficiency [6] which are faster in practice.
   - BST: uses a binary search tree such as red-black tree [3] to accumulate columns [2]. Unlike [2], we store columns as vectors.
   - BitTree: same as above but uses the BitTree data-structure. See Alg. 1, especially lines 9 to 15 and 21.
2. Twist:
   - Y: performs the twist optimization [2]. The columns are reduced in decreasing order of dimension (of the corresponding cell), see Alg. 1, line 4. Moreover, at line 18 point we know that the column with index $I.max()$ represent the boundary of a cycle. Therefore, we can set this column to zero, and avoid reducing it later.
   - N: standard algorithm [4]. All columns are explicitly reduced, which can be costly.
3. Dualization: computing persistent *co*homology is equivalent in practice. However, it can be more efficient, especially for so-called Rips complexes with

dimension cap [1]. This is done as a preprocessing, essentially transposing the input matrix. The last example in Table 1 uses this technique.

Implementation. Algorithm 1 uses the twist optimization and BitTree to handle column additions. Importantly, directly translating the pseudo-code to a language like C++ yields an efficient solution.

Experiments. To assess performance, we use the PHAT (Persistent Homology Algorithm Toolkit) library [7], which is a state-of-the-art C++ library for persistent homology computations and implements various techniques. Most of the used datasets are available as part of the library. Table 1 shows experimental results of the matrix reduction for a range of examples. Different combinations of techniques the used.

While the twist technique is not our contribution, we note that it can increase the efficiency a hundred times and is very simple to implement. The dualization technique is very useful in certain cases. Importantly, it improved the scaling of the last example from quadratic to roughly linear in the number of cell of the complex.

As for our contribution, using the proposed BitTree technique is always the most efficient choice. The speedup factor ranged from 1.3 to 12.1. We suspect that it would be more significant for more complicated examples. Apart from the performance gains, using BitTree reduces the number of control parameters to be tuned by a user. For this reason, it was incorporated into the PHAT library as the default column addition strategy. Using all the techniques in conjunction gives a speedup reaching the order of thousands, compared to the standard implementation. While none of these techniques improves the theoretical worst-case complexity, they are important in practical applications of persistent homology.

Table 1: Comparison of reduction time for different strategies.

dataset	nr cells	column	twist	time[s]
mixed	16M	BitTree	Y	**1.1**
		BST	Y	1.5
		list	Y	23.1
		BitTree	N	72.1
		list	N	2523.6
neptune	56M	BitTree	Y	**15.6**
		BST	Y	21.6
		vector	Y	3354.3
high-genus	2.7M	BitTree	Y	**35.6**
		BST	Y	165.0
		vector	Y	413.7
text-cohom	1.4M	BitTree	Y	**1.6**
		BST	Y	19.4
		vector	Y	24.6
		any	N	$> 10^5$

References

[1] U. Bauer, M. Kerber, J. Reininghaus, Clear and Compress: Computing Persistent Homology in Chunks, *TopoInVis*, 2013.

[2] C. Chen, M. Kerber, Persistent homology computation with a twist, *27th European Workshop on Computational Geometry (EuroCG 2011)*, 2011.

[3] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, Introduction to Algorithms, Third Edition, MIT Press, 2009.

[4] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification, *Discrete & Computational Geometry*, 28(4):511–533, 2002.

[5] H. Edelsbrunner, D. Morozov, Persistent Homology: Theory and Practice, Congress of Mathematics Krakow, 27 July 2012, preprint

[6] H. Wagner, C. Chen, E. Vuçini, Efficient computation of persistent homology for cubical data, *Topological Methods in Data Analysis and Visualization II*, pp. 91–106, 2012.

[7] PHAT: Persistent Homology Algorithm Toolkit, https://code.google.com/p/phat/

Algorithm 1 Compute reduced matrix using BitTree.

Input: Ordered binary matrix M of size $N \times N$, maximum dimension D
Output: Reduced binary matrix R

```
L = zero array of size N
R = vector of N empty columns
I = BitTree representing an empty column of size N
for d = D to 1 do
  for i = 1 to N do
    if M[i] has dimension d then
      copy M[i] to I
      while I ≠ ∅ and L[I.max()] ≠ 0 do
        lowest-one = I.max()
        colliding-column = R[L[lowest-one]]
        for e in colliding-column do
          if I.member(e) then
            I.erase(e)
          else
            I.insert(e)
      if I ≠ ∅ then
        L[I.max()] = i
        R[I.max()] = zero column
      for e in I do
        R[i].add(e)
      I.clear()
return R
```

Inverse-Beacon Guarding

Michael Biro*

Abstract

We consider an extension of beacon guarding, called *inverse-beacon guarding*. Beacons are points in a given polygon P that can be activated to effect a "magnetic pull" on objects in the polygon. Upon activation of a beacon b, an object p in P moves in order to greedily minimize its Euclidean distance to b. We say that b attracts p if p's greedy movement will eventually cause it to reach b. An *inverse-beacon guard* is a point in a polygon that guards with the notion of "inverse attraction", that is, a point b in a polygon P is inverse-beacon guarded by a point p if a beacon placed at b attracts p. We show that inverse-beacon guarding is computationally difficult and give art gallery type bounds on the number of guards sometimes necessary and always sufficient to inverse-beacon guard a polygon.

1 Introduction

The beacon model in this paper has been studied in a number of previous works. In previous papers the focus was on using beacons to aid routing between any two given points of P, as well as guarding a polygon with beacons. Biro et. al. [1] studied the combinatorics of *beacon coverage*, i.e. the minimum number of beacons that attract all of P. In this paper we discuss the related notion of the combinatorics of *inverse-beacon coverage*, i.e. the minimum number of guards in P so that at least one may be attracted by any beacon placement in P.

First we recall the description of beacon attraction from [1]. In our model, a beacon can occupy a point location in the interior or on the boundary of a polygon P. When a beacon is activated, an object p in P moves along a straight line toward b until either it reaches b or makes contact with the boundary of P, ∂P. If contact is made with ∂P, p will follow along ∂P as long as its Euclidean distance to b decreases monotonically. p may alternate between moving in a straight line path toward b in the interior of P and following along ∂P. If p is unable to move so that its distance to b decreases monotonically, we say p is 'stuck' and has reached a local minimum at a dead point. If p gets stuck and doesn't reach b, we say that b does not attract p, and we say that a beacon b *attracts* p if p will eventually reach b.

*Department of Mathematics and Statistics, Williams College
michael.j.biro@williams.edu

For the related idea of inverse-beacon coverage, we take a polygon P, and points p and b in P. We say that p *inverse-beacon guards* b if p is attracted by a beacon placed at b.

In the sensor network view of beacon attraction, (i.e. greedy routing of messages) guarding with beacons corresponds to placing stations such that at least one of them can receive a greedily routed message from any point in P. On the other hand, inverse-beacon guarding corresponds to placing stations such that any point can receive a greedily routed message from at least one station.

2 Results

We begin by stating that, like many other art gallery type problems [2, 3], inverse-beacon guarding is APX-hard. The reduction is from the Minimum Line Covering Problem [4], and the proof may be found in [5].

Theorem 1 *The* Inverse-Beacon Art Gallery Problem *is APX-hard.*

Next, we show bounds on the number of guards sometimes necessary and always sufficient to inverse-beacon guard a given polygon, for simple polygons, polygons with holes, and orthogonal polygons. Note that the portion of the polygon that is inverse-beacon guarded by a point is a superset of the portion of the polygon that is guarded by the point with standard visibility (i.e p sees q if the line segment pq lies in P). Therefore, bounds on the number of points that are always sufficient to visibility guard a polygon (as in [6, 7, 8]) automatically apply to inverse-beacon guards. While it might seem that this containment would mean inverse-beacons are significantly stronger than standard visibility, this is not the case with simple polygons where the sometimes necessary and always sufficient bounds are exactly $\lfloor \frac{n}{3} \rfloor$. However, the strength of inverse-beacons appears to show in polygons with holes and orthogonal polygons, where the best sometimes necessary bounds found are lower than the always sufficient bounds given by traditional visibility.

We modify the classic Chvátal comb polygon, creating internal angles greater than 270°, to construct a polygon with many witness beacons that must be inverse-beacon guarded independently. See Figure 1.

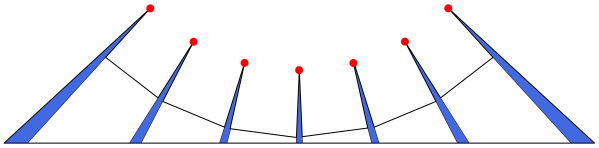


Figure 1: $\lfloor \frac{n}{3} \rfloor = \lfloor \frac{21}{3} \rfloor = 7$ points are necessary. The disjoint shaded regions correspond to the points that are attracted to beacons at the tip of each spike.

Theorem 2 *$\lfloor \frac{n}{3} \rfloor$ points are sometimes necessary and always sufficient to inverse-beacon guard a simple polygon with n vertices.*

In previous work [1] it was conjectured that adding holes to a polygon was actually beneficial in terms of beacon guarding, in the sense that holes decrease the worst-case number of guards necessary for polygons with n vertices. The same pattern appears for inverse-beacon guarding. We construct a polygon with n vertices and a collection of double-wedge shaped holes to show a lower bound of $\lfloor \frac{n}{3} \rfloor - 1$ guards, see Figure 2.

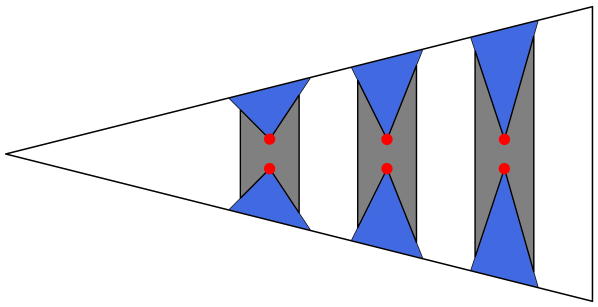


Figure 2: $\lfloor \frac{n}{3} \rfloor - 1$ points are sometimes necessary. The disjoint shaded regions correspond to the points that are attracted to beacons at the reflex vertices of each hole.

By appending two of the modified Chvátal comb polygons in Figure 1 to the polygon with holes in Figure 2, this bound can be improved to $\lfloor \frac{n-1}{3} \rfloor$ guards.

Theorem 3 *Given a polygon P with n vertices and $h \geq 1$ holes, $\lfloor \frac{n-1}{3} \rfloor$ points are sometimes necessary to inverse-beacon guard P, while $\lfloor \frac{n+h}{3} \rfloor$ points are always sufficient to inverse-beacon guard P.*

For orthogonal polygons, we construct a two-sided comb of 'T' spikes, see Figure 3. Ignoring end effects, each spike has 12 vertices and adds an additional two independent witness beacons. However, when n is small there are not enough vertices to properly construct the comb. A case analysis shows that $\lfloor \frac{n+4}{6} \rfloor$ are sometimes necessary for all $n > 14$.

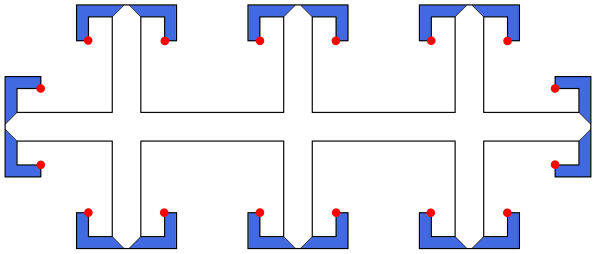


Figure 3: $\lfloor \frac{n+4}{6} \rfloor$ points are sometimes necessary. The disjoint shaded regions correspond to the points that are attracted to beacons at the far vertices of each spike.

Theorem 4 *Given an orthogonal polygon P with $n > 14$ vertices, $\lfloor \frac{n+4}{6} \rfloor$ guards are sometimes necessary to inverse-beacon guard P while $\lfloor \frac{n}{4} \rfloor$ are always sufficient.*

For orthogonal polygons and polygons with holes, we conjecture that our sometimes necessary bounds are also always sufficient.

References

[1] M. Biro, J. Gao, J. Iwerks, I. Kostitsyna, and J. S. Mitchell, "Combinatorics of beacon-based routing and coverage," in *Proc. of the 25th Canadian Conference on Computational Geometry (CCCG 2013)*.

[2] D. Lee and A. Lin, "Computational complexity of art gallery problems," *IEEE Transactions on Information Theory*, vol. 32, no. 2, pp. 276–282, 1986.

[3] D. Schuchardt and H. Hecker, "Two NP-hard art-gallery problems for ortho-polygons," *Mathematical Logic Quarterly*, vol. 41, no. 2, pp. 261–267, 1995.

[4] B. Brodén, M. Hammar, and B. Nilsson, "Guarding lines and 2-link polygons is APX-hard," in *Proceedings of the 13th Canadian Conference on Computational Geometry (CCCG 2001)*.

[5] M. Biro, "Beacon-based routing and guarding," Dissertation, Stony Brook University, 2013.

[6] V. Chvátal, "A combinatorial theorem in plane geometry," *Journal of Combinatorial Theory, Series B*, vol. 18, no. 1, pp. 39–41, 1975.

[7] F. Hoffmann, M. Kaufmann, and K. Kriegel, "The art gallery theorem for polygons with holes," in *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science (FOCS 1991)*.

[8] J. Kahn, M. Klawe, and D. Kleitman, "Traditional galleries require fewer watchmen," *SIAM Journal on Algebraic Discrete Methods*, vol. 4, no. 2, pp. 194–206, 1983.

Hierarchical distance-based aggregation

Mukulika Ghosh* Nancy M. Amato*

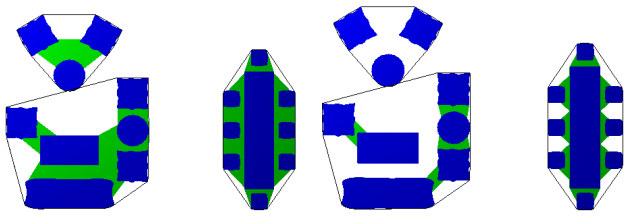
Abstract

Approximation is one of the key techniques used in manipulating geometric objects. Aggregation is a form of approximation that groups objects together as a single entity. This can improve the efficiency of future processing, reduce complexity and generate levels of detail. We present a general framework to aggregate nearby objects together. Grouped objects are approximated into shapes similar to alpha shapes and reduce the space covered as compared to a convex hull approximation. As traditional alpha shapes fail to adapt to non-uniform inputs, we use a function to adaptively determine the value of alpha. By varying the threshold distance that determines object groups, an aggregation hierarchy can be created for any environment. Our experiment shows that the aggregates created by our method have less approximation error than either convex hulls or typical alpha shapes.

1 Introduction

Approximation is one of the key techniques used in manipulating geometric objects. Various approximation methods such as decomposition [6], simplification [4] and clustering [7] are used to improve efficiency and reduce complexity for future applications [2, 8]. Most approximation methods simplify large complex geometric objects.

Aggregation is a form of approximation that groups objects together as a single entity. In this work, we group nearby disjoint objects into a single object. Our goal is that the structure of the aggregated object should be as similar as possible to the original. This can be important in applications such as motion planning in which corridors in the space are important but would be removed by a convex hull computation (Fig. 1). Hence, instead of using a convex hull, we approximate the aggregated shape in a manner similar to alpha shapes [3]. Traditional alpha shapes do not adapt to local variations in point density. Therefore, we use a function defined on the objects' properties to determine the alpha adaptively for bridging the disconnected objects.



(a) Alpha shape aggregation (b) Our aggregation

Figure 1: Aggregation of objects (shown in blue) using a distance threshold of 30. Aggregated objects are shown in green and the convex hulls as black boundaries.

By varying the threshold distance δ that determines object groups in the input environment, an aggregation hierarchy can be created for the environment. This also provides an insight how the free space is evolved in the environment as shown in Fig. 3.

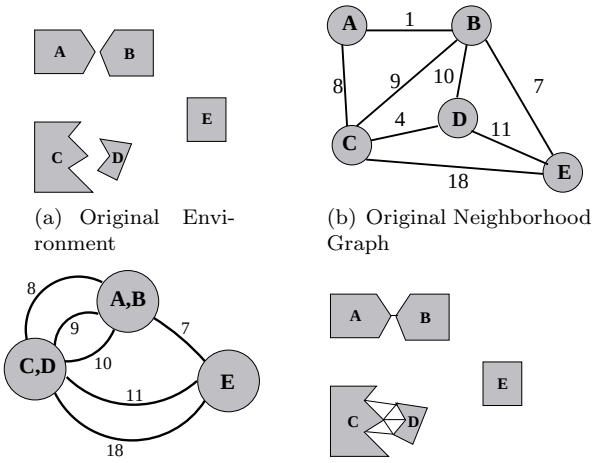
As shown in Fig. 1 (a room plan with furnitures as objects), our method (Fig. 1(b)) generates more compact aggregated shapes with less approximation error (measured in terms of volume of free space consumed in creation of the approximates) than alpha shapes (Fig. 1(a)) or convex hulls (black boundaries). More examples may be found in our website [1] and technical report [5].

2 Our Approach

Our hierarchical distance-based aggregation consists of two major steps: (1) identifying the groups of nearby objects and (2) creating an approximate shape that covers each group of objects. We present an overview of these steps here. Algorithmic details can be found in our technical report [5] and our website [1].

To identify the groups of nearby objects, we abstract the environment in a graph called a *neighborhood graph* where the objects correspond to vertices and weighted edges connect neighboring objects and have weights equal to the shortest distance between the objects. For the implementation details of our neighborhood graph construction method refer [5, 1]. Fig. 2(b) shows an example of the neighborhood graph for the environment in Fig. 2(a). For a given threshold distance δ that determines proximity of the objects, the edges with weight less than δ are collapsed such that the end-vertices are merged to a single vertex. The resulting neighborhood graph (Fig. 2(c)) contains the vertices that represent the aggregated groups of models (Fig. 2(d)).

*Parasol Labarotory, Department of Computer Science & Engineering, Texas A&M University, {mghosh,amato}@cs.tamu.edu.
This research is supported in part by NSF awards CNS-0551685, CCF-0833199, CCF-0830753, IIS-0916053, IIS-0917266, EFRI-1240483, RI-1217991, by NIH NCI R25 CA090301-11, by DOE awards DE-AC02-06CH11357, B575363, by Samsung, Chevron, IBM, Intel, Oracle/Sun and by Award KUS-C1-016-04, made by King Abdullah University of Science and Technology (KAUST).



(a) Original Environment (b) Original Neighborhood Graph

(c) Aggregated Neighborhood Graph (d) Aggregated Environment

Figure 2: An environment and corresponding neighborhood graph before and after aggregation.

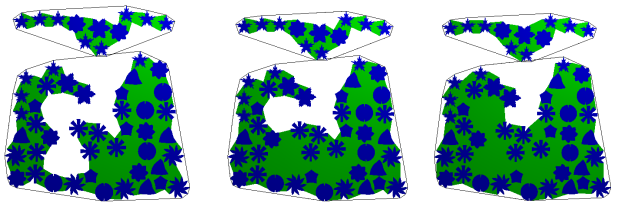
We use method similar to alpha shape construction to build the approximated shapes of the aggregated objects. Instead of using a constant α throughout the environment, we adapt α as returned by a function called *aggregate function*. Aggregate function balances the threshold distance δ with the variation in width of the free passage between the objects bridged by the approximated shape. Hence, α is adjusted considering the local topological variation. (For details refer [5, 1].)

3 Results

We evaluated our approach in terms of quality. For quality we use the circularity measure that states how close the resulting model is to a circle. It is the ratio of the total area to the square of the total perimeter of all objects in the environment. The circularity measure is then normalized to that of the original environment.

As shown in Fig. 1 (the room plan), using the convex hull as the shape descriptor of the aggregated objects requires more space and sometimes results in intersection of the resulting shapes (black boundaries in Fig. 1). The traditional alpha shape approximation (Fig. 1(a)) fails to include topological variation of the free space boundaries and creates more approximation error or wider connections than our method (Fig. 1(b)). The blocked space between the chairs in the dining area (right part of room plan) in Fig. 1(a) is opened in Fig. 1(b). The circularity measure for the environment using convex hull approximation is 5.31, using alpha approximation is 3.40 and using our method is 2.97. More results are provided in [1, 5]. Hence our method creates more compact connections with less approximation error as compared to convex hull or alpha shapes.

Varying the distance threshold, creates a hierarchy of



(a) δ=40 (b) δ=50 (c) δ=60

Figure 3: Evolution of free space using hierarchy of aggregation by varying distance threshold

aggregates (Fig. 3) which shows the evolution of free-space in the environment.

4 Conclusion

This work provides a general framework to aggregate a set of nearby disjoint objects. The algorithm includes grouping of nearby objects based on a threshold distance and approximating the shape of the aggregated models as determined by a function of the objects to be merged and the threshold distance. The results validate the algorithm and show better quality aggregation than traditional alpha shapes and convex hulls.

This aggregation framework can be used along with an object decomposition algorithm to manipulate environments used in various applications including motion planning, computer graphics and video games.

References

[1] https://parasol.tamu.edu/groups/amatogroup/research/app-cd/aggregate/.

[2] T. Brinkhoff and H.-P. Kriegel. Approximations for a multi-step processing of spatial joins. In *IGIS'94*, pages 25–34. Springer, 1994.

[3] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, 29(4):551–559, 1983.

[4] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *SIGGRAPH '97*, pages 209–216. ACM Press/Addison-Wesley Publishing Co., 1997.

[5] M. Ghosh and N. Amato. Distance-based aggregation. Technical Report TR14-006, Parasol Lab., Dept. Comp. Sc. & Engg., Texas A&M University, College Station, Apr. 2014.

[6] A. Golovinskiy and T. Funkhouser. Randomized cuts for 3d mesh analysis. In *SIGGRAPH Asia 2008*, pages 145:1–145:12, 2008.

[7] R. Liu and H. Zhang. Segmentation of 3d meshes through spectral clustering. In *PG 2004*, pages 298–305, 2004.

[8] J. Rossignac and P. Borrel. *Multi-resolution 3D approximations for rendering complex scenes*. Springer, 1993.

Flipping Edge-Labelled Triangulations*

Prosenjit Bose Anna Lubiw Vinayak Pathak Sander Verdonschot

Abstract

We initiate the study of the flip distance between two edge-labelled triangulations of a point set or graph. During a flip, one edge is removed and one new edge is added. With edge labels, the new edge inherits the label of the removed edge. We give tight worst case bounds of $\Theta(n \log n)$ on the flip distance between edge-labelled triangulations of a convex polygon, and edge-labelled combinatorial triangulations, in contrast to the $\Theta(n)$ bounds for the unlabelled cases. We also prove bounds for simultaneous flips on edge-labelled triangulations.

1 Introduction

Our paper is about *reconfiguration* of triangulations. The basic operation for reconfiguring triangulations, both in the combinatorial and the geometric setting, is the *flip* operation that removes one edge of the triangulation and adds the other diagonal of the resulting quadrilateral. In order to obtain a new triangulation, there is a constraint on the removed edge. In the geometric setting a triangulation of a point set is a maximal set of non-crossing edges joining pairs of points. The constraint on a flip is that the two faces incident to the removed edge must form a convex quadrilateral. In the combinatorial setting a triangulation is a maximal planar graph with the clockwise order of edges around each vertex specified. The constraint on a flip is that the other diagonal of the quadrilateral should not already be an edge of the triangulation.

The fundamental property of flips is that they can be used to reconfigure any triangulation to any other triangulation that has the same size and, in the geometric case, the same point set—this last result follows from the fact that every triangulation can be flipped to the Delaunay triangulation.

Flips in triangulations of a convex polygon (equivalently, a set of points in convex position) are especially interesting because they correspond exactly to rotations in a binary tree [11] and flip distance corresponds exactly to *rotation distance* [4].

In this paper we initiate the study of *edge-labelled flips*. If the edges of a triangulation are labelled and we perform a flip, the newly added edge is assigned the label of the removed edge. In particular, this means that the set of edge labels is preserved throughout any flip sequence. In general (for point sets) it is not possible to flip between any two edge-labelled triangulations, but this is possible for combinatorial triangulations and for triangulations of convex polygons, the settings that we consider in this paper. Our initial motivation was to understand the complexity of computing the flip distance between two triangulations of a convex polygon, a problem which is not known to be NP-complete or in P. Is the problem difficult because we do not know which edge flips to which edge? Having this information is the same as having a labelling of the edges.

Background: The *flip distance*—the minimum number of flips to transform one triangulation to another—is at most $6n$ for combinatorial triangulations [3]. For point sets the worst case bound is $\Theta(n^2)$ in general [7] and $2n-10$ for convex polygons [11]. The complexity of computing the exact flip distance between two given triangulations is open in the combinatorial setting and for convex polygons, but was recently proved NP-hard [8, 9] (even APX hard [9]) for general point sets and for simple polygons [1].

The idea of performing flips in parallel was introduced by Hurtado et al. [6], see also [5]. In the geometric setting, a set of edges may be *simultaneously flipped* if each edge may be flipped and no two of the edges are incident to the same face. $O(\log n)$ simultaneous flips are sufficient and sometimes necessary to reconfigure one triangulation of a convex polygon to another. Bose et al. [2] explored simultaneous flips in the combinatorial setting—in this case a simultaneous flip may be performed even if some edges in the set cannot be individually flipped.

2 Our Results

We prove tight $\Theta(n \log n)$ bounds on the worst-case flip distance.

Theorem 1 *Any edge-labelled triangulation of a convex polygon can be flipped to any other using $O(n \log n)$ flips and there exist two edge-labelled triangulations that require $\Omega(n \log n)$ flips to be transformed into one another.*

This contrasts with the $\Theta(n)$ bounds for unlabelled flips [3, 11]. The extra $\log n$ factor arises from sorting-

*A full version of the paper can be found at http://arxiv.org/abs/1310.1166.

related issues. We prove the upper bound by reducing to the problem of sorting a list using an operation that reverses a (possibly non-contiguous) subsequence at a cost proportional to the minimum length of a contiguous sublist containing the subsequence. Then, given two edge-labelled triangulations, we first ignore their labels and transform both of them using $O(n)$ flips to a canonical triangulation where all diagonals share an endpoint. Imagining the shared endpoint to be the topmost vertex of the polygon and reading out the labels of the diagonals from left to right, we get a permutation of the labels; using our reduction on this permutation gives us the desired flip sequence.

A special case of our model of sorting has been considered before. In particular, if we constrain the subsequence we want reversed to be contiguous we get the "length-weighted" reversals model introduced by Pinter and Skiena [10], which has applications in comparative genomics. Our model seems more powerful since the best-known bound for sorting in their model is $O(n \log^2 n)$. Our $\Omega(n \log n)$ lower bound generalizes theirs. Our result provides an efficient $O(\log n)$-factor approximation algorithm to compute the flip distance between edge-labelled triangulations of a point set in convex position.

Finally, we consider *simultaneous* flips. We prove the following bounds.

Theorem 2 *Any edge-labelled triangulation of a convex polygon can be transformed into any other using $O(\log^2 n)$ simultaneous flips and there exist two triangulations where $\Omega(\log n)$ simultaneous flips are required.*

Once again, it is interesting to contrast this with the $\Theta(\log n)$ [2] bound in the unlabelled case. This is the most difficult result in the paper. Using ideas from the non-simultaneous case and emulating quick sort only gives us an upper bound of $O(\log^3 n)$. To reduce it to $O(\log^2 n)$, we provide a sequence of $O(\log n)$ simultaneous flips for the "separation" step of quick sort, that is, we show that given any edge-labelled triangulation, there exists a sequence of $O(\log n)$ simultaneous flips that puts all diagonals with a label less than $n/2$ on the left half of the convex polygon and the other diagonals on the right half. Recursing on both halves simultaneously gives us the desired bound.

The bounds in Theorem 1 also apply to edge-labelled combinatorial triangulations. Once again, we first ignore labels and use $O(n)$ flips to convert both triangulations into a canonical triangulation where we then rearrange the labels. An obvious choice for the canonical triangulation is a Hamiltonian one since it can then be treated as the union of two triangulations of a convex polygon—the Hamiltonian cycle being the boundary of the polygon and the diagonals inside and the diagonals outside forming the two triangulations. We use a slightly different canonical triangulation, which we call a *double wheel*, where we have better control over the interactions between the boundary of the polygon and its two sets of diagonals.

References

[1] O. Aichholzer, W. Mulzer, and A. Pilz. Flip distance between triangulations of a simple polygon is NP-complete. In *European Symposium on Algorithms (ESA)*, volume 8125 of *LNCS*, pages 13–24. Springer, 2013.

[2] P. Bose, J. Czyzowicz, Z. Gao, P. Morin, and D. R. Wood. Simultaneous diagonal flips in plane triangulations. In *Proc. 17th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '06, pages 212–221, 2006.

[3] P. Bose, D. Jansens, A. van Renssen, M. Saumell, and S. Verdonschot. Making triangulations 4-connected using flips. *Computational Geometry*, 47:187197, 2014.

[4] K. Culik II and D. Wood. A note on some tree similarity measures. *Inform. Process. Lett.*, 15(1):39–42, 1982.

[5] J. Galtier, F. Hurtado, M. Noy, S. Perennes, and J. Urrutia. Simultaneous edge flipping in triangulations. *Int. J. Comput. Geometry Appl.*, 13(2):113–133, 2003.

[6] F. Hurtado, M. Noy, and J. Urrutia. Parallel edge flipping. In *Canadian Conference on Computational Geometry*, CCCG, pages 26–27, 1998.

[7] F. Hurtado, M. Noy, and J. Urrutia. Flipping edges in triangulations. *Discrete and Computational Geometry*, 22:333–346, 1999.

[8] A. Lubiw and V. Pathak. Flip distance between two triangulations of a point set is NP-complete. In *Canadian Conference on Computational Geometry*, CCCG, pages 119–124, 2012.

[9] A. Pilz. Flip distance between triangulations of a planar point set is APX-hard. *Computational Geometry*, 14:589–604, 2014.

[10] R. Pinter and S. Skiena. Sorting with length-weighted reversals. In *Proc. 13th International Conference on Genome Informatics*, GIW '02, pages 173–182, 2002.

[11] D. D. Sleator, R. E. Tarjan, and W. P. Thurston. Rotation distance, triangulations, and hyperbolic geometry. *J. Amer. Math. Soc*, 1:647–681, 1988.

Crossing numbers and characterizations of monotone drawings of K_n*

Martin Balko Radoslav Fulek Jan Kynčl

Abstract

In 1958, Hill conjectured that the minimum number of crossings in a drawing of K_n is exactly $Z(n) = \frac{1}{4}\lfloor\frac{n}{2}\rfloor\lfloor\frac{n-1}{2}\rfloor\lfloor\frac{n-2}{2}\rfloor\lfloor\frac{n-3}{2}\rfloor$. Generalizing the result by Ábrego *et al.* for 2-page book drawings, we prove this conjecture for plane drawings in which edges are represented by x-monotone curves. We further generalize the proof for a few other variants of the crossing number. We also give a combinatorial characterization of several classes of x-monotone drawings of complete graphs using a small set of forbidden configurations.

1 Introduction

In a *drawing D of a graph G* in the plane, the vertices are represented by distinct points and each edge is represented by a simple continuous arc connecting the images of its endpoints. A *crossing in D* is a common interior point of two edges where they properly cross and the number $\mathrm{cr}(D)$ of crossings in D is called the *crossing number of a drawing D*. The *crossing number* $\mathrm{cr}(G)$ *of a graph G* is the minimum of $\mathrm{cr}(D)$, taken over all drawings D of G. A drawing D is called *simple* if no two adjacent edges cross and no two edges have more than one common crossing. It is well known that every drawing of G which minimizes the crossing number is simple. We call a drawing of a graph *semisimple* if adjacent edges do not cross but independent edges may cross more than once.

According to the famous conjecture of Hill [5] (also known as Guy's conjecture), the crossing number of the complete graph K_n on n vertices satisfies $\mathrm{cr}(K_n) = Z(n)$, where $Z(n) = \frac{1}{4}\lfloor\frac{n}{2}\rfloor\lfloor\frac{n-1}{2}\rfloor\lfloor\frac{n-2}{2}\rfloor\lfloor\frac{n-3}{2}\rfloor$. It has been verified for $n \leq 12$ and there are drawings of K_n with exactly $Z(n)$ crossings for each n. [4]

The proofs of all the results mentioned here can be found in the full version of the paper [3].

*The authors were supported by the grant GAČR GIG/11/E023 GraDR in the framework of ESF EUROGIGA program. The first and the third author were also supported by the Grant Agency of the Charles University, GAUK 1262213, and by the grant SVV-2013-267313 (Discrete Models and Algorithms). The third author was also partially supported by ERC Advanced Research Grant no 267165 (DISCONV). The second author gratefully acknowledges support from the Swiss National Science Foundation Grant PBELP2_146705.

2 Monotone crossing number of K_n

A curve in the plane is *x-monotone* if it is intersected by every vertical line in at most one point. A drawing of a graph G in which every edge is represented by an x-monotone curve and no two vertices share the same x-coordinate is called *monotone*. The *monotone crossing number* $\text{mon-cr}(G)$ *of a graph G* is the minimum of $\mathrm{cr}(D)$, taken over all monotone drawings D of G. The drawings of complete graphs with $Z(n)$ crossings obtained by Blažek and Koman [4] are 2-*page book* drawings. In such drawings the vertices are placed on a line l and each edge is fully contained in one of the half-planes determined by l. Since 2-page drawings are obviously monotone, we have $\text{mon-cr}(K_n) \leq Z(n)$.

Ábrego *et al.*[1] recently proved that Hill's conjecture holds for 2-page book drawings of complete graphs. We generalize their techniques and show that Hill's conjecture holds for all monotone drawings of K_n.

Theorem 1 *We have* $\text{mon-cr}(K_n) = Z(n)$ *for all n.*

Independently, the authors of [1] also achieved this result [2] using the same techniques, i.e., generalizing the proof for 2-page book drawings. The key idea of the proof is to consider a generalized notion of k-edges, used already in [1]. For a semisimple drawing D of K_n and distinct vertices u and v of K_n, let γ be the oriented arc representing the edge $\{u, v\}$. If w is a vertex of K_n different from u and v, then we say that w is *on the left (right) side of γ* if the topological triangle uvw with vertices u, v and w traced in this order is oriented counter-clockwise (clockwise, respectively). A *k-edge*, $0 \leq k \leq \lfloor n/2 \rfloor - 1$, is an edge $\{u, v\}$ of D that has exactly k points on the same side (left or right). Let $E_k(D)$ denote the *number of k-edges in D*.

Using a double-counting argument, it is possible to express $\mathrm{cr}(D)$ in terms of $E_k(D)$ so that, to find a lower bound for $\mathrm{cr}(D)$, one can just use a lower bound for $E_k(D)$. However, bounds for $E_k(D)$ which would guarantee $\mathrm{cr}(K_n) \geq Z(n)$ do not have to hold in general. Fortunately, it turns out that for simple monotone drawings D one can find sufficiently strong bounds for the weighted sum $\sum_{i=0}^{k}(k+1-i)E_i(D)$, which is usually denoted as $E_{\leq\leq k}(D)$ [1, 2, 3]. Considering the expression of $\mathrm{cr}(D)$ in terms of $E_{\leq\leq k}(D)$, the following bound implies Theorem 1.

Theorem 2 *Let $n \geq 3$ and let D be a semisimple monotone drawing of K_n. Then for every k satisfying $0 \leq k < n/2 - 1$, we have $E_{\leq\leq k}(D) \geq 3\binom{k+3}{3}$.*

3 Combinatorial Characterization

We develop a combinatorial characterization of monotone drawings as a generalization of order types of planar point sets. Let T_n be the set of ordered triples (i, j, k), $1 \leq i < j < k \leq n$, and let Σ_n be the set of *signatures* $\sigma\colon T_n \to \{-, +\}$.

Let D be a monotone drawing of the complete graph $K_n = (V, E)$ with vertices $v_1, v_2, \ldots, v_n$ such that their x-coordinates satisfy $x(v_1) < x(v_2) < \cdots < x(v_n)$. We assign a signature $\sigma \in \Sigma_n$ to the drawing D according to the following rule. For every edge $e = v_i v_k \in E$ and every integer j, $i < j < k$, set $\sigma(i, j, k) = -$ if the point v_j lies above the arc representing the edge e and $\sigma(i, j, k) = +$ otherwise. If $\sigma \in \Sigma_n$ is obtained in this way from a drawing D, then we say that σ is *realized* by D.

Note that we can find a monotone drawing D realizing σ for every given $\sigma \in \Sigma_n$. However, this does not have to hold if D is required to be simple or semisimple. In Theorem 3 we give a full characterization of signatures that can be realized by (semi)simple monotone drawings. To prove the "if part" of Theorem 3, we take a monotone drawing D that realizes σ and has the minimum possible number of crossings.

For integers a, b, c, d, $1 \leq a < b < c < d \leq n$, signs $\xi_1, \xi_2, \xi_3, \xi_4 \in \{-, +\}$ and a signature $\sigma \in \Sigma_n$, we say that the 4-tuple (a, b, c, d) *is of the form* $\xi_1\xi_2\xi_3\xi_4$ *in* σ if $\sigma(a, b, c) = \xi_1$, $\sigma(a, b, d) = \xi_2$, $\sigma(a, c, d) = \xi_3$, and $\sigma(b, c, d) = \xi_4$. For a sign $\xi \in \{-, +\}$ we use $\overline{\xi}$ to denote the opposite sign.

Theorem 3 *A signature $\sigma \in \Sigma_n$ can be realized by a semisimple monotone drawing if and only if every 4-tuple of indices from $[n]$ is of one of the forms $++++, ----, ++--, --++, -++-, +--+, ---+, +++-, +---, -+++$ in σ. The signature σ can be realized by a simple monotone drawing if, in addition, there is no 5-tuple (a, b, c, d, e) with $a < b < c < d < e$ such that $\sigma(a, b, e) = \sigma(a, d, e) = \sigma(b, c, d) = \overline{\sigma(a, c, e)}$.*

Pseudolinear drawings of K_n can be characterized similarly by further restricting the conditions on σ.

Theorem 4 *A signature $\sigma \in \Sigma_n$ can be realized by a pseudolinear monotone drawing if and only if every ordered 4-tuple of indices from $[n]$ is of one of the forms $++++, +++-, ++--, +---, ----, ---+, --++, -+++$ in σ.*

We have used Theorem 3 in a computer search to find all simple monotone drawings of K_n, for $n \leq 10$, with minimum number of crossings. See [3].

4 Generalizations

We further strengthen Theorem 1 to more general drawings of K_n. We show that mon-ocr$_+(K_n) = Z(n)$ where mon-ocr$_+(K_n)$ denotes the smallest number of pairs of edges that cross an odd number of times in a monotone semisimple drawing of K_n. If we allow adjacent edges to cross an even number of times, then the result still remains true. The most general class of drawings for which Theorem 1 is known to hold are so-called *weakly semisimple s-shellable* drawings (see [2] and [3]).

It would be interesting to see if techniques similar to those used in the proof of Theorem 1 can be used to prove Hill's conjecture for general drawings of K_n. The same approach does not generalize to all drawings. For example, a so-called *cylindrical drawing* of K_{10}, with crossing number $Z(10)$, does not satisfy the lower bound from Theorem 2.

Extrapolating the definition of $E_{\leq\leq k}(D)$, we define the number $E_{\leq\leq\leq k}(D) := \sum_{j=0}^{k} E_{\leq\leq j}(D) = \sum_{i=0}^{k} \binom{k+2-i}{2} E_i(D)$. Then, similarly as before, we can express cr(D) using $E_{\leq\leq\leq k}(D)$. We conjecture that the following lower bound on $E_{\leq\leq\leq k}(D)$ is satisfied by all simple drawings of complete graphs.

Conjecture 1 *Let $n \geq 3$ and let D be a simple drawing of K_n. Then for every k satisfying $0 \leq k < n/2 - 1$, we have $E_{\leq\leq\leq k}(D) \geq 3\binom{k+4}{4}$.*

Conjecture 1 is stronger than Hill's conjecture. According to Theorem 2 it is true for all simple monotone drawings. All our examples of simple drawings of complete graphs, including the cylindrical drawings, also satisfy this conjecture. More details can be found in the full version [3] of the paper.

References

[1] B. M. Ábrego, O. Aichholzer, S. Fernández-Merchant, P. Ramos and G. Salazar, The 2-page crossing number of K_n, *Discrete Comput. Geom.* **49**(4) (2013), 747–777.

[2] B. M. Ábrego, O. Aichholzer, S. Fernández-Merchant, P. Ramos and G. Salazar, Shellable drawings and the cylindrical crossing number of K_n, arXiv:1309.3665v2, 2013.

[3] M. Balko, R. Fulek and J. Kynčl, Crossing numbers and combinatorial characterization of monotone drawings of K_n, submitted, arXiv:1312.3679v2, 2013.

[4] J. Blažek and M. Koman, A minimal problem concerning complete plane graphs, in: *Theory of Graphs and its Applications, Proc. Sympos. Smolenice, 1963*, 113–117, Publ. House Czechoslovak Acad. Sci., Prague, 1964.

[5] R. K. Guy, A combinatorial problem, *Nabla (Bulletin of the Malayan Math. Soc.)* **7** (1960), 68–72.

Detecting Weakly Simple Polygons

Hsien-Chih Chang* Jeff Erickson* Chao Xu*

Abstract

A closed curve in the plane is weakly simple if it is the limit (in the Fréchet metric) of a sequence of simple closed curves. We describe an algorithm to determine whether a closed walk of length n in a simple plane graph is weakly simple in $O(n \log n)$ time, improving an earlier $O(n^3)$-time algorithm of Cortese *et al.* [*Discrete Math.* 2009]. As an immediate corollary, we obtain an algorithm to determine whether a given n-vertex polygon is weakly simple in $O(n^2 \log n)$ time.

1 Introduction

Simple polygons in the plane have been standard objects of study in computational geometry for decades (and in the broader mathematical community for centuries). Many algorithms designed for simple polygons continue to work with little or no modification in degenerate cases, where intuitively the polygon overlaps itself but does not cross itself. We offer the first efficient algorithm to detect such degenerate polygons.

Formally, a closed curve in the plane is a continuous function $P\colon S^1 \to \mathbb{R}^2$. A closed curve is *simple* if it is injective, and *weakly simple* if for any $\varepsilon > 0$, there is a simple closed curve $\tilde{P}$ whose Fréchet distance from P is at most ε. A recent nontrivial result of Ribó Mor [4, Theorem 3.1] (conjectured by Connelly *et al.* [1]) implies that a *polygon* P with at least three vertices is weakly simple if and only if, for any $\varepsilon > 0$, we can obtain a simple polygon by perturbing each vertex of P within a ball of radius ε. See Figure 1 for an example.

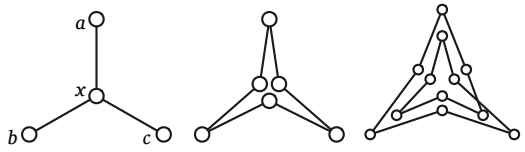


Figure 1. $axbxcxa$ is weakly simple; $axbxcxaxbxcxa$ is not.

Unfortunately, neither of these definitions imply efficient algorithms to determine whether a given polygon is weakly simple. Several authors have offered alternative characterizations that suggest efficient algorithms; unfortunately, *none* of these characterizations is consistent with the formal definition. For example, Toussaint [6, 7] defined a polygon P to be weakly simple if it has winding number ± 1 and no pair of subpaths of P have a "proper crossing". However, winding numbers are not well defined for polygons with *spurs*: vertices whose two incident edges overlap. It is not difficult to determine whether a polygon *without spurs* is weakly simple in $O(n)$ time, but determining the weak simplicity of *arbitrary* polygons seems much more difficult.

In fact, we can detect weakly simple polygons (with or without spurs) in polynomial time using an algorithm of Cortese *et al.* [2] that checks whether a closed walk of length n in a plane graph is weakly simple (in their terminology, a *rigid clustered cycle*) in $O(n^3)$ time. If a given polygon P contains two edges that cross, then P cannot be weakly simple; we can detect this condition in $O(n \log n)$ time using a standard sweep-line algorithm. Otherwise, the image of P defines a planar straight-line graph G, whose nodes are the distinct vertices of P, and whose edges are maximal segments between nodes. A single node in G may represent several coincident vertices of P. Again, G can be computed in $O(n \log n)$ time using a standard sweep-line algorithm. The polygon P coincides with a walk W of length $O(n^2)$ in this graph; the complexity increase is caused by vertices in the interior of many overlapping collinear edges. It follows immediately that we can determine whether any polygon is weakly simple in $O(n^6)$ time.

In this paper, we show that the running time of the algorithm of Cortese *et al.* can be reduced from $O(n^3)$ to $O(n^2)$ with more careful analysis, and then to $O(n \log n)$ with more careful bookkeeping within the algorithm. It then follows immediately that we can determine whether a given polygon is weakly simple in $O(n^2 \log n)$ time.

2 Edge Expansion

We now sketch the algorithm of Cortese *et al.* [2], necessarily omitting many details due to space constraints.

Let W be a closed walk of length n in a plane graph G. Let v be an arbitrary node in G and let uv be one of its incident edges. We call uv a ***base*** of v if each occurrence of v in the walk W is immediately preceded or followed by the other endpoint u. The algorithm of Cortese *et al.* begins with an $O(n)$-time preprocessing phase that either modifies both the walk W and the graph G so that every vertex has a base, or determines correctly that W is not weakly simple.

The algorithm then repeatedly applies an operation we call ***edge expansion***, which again either determines correctly that W is not weakly simple or modifies both the

*Department of Computer Science, University of Illinois at Urbana-Champaign. {hchang17,jeffe,chaoxu3}@illinois.edu. Supported in part by NSF grant CCF-0915519.

walk W and the underlying graph G, preserving the weak simplicity of W and the invariant that every vertex has a base. We call an edge uv of G ***expandable*** if (1) uv is a base for both endpoints u and v, and (2) uv is the *only* base for at least one of those two nodes. If no edges of G are expandable, then either G has been reduced to nothing (in which case the original walk W is weakly simple), or G is a simple cycle and W has no spurs (in which case W is weakly simple if and only if it traverses G exactly once).

The expansion of an expandable edge uv can be defined geometrically as follows. Consider an ellipse C with u and v inside and all other nodes outside, that intersects precisely the edges incident to either u or v but not both. Because edges of G are straight line segments, C intersects each edge at most once. We subdivide each edge up (where $p \neq v$) with a new node $[up]$ at the intersection point $up \cap C$; we similarly subdivide edges incident to v. Finally, we modify both G and W by replacing each subwalk of W that lies within the region surrounded by C and starts and ends on C with a straight line segment. In particular, we contract any subwalk that starts and ends at the same point of C to that point; otherwise, these new segments become new edges of G. If any of these new edges cross, the original walk W is not weakly simple; otherwise, all necessary invariants are maintained. See Figure 2 for an example.

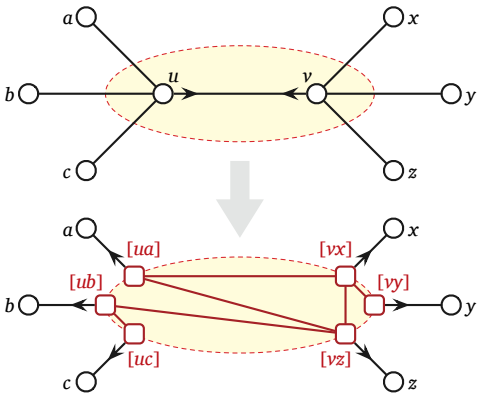


Figure 2. An example of edge expansion. An arrow leaving a node indicates the base of that node.

3 Analysis

The running time of the algorithm is dominated by edge expansions, so it suffices to analyze their total running time. Let $\boldsymbol{w(uv)}$ denote the number of times that W traverses the edge uv in G. We represent the walk W itself as a circular doubly-linked list of edges; each edge uv also maintains a circular doubly-linked list of the $w(uv)$ edges of W that traverse it. With this representation, any expandable edge uv can be expanded in $O(w(uv)) = O(n)$ time, essentially by brute force. In an actual implementation, the entire operation is performed combinatorially by querying and modifying the rotation system of the embedding of G; it is not necessary to compute an explicit ellipse C, or even to compute coordinates for the new nodes $[ux]$ and $[vy]$.

Cortese *et al.* [2] use a potential argument to show that the algorithm terminates after at most $O(n^2)$ edge expansions, and therefore runs in $O(n^3)$ time. We can improve their analysis by a factor of n by redefining the potential $\boldsymbol{\Phi(W,G)}$ to be the number of vertices of W minus the number of nodes in G. We always have $0 \leq \Phi(W,G) \leq n$, because every edge of G is traversed at least once. Exhaustive case analysis implies that each edge expansion decreases $\Phi(W,G)$ by at least 1.

To improve the running time further, we *reuse* the nodes u and v and the edge uv instead of deleting them. Let up^* be a non-base edge incident to u with largest weight $w(up^*)$, breaking ties arbitrarily. Instead of creating a new node $[up^*]$, we simply move node u to the point $up^* \cap C$. Similarly, instead of creating a new node $[vq^*]$ for the heaviest non-base edge vq^* incident to v, we simply move v to the point $vq^* \cap C$. After these moves, subwalks of W from up^* to vq^* either contain spurs (which can be removed in $O(1)$ time each) or are already single line segments and therefore require no additional work. Spur removals can be charged to the resulting decrease in potential $\Phi(W,G)$; otherwise, with careful bookkeeping, the time for the edge expansion is only $O(w(uv) - w'(uv))$, where $w'(uv)$ denotes the number of times W traverses uv *after* the edge expansion. A heavy-path decomposition argument [3, 5] now implies that the total time spent on edge expansions (except for spur removals) is only $O(n \log n)$. Again, this dominates the overall running time of the algorithm.

References

[1] Robert Connelly, Erik D. Demaine, and Günter Rote. Infinitesimally locked self-touching linkages with applications to locked trees. *Physical Knots: Knotting, Linking, and Folding of Geometric Objects in* $\mathbb{R}^3$, 287–311, 2002. American Mathematical Society.

[2] Pier Francesco Cortese, Giuseppe Di Battista, Maurizio Patrignani, and Maurizio Pizzonia. On embedding a cycle in a plane graph. *Discrete Mathematics* 309(7):1856–1869, 2009.

[3] Dov Harel and Robert Endre Tarjan. Fast algorithms for finding nearest common ancestors. *SIAM J. Comput.* 13(2):338–355, 1984.

[4] Ares Ribó Mor. *Realization and Counting Problems for Planar Structures: Trees and Linkages, Polytopes and Polyominoes*. Ph.D. thesis, Freie Universität Berlin, 2006.

[5] Daniel D. Sleator and Robert Endre Tarjan. A data structure for dynamic trees. *J. Comput. Syst. Sci.* 26(3):362–391, 1983.

[6] Godfried T. Toussaint. Computing geodesic properties inside a simple polygon. *Revue d'Intelligence Artificielle* 3(2):9–42, 1989.

[7] Godfried T. Toussaint. On separating two simple polygons by a single translation. *Discrete Comput. Geom.* 4(1):265–278, 1989.

Unfolding k-Monotone Linear Trees

Ching-Hao Liu * Sheung-Hung Poon **

Abstract

We study the problem of unfolding k-monotone linear trees in 2D. In this paper, we show that a 3-monotone linear tree can be flattened in $O(n)$ time and $O(n)$ moves, and a 4-monotone linear tree can be flattened in $O(n \log n)$ time and $O(n)$ moves, where n is the number of edges of the tree.

1 Introduction

A (*planar*) *linkage* is a graph in which every edge e is assigned some positive real number ℓ_e so that in any embedding in 2D (also called *configuration*) of the graph, e is embedded as a straight line segment of length ℓ_e. A configuration of a linkage L is called *linear* if all vertices and all edges of L lie on a line. A *k-monotone linear tree* T is a linear configuration of a tree linkage so that any vertical line intersects T in at most k vertices and edges. For linear linkages, we use the *self-touching assumption* as the theoretical basis [3]. We allow linkage edges to touch or overlap, but not cross. That is, if the interiors of edges intersect, then they must be parallel. A *move* is a continuous monotonic change of an angle between two edges incident to some vertex. A *motion* is composed of one move or several simultaneous moves. A tree linkage T is said to be *flattened* if there exists a series of motions on the plane to reconfigure any configuration of T to a linear tree with all root-to-leaf paths going to the right; otherwise, T is called *locked*.

Now we survey related works for monotone or linear tree linkages in 2D. Kusakari et al. [4] defined "monotone trees", where each chain from the root to any leaf is 1-monotone with respect to the x-axis. They showed that any monotone tree can be flattened in $O(n \log n)$ time and $O(n)$ moves. Later, Kusakari [5] proved that "radial monotone trees" can lock, where every directed chain from the root of the tree to a leaf is radially monotone. Moreover, Poon [6] proved that any 2-monotone orthogonal tree can always be flattened in $O(n^2)$ time and moves, and Ballinger et al. [2] showed that 3-monotone orthogonal trees can lock. Thus the minimal monotonicity of locked k-monotone orthogonal trees is 3. Furthermore, Ballinger et al. [2] also showed a locked 7-monotone linear tree of 8 edges and a locked equilateral tree. Abel et al. [1] showed that any linear equilateral tree can always be flattened in polynomial time and moves. In this paper, we present efficient algorithms to flatten 3-monotone and 4-monotone linear trees.

*Department of Computer Science, National Tsing Hua University, chinghao.liu@gmail.com

**Department of Computer Science & Institute of Information Systems and Applications, spoon@cs.nthu.edu.tw

2 Definitions

We give an example of a 4-monotone linear tree T in Figure 1. The *backbone* K of T (see chain $P[s,t]$) is the

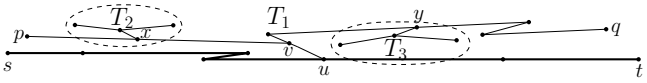


Figure 1: A 4-monotone linear tree T with a backbone $K = P[s,t]$ and a main subtree T_1 rooted at u.

chain from the left end of T to the right end of T. The components obtained from T subtracting K are called the *main subtrees* of T (see subtree T_1 in Figure 1). The *root* of T_1 (see u) is the common vertex of T_1 and the backbone of T. The *main stem* of T_1 (see $P[u,v]$) is the common part of two chains C and C' in T_1, where C (see $P[u,p]$) is the chain from the root of T_1 to the left end of T_1 and C' (see $P[u,q]$) is the chain from the root of T_1 to the right end of T_1. The other end of the main stem of T_1 which is not the root is called the *branching vertex* of T_1 (see v). The *main branches* of T_1 (see $P[v,p]$ and $P[v,q]$) are the two chains from the branching vertex of T_1 to the left and right ends of of T_1, respectively.

A *zigzag* Z is a chain forming a Z-shape or a mirrored Z-shape. The *middle chain* of Z is the straight chain whose two endpoints are the two turn vertices of Z. In the two remaining maximal straight chains of Z, the one incident to the left turn vertex is called the *right chain* of Z, and the other the *left chain* of Z. To *bind* two overlapping edges lying on a line is to glue them together to become one single edge. To *verticalize* a straight chain C around the pivot u in the $(+y)$- or $(-y)$-direction is to rotate C around u until it reaches the corresponding vertical direction. A subtree rooted at r is *canonically verticalized* in the $(+y)$- or $(-y)$-direction if the chain from r to any other leaf of the subtree is a straight chain lying in the direction. Note that the motion of *horizontalizing* and the state of being *canonically horizontalized* in the $(+x)$- or $(-x)$-direction are defined in a similar fashion.

3 Our Results

Theorem 1 *A 3-monotone linear tree can be flattened in $O(n)$ time and $O(n)$ moves.*

We sketch the algorithm for flattening a 3-monotone linear tree T with backbone K as follows. First, we canonically verticalize the main subtrees above K or below K in the $(+y)$ or $(-y)$-direction, respectively. Then we make K a straight chain by straightening each zigzag Z in K one by one from left to right. Finally, we canonically horizontalize all main subtrees one by one from right to left so that they finally all lie in $(+x)$-direction. This completes the flattening of T.

In particular, to verticalize the main subtrees above K, we make use of the covering structure of the main branches. We show that there is always an uncovered main branch, which can then be verticalized. Thus by repeating the branch-verticalization procedure, we eventually can verticalize all main subtrees.

Since we obtain the verticalizing ordering according to the tree structure, this algorithm runs in linear time.

Theorem 2 *A 4-monotone linear tree can be flattened in $O(n \log n)$ time and $O(n)$ moves.*

First, we sketch the procedure for flattening a particular 4-monotone linear tree T with straight backbone K. A horizontal straight chain C in T is called *one-end-free* if one endpoint v of C is a *virtual* leaf; that is, if we treat all verticalized edges and all edges covered by C invisible, then v is a leaf. The strategy of this algorithm is to repeatedly verticalize a one-end-free horizontal straight chain C where only vertical edges can cover C until all main subtrees are canonically verticalized. Thus there are three main tasks to verticalize all main subtrees T' above K as follows.

First, we select suitable disjoint horizontal straight chains from each main subtree T' into a set $\mathcal{C}$. Second, we use a trapezoidal map to store the covering relations between the chains in $\mathcal{C}$, which is in fact a partially ordered set, and we apply a topological sort to the chains in $\mathcal{C}$ to obtain a verticalizing ordering. Third, we verticalize the chains in $\mathcal{C}$ without edge crossings incrementally according to the verticalizing ordering.

Next, we sketch the algorithm for flattening a general 4-monotone linear tree T, which has a zigzagged backbone K. The algorithm uses the above procedure as a subroutine and we straighten the zigzags in K incrementally by considering covering relations as follows. We search for the first zigzag Z in K from left to right such that its middle chain, its right chain and the main subtrees rooted on its right chain can be bound, and then be verticalized without any edge crossing. Then we apply an opening process to Z and other unstraightened zigzags Z' in K on its left one by one from right to left (see Figures 2(a–d)). We further apply another straightening process to straighten these zigzags one by one from left to right (see Figures 2(d–f)). Thus by

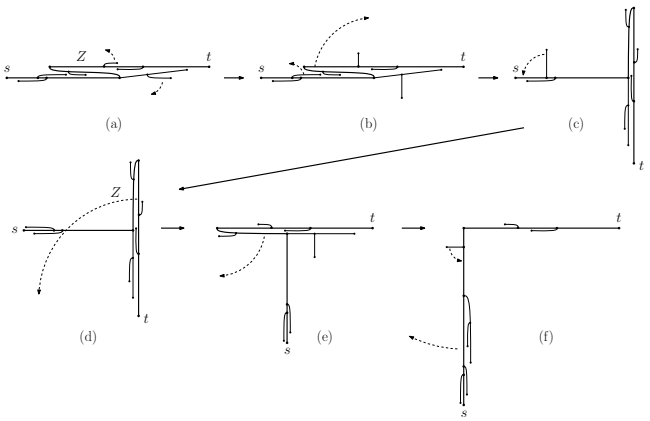


Figure 2: (a–d) The opening process for a zigzag Z of backbone K of T. (d–f) The straightening process for zigzag Z.

repeating the above procedure, we eventually can verticalize all main subtrees and also straighten all zigzags of K.

Since we use the trapezoidal map and the topological sort, this algorithm runs in $O(n \log n)$ time.

References

[1] Z. Able, E.D. Demaine, M.L. Demaine, S. Eisenstat, J. Lynch, T.B. Schardl and I. Shapiro-Ellowitz, Folding Equilateral Plane Graphs. In *Proc. of the 22nd Int. Conf. on Algorithms and Computation*, 574–583, (2011)

[2] B. Ballinger, D. Charlton, E.D. Demaine, M.L. Demaine, J. Iacono, C.-H. Liu and S.-H. Poon, Minimal Locked Trees. In *Proc. of the 11th Int. Symp. on Algorithms and Data Structures*, 61–73, (2009)

[3] R. Connelly, E.D. Demaine and G. Rote, Infinitesimally Locked Self-Touching Linkages with Applications to Locked Trees. *Physical Knots: Knotting, Linking, and Folding of Geometric Objects in R^3*, AMS, 287–311, (2002)

[4] Y. Kusakari, M. Sato and T. Nishizeki, Planar Reconfiguration of Monotone Trees. *IEICE Transactions*, E85–A, 938–943, (2002)

[5] Y. Kusakari, On Reconfiguring Radial Trees. *IEICE Transactions*, 89–A, 1207–1214, (2006)

[6] S.-H. Poon, On Unfolding 3D Lattice Polygons and 2D Orthogonal Trees. In *Proc. of the 14th Annual Int. Conf. on Computing and Combinatorics Conference*, 374–384, (2008)